ugr | Universidad de Granada

Department of Computer Architecture
and Technology

Vision systems for 3D object pose estimation in real-time

Sistemas de visión para el seguimiento de poses 3-D de objetos en tiempo real

PhD Thesis

Leonardo Rubio Navarro

Granada, January 2014

Editor: Editorial de la Universidad de Granada
Autor: Leonardo Rubio Navarro
D.L.: GR 1868-2014
ISBN: 978-84-9083-052-9

ugr | Universidad de Granada

Sistemas de visión para el seguimiento de poses 3-D de objetos en tiempo real

Vision systems for 3D object pose estimation in real-time

Presented By

Leonardo Rubio Navarro

To apply for the

International PhD Degree in Computer and Network Engineering

January 2014

ADVISORS

Eduardo Ros Vidal

Javier Díaz Alonso

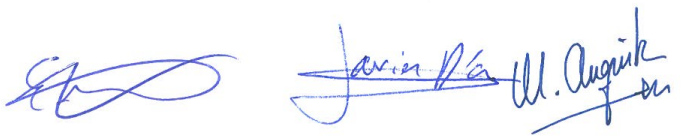
Mancia Anguita López

D. Eduardo Ros Vidal, D. Javier Díaz Alonso y D^{a}. Mancia Anguita López, Catedrático de Universidad, Ayudante Doctor y Ayudante Doctora respectivamente del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada

CERTIFICAN

Que la memoria titulada "Sistemas de visión para el seguimiento de poses 3-D de objetos en tiempo real" ha sido realizada por D. Leonardo Rubio Navarro, bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor Internacional en Ingeniería de Computadores y Redes.

Granada, a 5 de Diciembre de 2013



Fdo. Eduardo Ros Vidal, Javier Díaz Alonso, Mancia Anguita López

Directores de la Tesis

El doctorando Leonardo Rubio Navarro y los directores de la tesis D. Eduardo Ros Vidal, D. Javier Díaz Alonso y Dª. Mancia Anguita López, Catedrático de Universidad, Ayudante Doctor y Ayudante Doctora respectivamente del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada. Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, a 5 de Diciembre de 2013

Fdo. Eduardo Ros Vidal, Javier Díaz Alonso, Mancia Anguita López

Directores de la Tesis

Fdo. Leonardo Rubio Navarro

Doctorando

Editor: Editorial de la Universidad de Granada
Autor: Leonardo Rubio Navarro
D.L: En trámite
ISBN:En trámite

Contents

List of Figures

List of Abbreviations

6D-PF	Six Degrees Particle Filter
AICP	Articulated Iterative Closest Point
CUDA	Compute Unified Device Architecture
DLT	Direct Linear Transformation
DOFs	Degrees Of Freedom
DoG	Difference of Gaussian
FPS	Frames Per Second
GPU	Graphics Processing Unit
HR	Hierarchical Recognition
ICP	Iterative Closest Point
KLT	Kanade-Lucas-Tomasi points
LoG	Laplacian of Gaussian
OpenCV	Open Computer Vision Library
OpenNI	Open Natural Interaction

P3P	Perspective-3-Point
PCL	Point Cloud Library
PF	Particle Filter
PIC	Post-Imposed Constraints
POSIT	Perspective from Orthography and Scaling with ITerations
PnP	Perspective-n-Point Problem
RANSAC	RANdom SAmple Consensus
RGB-D	Red Green Blue - Depth
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVD	Singular Value Decomposition

Abstract

This dissertation presents a work related with image analysis and 3D object pose estimation. A wide variety of object's types and pose estimation methods have been addressed along this PhD manuscript. Multiple pose estimation systems have been implemented with improvements that outperform state-of-the-art methods. The comparison is carried out in real-world and synthetic benchmarks where the ground-truth is known and error measurement mechanisms are provided.

This dissertation is structured in four main parts:

In the first part, an overview of the state-of-the-art is exposed. We review the different camera representations used in computer vision systems. We continue with a description of the different computer vision features and the algorithms for their extraction. In the final part of this section we define the pose estimation algorithms that use the camera's representation and the extracted visual features to compute the object pose estimation.

After the literature's overview, we continue with a second part that states a generic architecture for object pose estimation. This architecture defines a development environment for further contributions. We develop a pose detection system that uses single-frame information and is based on state-of-the-art methods. We combine this system with tracking methods that use temporal information in order to implement a robust and accurate pose estimation system. A comparison with the literature's systems is

carried out using a synthetic benchmark that has been produced specifically for this comparative study.

Following the generic architecture defined in the previous part, the third section explains one of the main contribution related with articulated object detection. A detection system for articulated object is described where features, models and algorithms are adapted for this purpose. A synthetic benchmark for articulated objects was developed along with error measurement methods that allow comparing the system to state-of-the-art algorithms.

Another main contribution of this work are improvements adopted in pose estimation systems that enhance performance in terms of accuracy, execution time and robustness. The improvements involve object's model simplification and environmental knowledge adaptation to the pose estimation process. Using augmented reality systems and depth cameras, we create a benchmark for real-world rigid object tracking that allows the comparison with alternative methods. We also extend the system applications to diverse situations such as camera pose estimation.

In conclusion, this dissertation implements multiple pose estimation systems for different types of objects in real-world scenarios. Proposed systems are compared with alternative methods through the developed benchmarks. The different methods evaluation aims remarkable results.

Resumen

Esta tesis presenta un trabajo relacionado con el análisis de imágenes y la estimación de pose de objeto 3D. A lo largo de esta tesis se hace referencia a una amplia variedad de tipos de objetos y de métodos de estimación de pose. Se ha realizado la implementación de múltiples sistemas de estimación de posición con mejoras que superan a los métodos disponibles en el estado-de-la-técnica. Los métodos son comparados en benchmarks del mundo real y sintéticos donde conocemos la pose de los objetos que se encuentran en la escena. Mediante los mecanismos de medición del error de la pose aportados, podemos realizar un estudio y comparación de los diferentes sistemas.

Esta tesis se estructura en cuatro partes principales:

En la primera parte, se expone una visión general del estado-de-la-técnica. En este apartado, se revisan las diferentes representaciones de cámaras utilizadas en los sistemas de visión por computador. Continuamos con una descripción de las diferentes características (features) de visión por ordenador y los algoritmos empleados para su extracción. En la parte final de esta sección, se definen los algoritmos de estimación de pose que utilizan las representaciones de las cámaras definidas y las características visuales extraídas de la escena, con el fin de calcular la estimación de pose del objeto.

Después de la revisión general de la literatura, se continúa con una segunda parte donde se define una arquitectura genérica para los sistemas de

estimación de pose. Esta arquitectura define un entorno de desarrollo, el cual, es usado para las contribuciones aportadas en este trabajo. En esta sección desarrollamos un sistema de detección de pose donde no se hace uso de información temporal y que se basa en los métodos del estado-de-la-técnica. El sistema desarrollado es combinado con métodos de seguimiento de objetos, que emplean información temporal, con el fin de implementar un sistema de estimación de posición robusto y preciso. Se realiza una comparación con los sistemas disponibles en la literatura mediante un benchmark sintético que ha sido generado para este estudio comparativo.

Tras la arquitectura genérica definida en la parte anterior, en la tercera sección de esta tesis, explicamos una de las contribuciones principales que está relacionada con la detección de objetos articulados. En el sistema de detección de objetos articulados realizamos una adaptación de las características visuales, modelos y algoritmos para este propósito. Al mismo tiempo, un benchmark sintético para objetos articulados fue desarrollado junto con los métodos de medición de error, lo cual permiten la comparación del sistema con los algoritmos de estado-de-la-técnica.

Otra contribución principal de este trabajo está relacionada con las mejoras adoptadas en los sistemas de estimación de pose, que tienen por objeto una mejora de la precisión, el tiempo de ejecución y la estabilidad del sistema de seguimiento. Las mejoras incluyen la simplificación de los modelos de los objetos y la adaptación de información del entorno al proceso de estimación de pose. Utilizando sistemas de realidad aumentada y cámaras de profundidad, se crea un benchmark del mundo real para el seguimiento de objetos, que permite la comparación con otros métodos. También extendemos las aplicaciones del sistema a diversas situaciones, como la estimación de la posición de la cámara.

Como conclusión, esta tesis implementa múltiples sistemas de estimación de pose para diferentes tipos de objetos en escenarios del mundo real. Los sistemas propuestos son comparados con métodos alternativos a través de los benchmark desarrollados. La evaluación de los diferentes métodos proporciona notables resultados.

Chapter 1

Introduction

1.1 Computer Vision

The history of this problem goes back decades beginning with Leonardo da Vinci (1452-1519) who in a very elegant way says:

> *"Perspective is nothing else that the seeing of an object behind a sheet of glass, smooth and quite transparent, on the surface of which all the things may be marked that are behind this glass. All things transmit their images to the eye by pyramidal lines, and these pyramids are cut by the said glass. The nearer to the eye these are intersected, the smaller the image of their cause will appear".*

These words introduce the concept of perspective and flat image plane that led to the development of photogrammetry in the mid-nineteenth century. The photogrammetry is defined as the science of measurement from optics. The photogrammetry is considered the predecessor of modern photography. The evolution of this art and the introduction of computers led to computer vision's creation when in 1957, Gilbert Hobrogh introduces

the first analog machine for corresponding points correlation in two images [1]. Its patent from 1960 is shown in Fig. 1.1.

This domain combines knowledge from computer science, electrical engineering, mathematics, physiology, biology and cognitive science. Computer vision is the science that aims scene understanding inspired by human vision. The scene understanding is carried out from sensed image data using computer software and hardware. The image data can represent many forms of information such as video sequences, views from multiple cameras, depth cameras or multi-dimensional data from medical scanners.

The computer vision discipline is hierarchically structure in three levels of complexity:

- **Low-level Vision:** They are the basic features extracted from images such as edges, corners, optical flow, appearance features, stereovision, depth, etc. Some low-level vision examples are shown in Fig. 1.2.
- **Intermediate-level vision:** This refers to 3D scene interpretation using *low-level vision* features as input. Some interpretations are related with object tracking, face detection, plane segmentation, egomotion, etc [2, 3].
- **High-level vision:** Creates an interpretation of *low-level* and *intermediate-level vision* information. The *high-level* inferred knowledge from the scene and analyzes the activity, interaction and behavior of the different elements of the environment.

The different vision levels are combined with the goal of solving a wide variety of computer vision tasks. These objectives are classified into the following classes:

- **Segmentation:** The image data is divided into meaningful parts that simplify the environmental representation. Tasks such as planes

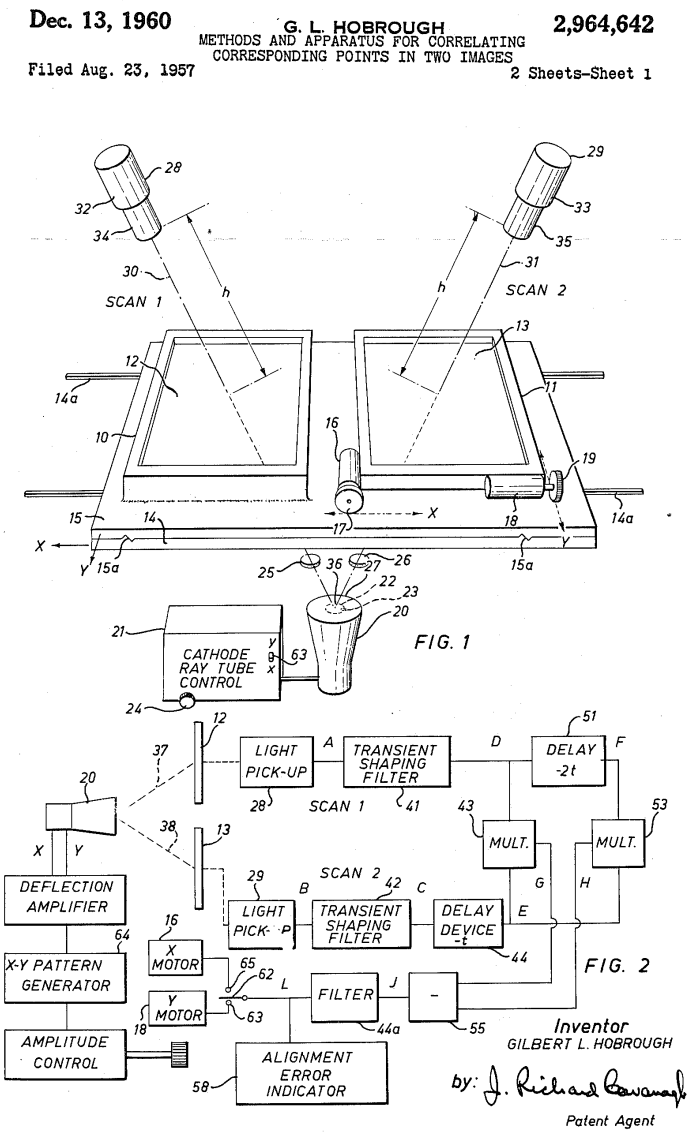


Figure 1.1: Stereo Machine. Gilbert Hobrogh introduces the first machine to locate correlating corresponding points in two images. Pixels are illuminated in both images with a determined displacement. Their intensities are compared. If the intensity values match, they are considered a correspondence. Otherwise, one of the image shifts its position [1].

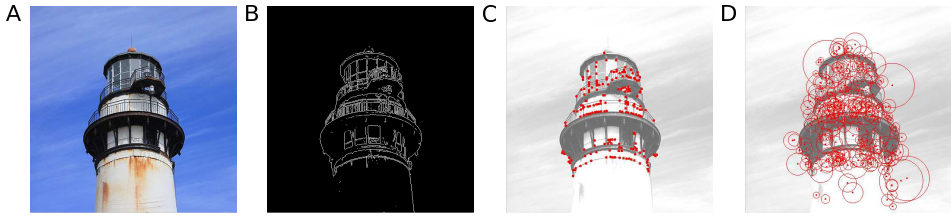


Figure 1.2: Low-level vision. A Original image. B Edge subtraction. C Corner subtraction. D Appearance features.

segmentation, background subtraction or element segmentation, try to classify image regions by associating a particular meaning (planes, objects, static elements, etc.).

- **Reconstruction:** This topic refers to tasks that model the 3D world from image data. Multiple views of an object, light changes in a scene or structural models, provide valuable information that is used for environmental reconstruction.
- **Recognition:** An understanding of the different scene's elements is vital in computer vision. Classification and tracking algorithms give a general idea of what is happening in the scene at every moment.
- **Control:** Accurate environmental control leads to improved decision capabilities that are used in many scopes such as obstacle avoidance, object grasping, planning motion trace, visual servoing, navigation, etc.

Every computer vision concept works together towards the goal of achieving human vision functionality with artificial systems.

1.2 Pose Estimation Recognition

This introduction brings us to the subject of this dissertation that is related to computer vision science. The presented work is focused on the

recognition field and in particular, the study of pose detection and tracking in the 3D world. The pose estimation covers different object's classes by extending the object's topology. This extension allows multiple object tracking such as articulated objects. Pose detection and tracking is used in multiple scenarios and it is also applied in many different situations such as navigation tasks.

3D object detection and tracking is key for a correct scene understanding [4]. This research field is also related with multiple computer vision tasks such as object reconstruction, scene segmentation, feature detection and model generation. All of which are addressed in the course of this thesis.

1.3 Objectives

The scientific objectives defined at the beginning of the thesis were related with object tracking and detection referred in the previous section. In particular, the stated main goals are the following:

- The pose estimation tasks in the dissertation are oriented for a real world environment with real objects. Therefore it is required to implement an application that extract the required object data that is used in its tracking.
- Development and implementation of a platform for synthetic sequence generation. The synthetic sequences have the functionality to recreate scenarios where detection and tracking methods can be tested.
- Study applicability of the state-of-the-art methods in rigid object detection and their applications. The main purpose is to develop a system capable of rigid objet tracking in real-time and in real scenarios.
- Improvement in rigid object detection. A system extension to articulated object detection represents a challenge that requires an adap-

tation from model representation and from the synthetic scene generation.

- Improvement in rigid object tracking. A simplification of the object tracking requirements and a system performance improvement represent a significant advantage in computer vision recognition and for the research project framework in which this thesis is developed.

- Finally, we proposed a study of possible system applications that enforce the advantages of the presented research.

1.4 Projects Framework

This dissertation was developed within the framework of the European project TOMSY, a national (Spanish) project called DINAM-VISION and a Project of Excellence from Junta de Andalusia named VITVIR.

1.4.1 TOMSY

TOMSY, Topology Based Motion Synthesis for Dexterous Manipulation, EU project IST-FP7-Collaborative Project-270436.

Project Description. The aim of TOMSY is to enable a generational leap in the techniques and scalability of motion synthesis algorithms.

We propose to do this by learning and exploiting appropriate topological representations and testing them on challenging domains of flexible, multi-object manipulation and close contact robot control and computer animation.

Traditional motion planning algorithms have struggled to cope with both the dimensionality of the state and action space and generalisability of solutions in such domains. This proposal builds on existing geometric notions of topological metrics and uses data driven methods to discover

multi-scale mappings that capture key invariances - blending between symbolic, discrete and continuous latent space representations. We will develop methods for sensing, planning and control using such representations.

TOMSY, for the first time, aims to achieve this by realizing flexibility at all the three levels of sensing, representation and action generation by developing novel object-action representations for sensing based on manipulation manifolds and refining metamorphic manipulator design in a complete cycle. The methods and hardware developed will be tested on challenging real world robotic manipulation problems ranging from primarily 'relational' block worlds, to articulated carton folding or origami and all the way to full body humanoid interactions with flexible objects.

The results of this project will go a long way towards providing some answers to the long standing question of the 'right' representation in a sensorimotor control and provide a basis for a future generation of robotic and computer vision systems capable of real-time synthesis of motion that result in fluent interaction with their environment.

1.4.2 DINAM-VISION

DINAM-VISION, Spanish national research project, DPI2007-61683.

Project Description. This project aims at developing a real-time vision system, capable of dynamically adapting the inherent characteristics (for example, the dynamic range of the spatio-temporal filters used in the low-level vision) of the used model(s) in order to improve information extraction. First stage of the system deals with low-level visual cues (e.g. local contrast changes and related magnitude, orientation and phase), while in the second stage these primitives are fused into multimodal disperse entities. The system has feed-back loops that allow feeding back information from later stages to the earlier stages, so that optimal functionality at each stage is achieved. Real-time processing is achieved by utilizing massively parallel platforms, such as FPGAs.

The project will explore potential use of the system in different application areas, where the group has expertise, such as driver assistance systems. The system will be tested in both daylight and night-time scenarios, using cameras working in the visible light and infrared wavelengths. We will concentrate on IMOs (independently moving objects) and ego-motion.

1.4.3 VITVIR

The project Tridimensional Vision for Interactive Video Analytics and Augmented Reality (VITVIR), Project of Excellence from Junta de Andalusia (TIC-8120), is focused on the development of a vision system to extract 3D information from a scene by means of information provided by different cameras or 3D sensors such as ToF cameras. The objects in the scene are classified as static (walls, columns, etc.) or dynamic (mobile furniture, people, etc.) in order to resolve the actions to perform in different application fields such as video surveillance, augmented reality or domotics.

1.5 Tools and Methods Used

We define MATLAB as the development environment used to implement and test the different algorithms in this work. MATLAB allows efficient development in a dynamic environment where it is easy the visualization of any type of information. These qualities make MATLAB a perfect environment for computer vision system development.

In order to increase MATLAB potential, we extend implementations with MEX (Matlab EXecuable) code that allows external C/C++ code execution.

Part of our work is based on object modeling and synthetic sequence generation. Therefore, it is required a powerful rendering engine. In this case, we use OpenGL through C++ code. The OpenGL rendering engine can be used in MATLAB by MEX code.

We use Blender as a professional tool for object geometry modeling that we combine with OpenGL in order to generate the required model and sequences.

Native MATLAB code is processed as an interpreted language and therefore its execution is known to be slow. However, MEX code used precompiled sources that have better execution time performance.

Nevertheless, we use MATLAB environment for an initial algorithm development. Since we aim for a real-time execution system, once algorithms performances are tested, we migrate the different systems to C++ code.

For the algorithms implementation, we take advantage of the state-of-the-art methods implemented in OpenCV (Open Computer Vision) library for 2D computer vision algorithm and PCL (Point Cloud Library) for 3D computer vision algorithm. We use these libraries for effective developing or comparison purpose.

We extend our algorithm implementations with CUDA code that allows GPU computation. Executing parts of the algorithm's code in the GPU provides a speed up of the execution time and therefore improved implementation performances.

The real-world sequences are recorded with standard cameras for 2D data information or RGB-D cameras (Microsoft Kinect) for 3D data information. We use OpenCV (2D data) and OpenNI (3D data) as sensing libraries.

All the code integration was carried out with Qt project developer and CMake that ease code compiling, particularly, when there is a high number of dependencies (GPU code, OpenGL code, OpenCV code, PCL, OpenNI, etc.).

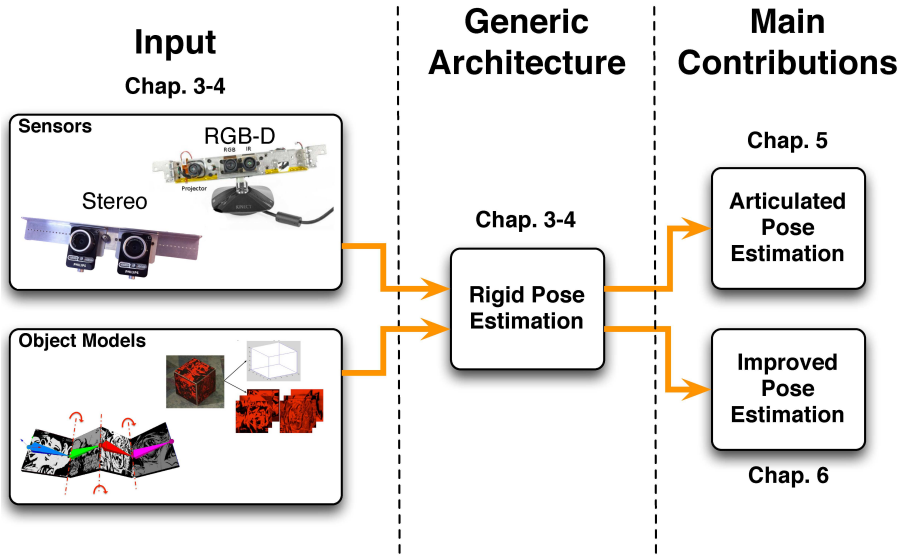


Figure 1.3: Dissertation Structure. This scheme shows the different parts that composed this thesis and the relations between chapters.

1.6 Dissertation Outline

This dissertation is structure in four main parts that represent the evolution of this work:

- Chapter 3 - State-of-the-art: This chapter reviews the computer vision literature in object tracking and detection. The overview analyses the different visual sensors and computer vision cues used in pose estimation. In this chapter is also addressed the state-of-the-art methodology for object pose estimation based on computer vision cues.
- Chapter 4 - Rigid Pose Estimation: After the literature outline, a generic architecture for object tracking is defined in Chap.4. The

generic architecture aims a robust and accurate pose estimation that obtains competitive results. This architecture defines the basis of this work and states a development environment for further contributions. Therefore, the generic architecture shows the evolution and connection between further chapters. The pose estimation scheme described in this chapter defines the input data source, the object model and the pose estimation methodology based on the state-of-the-art study.

- Chapter 5 - Articulated Pose Estimation: Starting from the development environment established with the generic architecture, Chap. 5 describe the improvement in the object class domain. The exposed contribution extends object pose estimation from rigid object to articulated object. This chapter describes the modifications carried out in the generic scheme such as object model adaptations. It is also presented the novel methodology for articulated object detection.

- Chapter 6 - Improved Pose Estimation: In Chap.6 a redesign of the architecture is carried out. Modifications in every pose estimation stage are described in order to improve the pose estimation performance. These improvements also involve object and visual cues simplification, and the incorporation of environmental knowledge to the pose estimation process. In this chapter, the pose estimation methodology moves from object detection to object tracking algorithm in order to aim precise pose estimation.

This structure is described in Fig. 1.3, where the relations between chapters are highlighted. After the four parts, the dissertation ends with a conclusions chapter (Chap.7) where the main contribution are stressed and the future work is proposed.

Capítulo 2

Introducción (Español)

2.1 Visión Artificial

La historia de este problema se remonta décadas, empezando con Leonardo da Vinci (1452-1519), quien, de una manera muy elegante dijo:

> *"La perspectiva no es otra cosa que observar un objeto a través de un trozo de cristal, liso y transparente, en cuya superficie podemos marcar los elementos que vemos tras el cristal. Todas los objetos transmiten su imagen al ojo mediante lineas piramidales, las cuales son cortadas por dicho cristal. Cuanto más cerca al ojo se intercepten las lineas, más pequeña será la imagen formada.".*

Estas palabras introducen el concepto de perspectiva y de plano de imagen, lo cual, llevo al desarrollo de la fotogrametría a mediados del siglo diecinueve. La fotogrametría se define como la ciencia que realiza mediciones a través de la óptica. La fotogrametría es considerada como la predecesora de la fotografía moderna. La evolución de esta ciencia y con la introducción de los ordenadores llevo a la creación de la visión artificial

cuando en 1957 Gilbert Hobrogh introdujo la primera máquina analógica para identificar puntos correlativos en dos imágenes [1]. La patente de 1960 se muestra en la Fig. 2.1.

Este campo de la ciencia combina conocimientos de ciencias de la computación, ingeniería electrónica, matemáticas, phisiología, biología y ciencias cognitivas. La visión artificial es la ciencia que busca el entendimiento del entorno y se inspira en la visión humana. El entendimiento del entorno se lleva a cabo con datos de imágenes y mediante el uso de software y hardware de ordenador. Los datos de imágenes pueden hacer referencia a distintas informaciones de sensores, como por ejemplo, secuencias de video, vistas desde múltiples cámaras, cámaras de profundidad o datos multidimensionales de escáneres médicos.

La disciplina de visión artificial se estructura jerárquicamente en tres niveles de complejidad:

- **Visión de bajo nivel:** Son las características básicas que se extraen de una imagen, como por ejemplo, bordes, esquinas, flujo óptico, características de apariencia, visión estéreo, profundidad, etc. Algunos ejemplos de las características de bajo nivel se muestran en la Fig. 2.2.

- **Visión de nivel intermedio:** Esto hace referencia a la interpretación 3D de la escena empleando las características obtenidas en la *visión de bajo nivel*. Algunas interpretaciones están relacionadas con el seguimiento de objetos, detección de caras, segmentación de planos, ego-motion, etc [2, 3].

- **Visión de alto nivel:** Crean una interpretación de la información obtenida por la *visión de bajo nivel* y *la visión de nivel intermedio*. La *visión de alto nivel* infiere conocimiento de la escena y analiza la actividad, interacción y comportamiento de los distintos elementos del entorno.

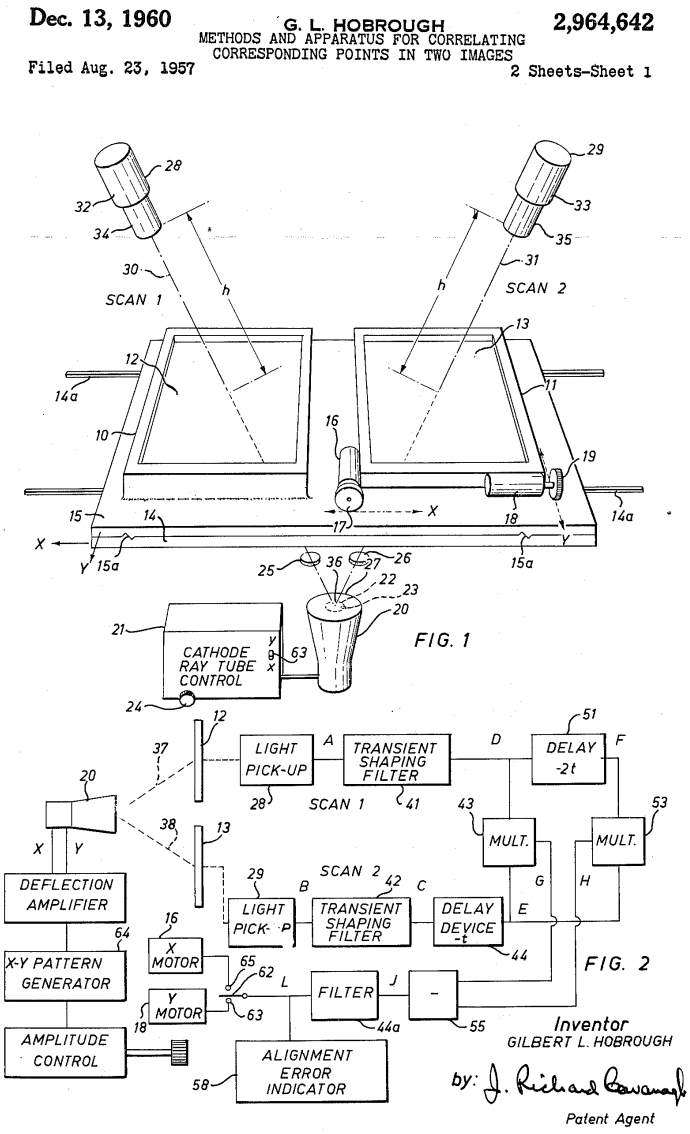


Figura 2.1: Máquina estéreo. Gilbert Hobrogh introdujo la primera máquina para localizar puntos de correspondencia en dos imágenes. Los pixeles son iluminados en ambas imágenes con un desplazamiento determinado. La intensidad es comparada y si los valores coinciden, son considerados una correspondencia. En otro caso, una de las imágenes desplaza su posición [1].

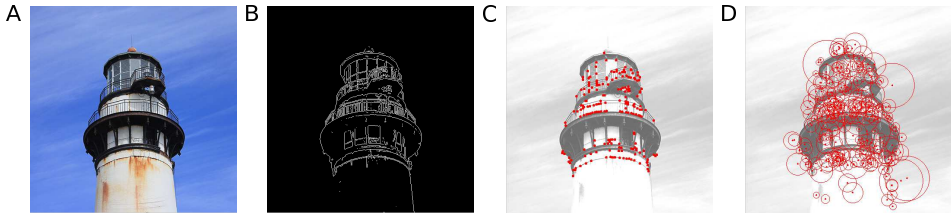


Figura 2.2: Visión de bajo nivel. A Imagen original. B Extracción de bordes. C Extracción de esquinas. D Características de apariencia.

Los distintos niveles de visión se combinan con el propósito de resolver un amplio rango de objetivos. Existe una gran variedad de objetivos que pueden ser clasificados en las siguientes clases:

- **Segmentación:** La información de la imagen es divida en parte con un significado asociado que simplifica la representación del entorno. Tareas como la segmentación de planos, la substracción de fondo o segmentación de elementos, intentan clasificar las regiones de la imagen con un cierto significado (planos, objetos, elementos estáticos, etc.).

- **Reconstrucción:** Este tema hace referencia a tareas de modelado del mundo real a partir de imágenes. Múltiples vistas de un objeto, cambios de iluminación en una escena o modelos estructurales aportan información valiosa que es usada para una reconstrucción del entorno.

- **Reconocimiento:** Un entendimiento de los distintos elementos que componen una escena es vital en visión artificial. Los algoritmos de clasificación y seguimiento dan una idea del tipo de objetos que podemos observar en un entorno y una idea general de qué está pasando a cada momento.

- **Control:** Un control preciso del entorno conlleva una mejora en las capacidades de decisión que son usadas en muchos ámbitos, tales como evitación de obstáculos, manejo de objetos, planificación de rutas de movimiento, control visual, navegación, etc.

Cada uno de los conceptos en visión artificial tiene como objetivo conseguir un sistema artificial con una funcionalidad similar a la conseguida con la visión humana.

2.2 Reconocimiento de la Estimación de Pose

Esta introducción nos lleva al tema tratado en esta tesis, el cual está relacionado con la ciencia de la visión artificial. El trabajo que se presenta, se centra en el campo del *reconocimiento* y en particular, en el estudio de la detección de pose y el seguimiento 3d de objetos en el mundo real. La estimación de pose trabaja con distintas clases de objetos mediante la extensión de la topología del objeto. La detección y seguimiento de poses se usa en diversos escenarios y situaciones, como por ejemplo navegación y localización de la posición de la cámara.

La detección y seguimiento de objetos 3D es clave para una correcta compresión de la escena [4]. Este campo de investigación está relacionado con múltiples tareas de la visión artificial, como por ejemplo, reconstrucción de objetos, segmentación de escenas, detección de características y generación de modelos. A lo largo de esta tesis haremos referencias a todas estas tareas.

2.3 Objetivos

Los objetivos científicos definidos al inicio de esta tesis están relacionados con la detección y seguimiento de objetos que se hace referencia en la sección anterior. En particular, los objetivos principales definidos son los siguientes:

- Las tareas de estimación de pose se orientan a un ámbito en el mundo real donde se trabaje con objetos reales. Por tanto, se requiere la implementación de una aplicación que extraiga los datos necesarios

de los objetos. Dichos datos son empleados para el seguimiento del objeto.

- Desarrollo e implementación de una plataforma para la generación de secuencias sintéticas. Estas secuencias sintéticas poseen la funcionalidad de recrear escenarios donde los métodos de detección y el seguimiento pueden ser probados

- Estudio de la funcionalidad de los métodos del estado-de-la-técnica en detección de objetos y sus aplicaciones. El objetivo principal es el desarrollo de un sistema capaz del seguimiento de objetos rígidos en tiempo real en un entorno real.

- Mejora de la detección de objetos rígidos. Una extensión en el sistema de detección de objetos rígidos a objetos articulados representa un gran desafío que requiere una adaptación del modelo de representación y de la forma de generar escenas sintéticas.

- Mejora en el seguimiento de objetos rígidos. Una simplificación de los requisitos de seguimiento de objetos y una mejora del rendimiento del sistema representan una ventaja importante en el campo de reconocimiento de la visión artificial y en el marco del proyecto de investigación en el que se a desarrollado esta tesis.

- Por último, proponemos un estudio de las posibles aplicaciones del sistema que adapten las ventajas de la investigación presentada a entornos reales.

2.4 Marco de Proyectos

Esta tesis se ha desarrollado en el marco del proyecto europeo TOMSY, un proyecto nacional (español) llamado DINAM-WISION y un proyecto de Excelencia de la Junta de Andalucía llamado VITVIR.

2.4.1 TOMSY

TOMSY, Topology Based Motion Synthesis for Dexterous Manipulation, proyecto europeo IST-FP7-Collaborative Project-270436.

Descripción del Proyecto. El objetivo de TOMSY es permitir un salto generacional en las técnicas y la escalabilidad de los algoritmos de síntesis de movimiento.

Proponemos hacer esto mediante el aprendizaje y la explotación de las representaciones topológicas adecuadas y ponerlas a prueba en ámbitos complejos de flexibilidad, manipulación de varios objetos y un estricto control del contacto del robot y la representación por ordenador.

Algoritmos de planificación de movimientos tradicionales han luchado para hacer frente tanto a la dimensionalidad del espacio de estados, del espacio de acción y la generalización de soluciones en estos ámbitos. Esta propuesta se basa en las nociones geométricas existentes de indicadores topológicos y utiliza datos impulsados métodos para descubrir correlaciones multi-escala que capturan invariantes claves - la mezcla entre representaciones espaciales simbólicas, discreta y continua. Vamos a desarrollar métodos para la detección, planificación y control mediante el uso de dichas representaciones.

TOMSY, por primera vez, tiene como objetivo lograr esto mediante la realización de la flexibilidad en todos los tres niveles de la percepción, la representación y la generación de la acción por el desarrollo de nuevas representaciones de objetos de acción para la detección basados en la manipulación de múltiples elementos y el perfeccionamiento de manipuladores metamórficos. Los métodos y hardware desarrollados se pondrán a prueba en problemas de robótica del mundo real que irán desde problemas simples con objetos rígidos geométricos, a elementos de cartón plegable y articulados y llegando a las interacciones humanas con objetos flexibles.

Los resultados de este proyecto contribuirán en gran medida a proporcionar algunas respuestas a la pregunta de la correcta representación en un control sensoriomotor y servir de base para una futura generación de sis-

temas de visión robótica e informática capaces de realizar síntesis en tiempo real lo cual resultad en una interacción fluida con su entorno.

2.4.2 DINAM-VISION

DINAM-VISION, Proyecto nacional de investigación, DPI2007-61683.

Descripción del Proyecto. Este proyecto tiene como objetivo desarrollar un sistema de visión en tiempo real, capaz de adaptar dinámicamente las características inherentes (por ejemplo, el rango dinámico de los filtros espaciotemporales utiliza en la visión de bajo nivel) del modelos utilizados con el fin de mejorar la extracción de información. La primera etapa de los sistemas trabajo con señales visuales de bajo nivel (por ejemplo, cambios de contraste locales y magnitud relacionada, la orientación y la fase), mientras que en la segunda fase, estas primitivas se combinan en entidades multimodales. El sistema tiene bucles de realimentación que permiten refinar la información de las etapas posteriores, de esta forma se consigue la funcionalidad óptima en cada etapa. Procesamiento en tiempo real se logra mediante la utilización de plataformas de computo paralelo masivo, tales como FPGAs.

El proyecto explorará el uso potencial del sistema en diferentes áreas de aplicación, donde el grupo tiene experiencia, como los sistemas de asistencia al conductor. El sistema se pondrá a prueba tanto en escenarios nocturnos donde se trabaja con cámaras de onda infrarrojas. Nos concentraremos en la detección de IMOs (objetos en movimiento de forma independiente) y ego-motion.

2.4.3 VITVIR

El proyecto Tridimensional Vision for Interactive Video Analytics and Augmented Reality (VITVIR), es un proyecto de Excelencia de la Junta de Andalucía (TIC-8120), se centra en el desarrollo de un sistema de visión para extraer información 3D de una escena a través de la información prove-

niente de diferentes cámaras o sensores 3D tales como cámaras TOF. Los objetos de la escena se clasifican como objetos estáticos (paredes, columnas, etc) o dinámicos (muebles, personas, etc.) con el fin de resolver las acciones a llevar a cabo en diferentes campos de aplicación como la videovigilancia, la realidad aumentada o la domótica.

2.5 Herramientas y Métodos Empleados

Definimos MATLAB como el entorno de desarrollo utilizado para implementar y probar los diferentes algoritmos implementados en este trabajo. MATLAB permite el desarrollo eficiente en un entorno dinámico en el que es fácil la visualización de cualquier tipo de información. Estas cualidades hacen de MATLAB el entorno perfecto para el desarrollo del sistema de visión por computador.

Con el fin de aumentar el potencial de MATLAB, extendemos las implementaciones con código MEX (Matlab EXecuable) que permite la ejecución de código externo C/C++.

Parte de nuestro trabajo se basa en el modelado de objetos y la generación de secuencias sintéticas. Por lo tanto, se requiere un motor de renderizado de altas prestaciones. En este caso usamos OpenGL a través de código C++. El motor de renderizado de OpenGL se puede acceder desde MATLAB mediante código MEX.

Utilizamos Blender como una herramienta profesional para el modelado de objetos geométricos que combinamos con OpenGL con el fin de generar los modelos y las secuencias virtuales.

El código nativo de MATLAB se procesa como un lenguaje interpretado. Esto implica que su ejecución se realice de manera menos eficiente. Sin embargo, el código MEX utiliza fuentes precompilados que tienen un mejor rendimiento durante la ejecución.

Sin embargo, utilizamos el entorno MATLAB para un desarrollo inicial de los algoritmos. Dado que nuestro objetivo es un sistema de ejecución en

tiempo real, una vez que son comprobadas las prestaciones de los algoritmos, migramos los diferentes sistemas a código C++.

Para la implementación de algoritmos, empleamos los métodos del estado-de-la-técnica implementados en la biblioteca OpenCV (Open Computer Vision) para algoritmos de visión artificial en 2D y PCL (Point Library Cloud) para algoritmos de visión artificial en 3D. Utilizamos estas bibliotecas para el desarrollo eficaz o con el fin de una comparación entre métodos.

Extendemos nuestra implementación de algoritmos con código CUDA que permite la computación en GPU. Ejecutando partes del algoritmo en la GPU proporcionamos una aceleración de los tiempos de ejecución y por lo tanto la mejora de las prestaciones de la implementación.

Las secuencias reales se graban con cámaras estándar de datos 2D o con cámaras RGB-D (Kinect Microsoft) para obtener información 3D de la escena. Utilizamos OpenCV (datos 2D) y OpenNI (datos 3D) como bibliotecas de captura de datos.

Todo la integración de códigos se llevó a cabo con el entorno de desarrollo Qt y CMake que facilitan la compilación de código, sobre todo, cuando hay un gran número de dependencias (código de GPU, código de OpenGL, código de OpenCV, PCL, OpenNI, etc.).

2.6 Estructura de la Tesis

Esta tesis se estructura en cuatro partes principales que representan la evolución de este trabajo:

- Capítulo 3 - Estado-de-la-técnica: En este capítulo se revisa la literatura de visión artificial para el seguimiento y detección de objetos. La revisión general analiza los diferentes tipos de sensores y características de visión artificial utilizados en la estimación de pose. En este capítulo también se aborda la metodología del estado-de-la-

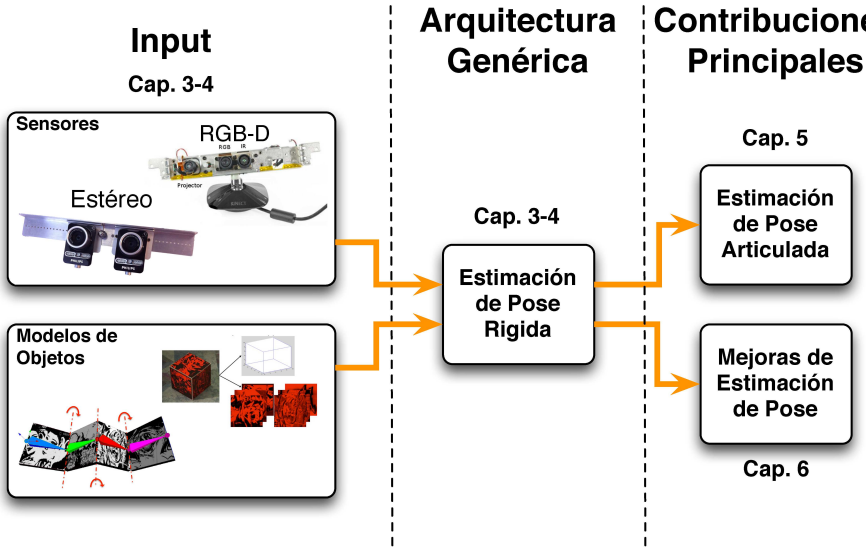


Figura 2.3: Estructura de la tesis. Este esquema muestra las diferentes partes que componen esta tesis y las relaciones entre capítulos.

técnica para la estimación de la pose de objetos basada en características de la visión artificial.

- Capítulo 4 - Estimación de pose de objetos rígidos: Después de la revisión de la literatura, una arquitectura genérica para el seguimiento de objeto se define en el Cap. 4. La arquitectura genérica tiene como objetivo una estimación de pose robusta y precisa con el objetivo de obtener resultados competitivos. Esta arquitectura define las bases de este trabajo y establece un entorno de desarrollo para futuras contribuciones. El esquema de estimación de posición descrito en este capítulo, define el tipo de datos de entrada, el modelo de objetos y la metodología de estimación de posición basando en el estudio del estado-de-la-técnica.

- Capítulo 5 - Estimación de Pose Articulada: Desde el entorno de desarrollo creado con la arquitectura genérica, el Cap. 5 describir las mejoras realizadas en el dominio del tipo de objeto empleados. La contribución extiende la estimación de pose de objetos rígidos a objetos articulados. En este capítulo, también se describen las modificaciones realizadas en la arquitectura general, tales como adaptaciones del modelo de objetos. También se presenta la nueva metodología para la detección del objeto articulado.

- Capítulo 6 - Mejoras en la Estimación de Pose: En el Cap. 6 se realiza un rediseño de la arquitectura. Modificaciones en cada etapa de la estimación de pose son descritas con el fin de mejorar el rendimiento de estos sistemas. Estas mejoras también involucran simplificaciones en el tipo de características empleadas y la incorporación de información del entorno en el proceso de estimación de pose. En este capítulo, la metodología de estimación de pose cambia de detección de objetos a seguimiento de objetos con el fin de mejorar la precisión de la estimación de pose.

Esta estructura es descrita en la Fig. 2.3, donde se destacan las relaciones entre capítulos. Después de las cuatro partes principales, la tesis concluye con un capítulo de conclusiones (Cap.8), donde se destacan las contribuciones principales y se proponen el trabajo futuro.

Chapter 3

State of the Art

Object pose estimation has been study for many authors in the last decades and it is an ongoing problem. Due to an extend literature, we aim to give an outline to ease the reading of this work. Further literature can be found that extends the algorithms and methods here addressed [4, 5].

This chapter reviews the different methods that have been developed for object pose estimation. According to the hierarchical structure of computer vision, we will describe the different techniques and algorithms related with this topic.

In this chapter we review the most used camera representations for standard cameras and depth cameras. We continue with a summary of the most used computer vision cues subtraction algorithms. And we end this section with the correlation between computer vision cues and camera representations using camera pose and feature tracking algorithms.

3.1 Camera Representations

In this dissertation we refer to two different types of cameras: Standard cameras and depth camera sensors.

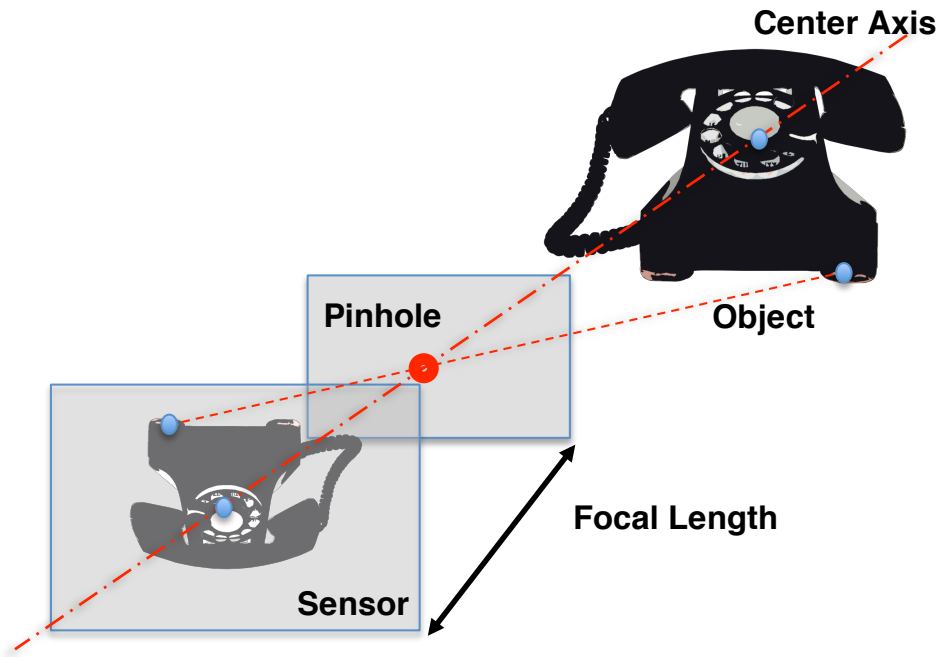


Figure 3.1: Pinhole camera model describes the mapping from a 3D scene to a 2D image captured by the sensor.

3.1.1 Standard Cameras

The standard camera representation is based on the pinhole camera model (Fig. 3.1). This scheme represents the vast majority of the cameras used in computer vision, in particular, in object tracking systems.

The pinhole camera model represents a simplified optic model that describes the mapping from a 3D scene to a 2D image. The pinhole aperture is assumed a point through which all projection lines must pass. The pinhole model represents the perspective projection model and defines the image formation as the projection from the 3D world to the image plane through the camera's pinhole. The reprojection is primarily affected by the focal

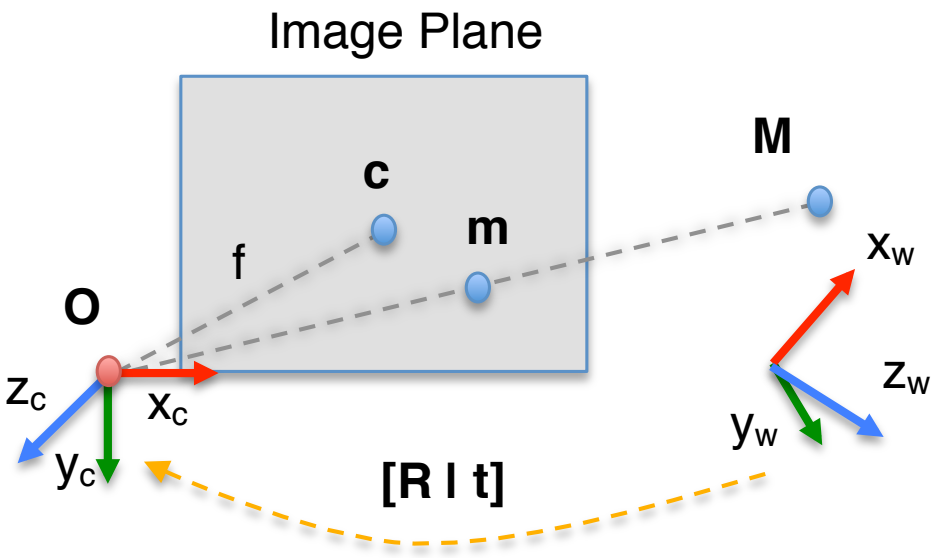


Figure 3.2: The camera coordinate system and the real world coordinate system define a correlation between real world points (3D) and image points (2D).

length. The distance between the pinhole and the sensor (focal length) defines the scene scaling and deformation onto the image plane.

Given a 3D point M with a 4-dimensional vector $(x, y, z, 1)$ as the Euclidean world coordinates system, the corresponding 2D point m with a 3-dimensional $(u, v, 1)$ image coordinated can be computed by:

$$sm = PM \tag{3.1}$$

Where s is a scale factor and P is a 3x4 projection matrix. The P matrix:

$$P = K[R|t], \tag{3.2}$$

is composed by the camera calibration matrix K and the transformation from the world coordinates to the camera coordinates (See Fig. 3.2). R represents the rotation matrix and t the translation vector.

In particular, the camera calibration matrix is defined by the following matrix:

$$K = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where α_u and α_v are proportional to the focal length f and defined by:

$$\alpha_u = m_u f \tag{3.4}$$

$$\alpha_v = m_v f, \tag{3.5}$$

where m_u and m_v represent the pixel dimension in u and v direction respectively. Therefore α_u and α_v define the focal length in pixel dimensions. The 2D point $c = \{u_0, v_0\}$ defines the principal point that represents the intersection of the optical axis with image plane (See Fig. 3.2). The parameter s refers to the skew parameter that is used if the image coordinates axes u and v are not orthogonal to each other. The estimation of the internal camera calibration parameters is done with calibration methods. See [6, 7, 8] for more information.

The presented projection model does not include the possible camera lens distortion. This deformation can be modeled as a 2D deformation of the image and corrected in an un-distortion process [9].

Knowing the camera calibration matrix is crucial for an accurate transformation between the image coordinate system and the real world coordinate system using the transformation $[R|t]$.

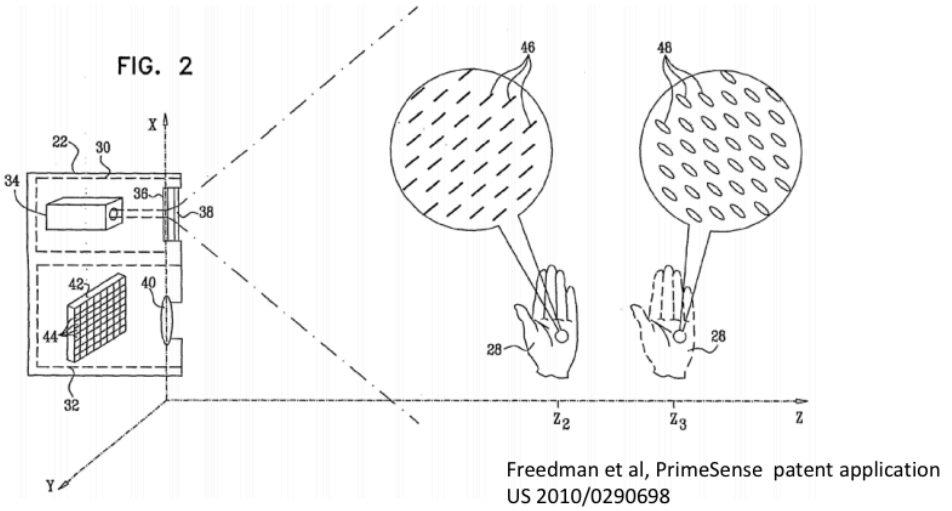


Figure 3.3: Freedman et al, PrimeSence patent [10]. First RGB-D camera patent. The image shows the projected pattern and its deformation according to the scene's depth.

3.1.2 RGB-D Cameras

The second camera representation refers to depth sensors. RGB-D sensors combine RGB color information with per-pixel depth information. The sensor projects an infrared pattern that is captured by an infrared camera in the sensor. Per-pixel depth is inferred from the deformation of the known pattern. The astigmatic lens causes a projected circle to become an ellipse whose orientation depends on depth [10] as shown in Fig. 3.3. An example of the infrared pattern is shown in Fig. 3.4.

The Fig. 3.4 also shows the elements that compose the depth camera. Beside the pattern projector and the infrared camera that capture the pattern, the device also features an RGB camera. The depth information is associated with the calibrated RGB camera that gives the per-pixel RGB information and it is based on the pinhole camera model explained in

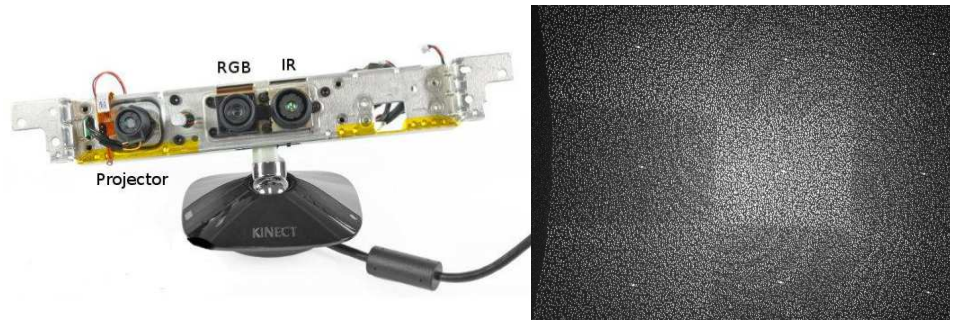


Figure 3.4: Microsoft Kinect - RGB-D Camera [11]. The left image shows the sensor with the different elements that compose the devices: Pattern projector, Infrared camera to capture the pattern and a standard RGB camera. The right image shows an infrared image of the projected pattern.

Sec. 3.1.1. The output of these cameras is defined as a point cloud that represents a collection of points in the three-dimensional space with an RGB associated attribute.

3.2 Computer Vision Features

Computer vision methods, and specially 3D tracking methods, are based on the available information that can be extracted from the images. The extracted image information is defined as image features that represent an abstraction of the scene data and which are used to simplify any computer vision task. Depending on their origin we distinguish between synthetic features and natural features.

3.2.1 Synthetic Features

Synthetic features refer to specific markers that are added to the environment and which are easy to locate (See Fig. 3.5). These markers allow an accurate and reliable feature detection.

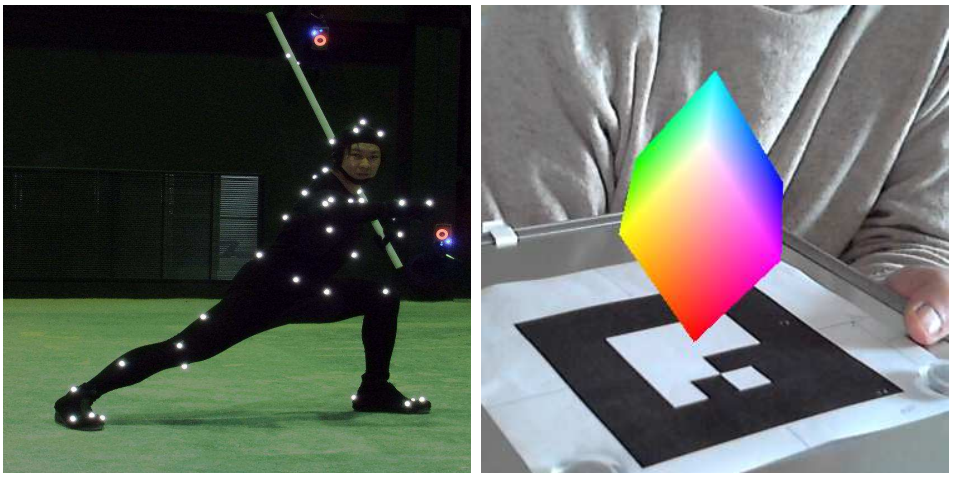


Figure 3.5: Artificial features. The left image shows a motion capture system (MoCap) that defines 2D coordinates on the image plane [12]. The right image shows a fiducial object used in ARToolKit that defines 3D coordinates [13].

There are two synthetic features classes. The first class is defined by *point markers* (Fig. 3.5). These features define a specific point on the image plane that implies two degrees of freedom (DOFs). A combination of multiple point markers shaped in a known structure can give an extra information that determines the 3D location of the features and therefore, defining six DOFs.

The second class is defined by *planar markers* (Fig. 3.5). *Planar markers*, also known as fiducials, are usually small figures that contain black and white squares and they carry more information that the *point markers*. They were designed for easy detection and they define a 3D position and orientation with 6 DOFs. Also, multiple *planar markers* can be combined with the purpose of increasing the information accuracy.

The main disadvantage of synthetic features is the requirement of adding extra information to the scene that makes the scenario unrealistic. In some

situations this option may not be feasible or even convenient such as outdoor scenarios.

3.2.2 Natural Features

This type of features overcomes the above problems by using natural features presented in the scene. Natural features do not interfere the scenario but are harder to detect that synthetic features and the information provided is not as accurate. There are many different classes of natural features and computer vision methods able to extract them from real world environment. We can classify these features in two main classes:

3.2.2.1 Sparse Visual Features

Sparse visual features represent meaningful parts of the image that define relevant information of the scene. This information refers to edges, corners, planes, objects, etc; that are identified based on their appearance on the image plane. Exist many different algorithms for *sparse visual features* extraction. Some of the main approaches are described below.

3.2.2.1.1 Edge-Based Methods The first approaches for natural features extraction were all edge-based methods because of their simplicity and their computational efficiency [14].

Edge features are pixels around which the image intensity information defines a severe variation. Edges are caused by a variety of factors such as surface normal discontinuity, depth discontinuity, surface color discontinuity or illumination discontinuity. One of the main advantages of edge features is the detection robustness in changing light conditions. Edges features are very useful in computer vision tasks such as scene decomposition and segmentation, camera calibration, motion analysis, etc.

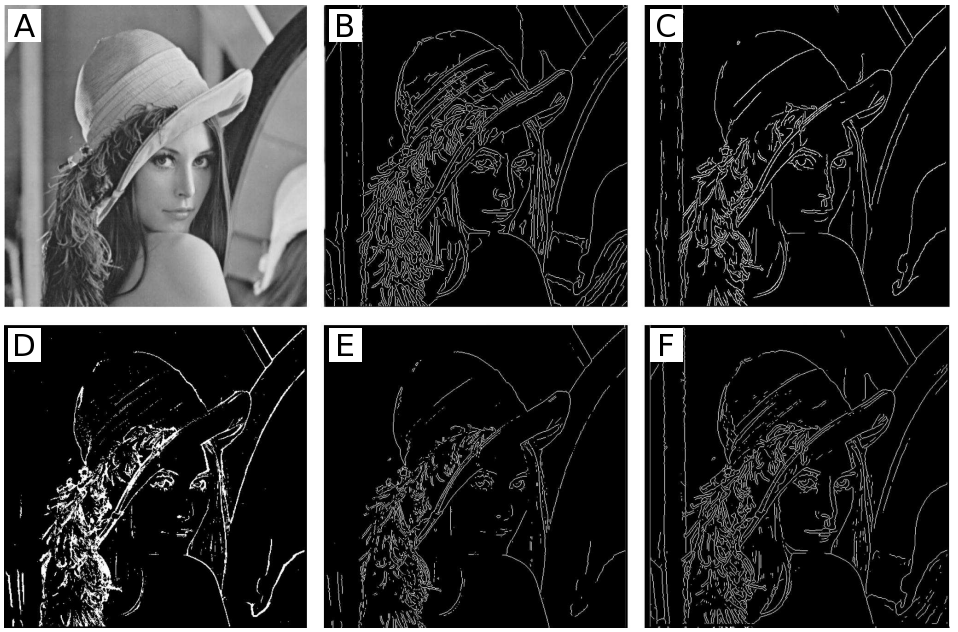


Figure 3.6: Edge extraction methods. A Original Lena Image, B Canny Edge Detector, C Edison Edge Detector, D SUSAN Edge Detector, E Sobel Edge Detector and F Rothwell Edge Detector.

The principal problem of edge-based methods is caused by noisy data that can provoke false positives or leads to information loss by blurring sharp intensity variations.

There are many different edge detector methods, Canny edge detector, Sobel edge detector, Edison edge detector, Rothwell edge detector, SUSAN edge detector, etc. Fig. 3.6 shows an example of different edge extraction methods. The image shows in this example is the well known *Image of Lena Söderberg* used in many image processing experiments. However, we review the two first algorithms that define the principles in which the other methods are based.

3.2.2.1.1.1 Canny Edge Detector This method uses a multi-stage algorithm to subtract edge information [15]. The algorithm is divided in five separate steps.

1. **Smoothing:** The first step applies a blurring filter that removes possible image noise.
2. **Gradient detection:** In this step, large gradient magnitudes are selected as potential edges.
3. **Edges sharpening:** Edges blurred in the initial step are converted to sharp edges.
4. **Edges classification:** A threshold is applied to classify edges into strong edges and weak edges.
5. **Edges connectivity:** Strong edges are interpreted as certain edges and weak edges are determined as certain edges if they are connected to strong edges. Certain edges are defined as the final edge image.

3.2.2.1.1.2 Sobel Edge Detector The Sobel filter is a process for edge detection that is divided in three steps. The two first steps apply 3x3 filter (kernels) that approximate the partial derivatives in x (horizontal) and y (vertical):

$$D_x = \begin{bmatrix} +1 & + & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I \text{ and } D_x = \begin{bmatrix} +1 & + & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I, \tag{3.6}$$

where I represents the image data matrix. The results of these two filters is $D_x(x, y)$ and $D_y(x, y)$. The final step is a gradient magnitude approximation calls the Sobel Filter:

$$S(x, y) = sqrt((D_x(x, y)^2) + (D_y(x, y)^2)). \tag{3.7}$$

The Sobel Filter is applied on the horizontal and vertical partial derivatives. The result of this filter defines the edge image.

In the literature is possible to find many variations to the algorithm such as Logarithmic Image Processing (LIP-Sobel) [16] or Parameterized LIP (PLIP-Sobel) [17].

Edge detection algorithms are usually extended as corner detection algorithms that define a corner as the intersection between two detected edges (See Fig. 1.2).

3.2.2.1.2 Interest Point-Based Methods Another types of sparse visual features are interest points. The methods that extract this kind of features are based on local appearance information. These methods rely on matching individual features between images and therefore they present a robust behavior in the presence of occlusion and clutter scenarios. The interest point-based methods also reduce computational complexity and it is possible to achieve robustness in light condition changes. These algorithms are based on patches matching that is a task divided in two phases: features extraction and matching.

Extraction:

The extraction phase handles the patches selection that can be done manually or automatically. The selected patches should have specific qualities such as textured patches that ease their identification. They should be different from their neighbors in order to avoid incorrect matches. And finally, patches that define repetitive patterns should have a lower weight in the algorithm process since they may lead to ambiguous features matching. The defined qualities assure a precise and reliable feature pose estimation [4].

Many different approaches have been proposed to describe the invariance of a selected pixel. Nowadays most of the methods rely on the auto-correlation matrix Z [18, 19, 20].

$$Z = \begin{pmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_u I_v & \sum I_v^2 \end{pmatrix}, \tag{3.8}$$

where Z's coefficients are the sum over a window of the derivative I_u and I_v of the intensities values with respect to the pixel coordinates [4]. The Z's coefficients are computed for every pixel in the image and it is used as a measurement for candidate's selection.

Matching:

The interest points selected in the first stage from one single image define a set of features m_i that can be matched to a different set of features m'_j extracted from a different image. Features in both sets are matched between each other and their similarities are measured using zero-normalized cross-correlation that is invariant to image intensity changes. This quality makes the procedure robust to illumination variations.

In the literature we can find a wide variety of interest point-based methods. The main algorithms are described below.

3.2.2.1.2.1 SIFT (Scale-Invariant Feature Transform) The SIFT features are based on multiple orientation histograms, which tolerate significant local deformations. SIFT is one of the most effective feature representations [21]. This method can robustly identify patches in challenging situations such as clutter environments and partial occlusions. The scale-invariant feature transform (SIFT) descriptors, which represent the image patches (keypoints), are invariant to uniform scaling, orientation, partially invariant to affine distortion and illumination change [22].

The significant invariance of the descriptors defined in SIFT is achieved by the following stages:

Keypoint location and scaling:

In the first stage of the algorithm, the location and scale of the keypoints are precisely determined by the Difference of Gaussian (DoG)

function applied in scale space. The DoG function is an efficient approximation of the Laplacian of Gaussian (LoG) and can be computed through an image pyramid that reduces the image dimensions in each level.

Keypoint stability:
The rotation invariance associated with each keypoint is achieved through the assignment of an orientation to the feature. The keypoint orientation provides stability under image rotations. The selected orientation is determined by the peak in the histogram of the gradient orientation within a region around the keypoint. The purpose of the describe orientation selection pursuits the stability of this algorithm against lighting condition changes.

Keypoint descriptor:
The keypoint descriptor is a feature's associated information that defines the 3D histogram of gradient locations and orientations of the keypoint. The point's neighborhood is divided into multiple subregions (typically 4x4) which contents are summarized by an 8 bins orientation histogram. Therefore, a vector with 128 elements (4x4 regions and 8 bins per region) defines the keypoint descriptor. The final vector is normalized in order to reduce possible light alterations.

3.2.2.1.2.2 SURF (Speeded-Up Robust Features) Based on SIFT method, SURF [23] was presented as speeded-up version of SIFT. The main differences between both methods are the region detector and descriptor generation. Firstly, SURF keypoints are described based on a Hessian matrix. In the case of descriptors, a vector of 64 elements is used to described the keypoints and its information is based on Haar-wavelet distribution. This type of descriptor is essential to provide a faster keypoint subtraction.

3.2.2.1.2.3 KLT (Kanade-Lucas-Tomasi) Points The KLT points represent an alternative for feature subtraction. Beside feature's subtrac-

tion from a single frame as in SIFT and SURF method, this algorithm track features across a sequence of images [24]. Keypoints are extracted from an initial frame and then the points are tracked in the following images.

Point tracking provides extra information by defining the feature position through the image sequence as a point trace. Acceleration, motion behavior, etc; can be specify by knowing the feature's trace. In the other hand, this method also implies disadvantages. KLT points are less precise than keypoints extracted with SIFT or SURF and features that disappear in the frame sequence may lead to incorrect features location estimation.

3.2.2.2 Dense Visual Features

In the previous section, the computer vision algorithms refer to sparse information that describes local information at a certain location on the image plane. Dense visual features are a different class of computer vision features that describe a specific attribute for every pixel in the image.

In the literature, there are many dense feature methods. The most addressed methods define a velocity field (optical flow) or a depth field (disparity). Here we give a brief overview of both methods.

3.2.2.2.1 Optical Flow Optical flow represents the apparent motion of the image caused by the relative motion between the observer and the scene [25]. The velocity of each pixel position is computed assuming constant pixel intensity:

$$m' = m + \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} dt, \tag{3.9}$$

where m is the image projection of a point at time t, m' represents the pixel location at time $t + dt$ and $(\dot{u}, \dot{u})^T$ the apparent speed of the 2D motion [4]. The velocity field produced by the optical flow algorithms has many different applications in several computer vision domains. One of its applications is the combination of optical flow with sparse visual

Figure 3.7: Stereo cameras set used for disparity computation. Typically, the separation between cameras is similar to the human eyes distance.

features. Edge detector can benefit from optical flow information avoiding error accumulation. KLT points computational time can be improve by including pixel velocity that leads the matching search.

3.2.2.2.2 Depth Perception The depth perception of the scene defines each pixel distance from the image plane to the real scene. There are two main types of depth perception: based on passive sensors and based on active sensors.

Passive sensors (Fig. 3.7) observe the scene in order to create the depth estimation. This type of sensors are used in disparity algorithm. One of the most common disparity algorithms that compute depth information are stereo vision systems [21]. These stereo vision systems are bio-inspired in human vision. Therefore, two cameras that are taking images from the same scene are set at a certain distance creating a system fully calibrated. Images taken from the different cameras at the exact same moment are compared. The aim of this comparison is to determine correspondences between the different images. With the different correspondences, the system performs a scene reconstruction that allows depth estimation.

In the other hand, active sensors performs a depth estimation based on sensing data broadcast that is measured to compute the depth information of the scene. There are many type of active sensor such as time-of-flight cameras, laser sensor, RGB-D cameras, etc. In particular, RGB-D cameras have become very popular in the recent years due to their performances, prices and dimensions (See Sec. 3.1.2).

3.3 Pose Estimation

The main core of this dissertation is focused on computer vision pose estimation tasks. The pose estimation aims to compute the state of an element in a real-world scenario. The scene's elements can represent objects, floor, doors, camera, etc; and their states typically define the element 3D location in the scene. Furthermore, the element's states can be extended to more complex states representation, such as internal object articulations, etc.

In computer vision, the hidden states of the elements are estimated base on the visual features extracted by the different algorithms. This task is usually called pose estimation that is responsible of transforming the features information into accurate and robust state estimations.

Adding extra information about the scene such as CAD models, planes or rough environmental models, ease the pose estimation task which aims more effective, robust and precise results.

In other to illustrate the connection between features and pose estimation, we review the main algorithms used in computer vision pose estimation.

Firstly, we have to distinguish between the two different types of pose estimation methods:

Pose Detection:
The first type of pose computation refers to the computer vision methods that do not based their estimation in previous stages. In other

words, no temporal information is used in the pose estimation process. This restriction affects the type of features that are used or the type of algorithm's input information. The main properties of these methods are a less accurate pose estimation but in the other hand, the algorithms are more robust in case of failure. Features such as edge (without optical flow improvement), cornet, SUFT, SIFT and depth information are suitable for pose detection algorithm.

Pose Tracking:
The second type of pose estimation is based on temporal information that can be extracted through the frame sequence. Including temporal information to the algorithms simplify dramatically the pose estimation task. Temporal knowledge, such as traces or accelerations, is used by these methods and requires a perfect tracking of the elements through the frame sequence since the algorithm estimations are computed base on previous state information. The advantage of tracking methods is that they have a very accurate pose estimation. By contrast, pose tracking drawbacks lie on the requirement of external initialization and resetting systems. These external systems start or restart the tracking system in case of non temporal information available. These problems appear at the begging of the sequence and in case of a misleading trace. Computer vision features such as optical flow and KLT points are computed using a sequence of images and therefore, temporal information. These methods can combine temporal information with non-temporal information since no temporal constrains are applied.

Based on this classification we distinguish between algorithms suitable for pose detection or pose tracking. These algorithms are well known mathematical tools that can be adapted into computer vision pose estimation task.

Here we are going to focus on some of literature methods that are addressed along this dissertation.

3.3.1 Direct Linear Transformation (DLT)

This algorithm was the first addressed method for estimating the 3D location of a scene's element (matrix P from equation (3.2)). This method estimates the object pose solving a linear system. Each feature defines a point in the 3D space M_i that has a 2D location on the image plane m_i. We assume that n correspondences between 3D points M_i and their projections m_i are given. A perspective projection matrix P relates these two sets of points projecting M_i points on m_i points (3.1).

The relation between the M_i and m_i defines two linearly independent equations

$$u_i = \frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \tag{3.10}$$

$$v_i = \frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}}. \tag{3.11}$$

For simplicity's sake, the equation can be rewritten as $Ap = 0$ where p defines the coefficients P_{ij} of the P matrix. The solution is computed using the Singular Value Decomposition (SVD) of A. This approach computes P matrix that contains the calibration matrix K and the transformation matrix $[R|t]$. Computing the entire P matrix will require a large number of correspondences and therefore, a higher computational cost. The problem can be simplified if the calibration matrix is given:

$$[R|t] \sim K^{-1}P. \tag{3.12}$$

This simplification of the problem aims more reliable results in pose estimation scenarios [4].

3.3.2 Perspective-*n*-Point (P*n*P) Problem

Fischler and Bolles [26] introduce the term Perspective-n-Points problem that solves the P matrix base on n-features points. The PnP is based on the

DLP algorithm but instead of solving the P matrix, strictly solves the R and t parameters independently. In order to compute the transformation pose, four is the minimum number of correspondence points required. With four points, the algorithm is able to find the unique solution. This algorithm has no restriction in terms of point distribution, whether is possible to compute the solution transformation with coplanar or non-coplanar points, as long as they are not distributed in a single line.

They have been proposed several approaches to the PnP problem [26, 27, 28]. Some of these solutions cope the problem with a linear system, such as P3P method, or approximating the estimation with an iterative approach, such as POSIT method.

3.3.2.1 Perspective-3-Point (P3P)

The P3P is the most basic case of the PnP problem. In this approach, four correspondences that define a 3D location in the object and a 2D location in the image coordinate system are required (See Fig. 3.8).

The four points are used to determine the distances between the four points and the camera origin O are computed as $\|OA\|$, $\|OB\|$, $\|OC\|$ and $\|OD\|$ and converted into the pose configurations.

Using the ABC triangle (Fig. 3.8), we can determine an equation for each triangle side (AB, AC, BC) based on the law of the cosine. For instance, the triangle determined between the side AB and the origin O with u and v projection of the points A and B on the image plane (see Fig. 3.9), it defines the next equation based on the law of the cosine:

$$AB^2 = OA^2 + OB^2 - 2 \cdot OA \cdot OB \cdot cos(\alpha_{u,v}) \tag{3.13}$$

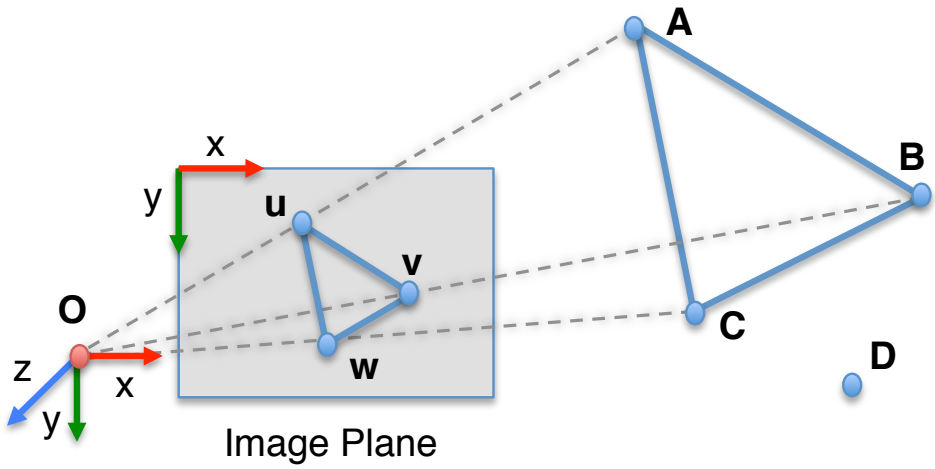


Figure 3.8: The four points ($ABCD$) required for the P3P algorithm are represented in the 3D space. The ABC triangle is projected on the image plane defining a projected triangle uvw. The camera origin is defined by the O point.

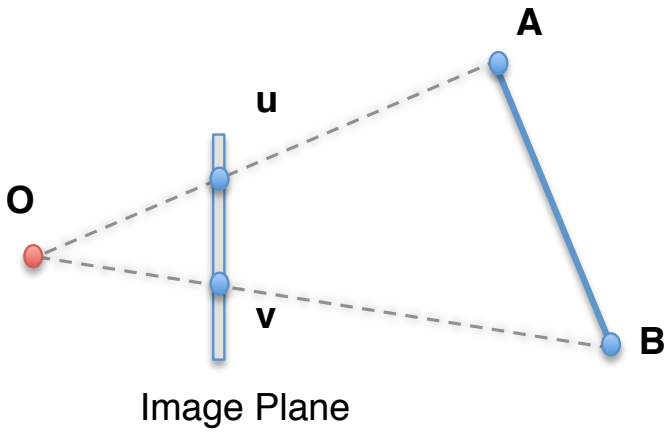


Figure 3.9: Triangle defined by the segment AB and the camera origin O. The A and B points are reprojected on the image plane and denoted as u and v.

The equations defined for each side give the following equation system:

$$\begin{cases} AB^2 = OA^2 + OB^2 - 2 \cdot OA \cdot OB \cdot cos(\alpha_{u,v}) \\ AC^2 = OA^2 + OC^2 - 2 \cdot OA \cdot OC \cdot cos(\alpha_{u,w}) \\ BC^2 = OB^2 + OC^2 - 2 \cdot OB \cdot OC \cdot cos(\alpha_{v,w}) \end{cases} \tag{3.14}$$

The P3P equation system can be simplified (see [29]) and solved to obtain the OA, OB and OC distances using the Wu Ritt's zero decomposition methods [30]. The 3D location of the points A, B and C can be obtained using the following equation:

$$M_i = m_i \left\| M_i O \right\| \tag{3.15}$$

Given the 3D points A, B and C, we want to determine the $[R|t]$ perspective transformation that aligned the A, B, C points to the u, v, w points. The solution to this problem was referred in [29] and represented by:

$$m_i = RM_i + t, \tag{3.16}$$

where the R and t transformation align both point sets. The optimal solution to determine R and t is achieved in three steps:

- Find the centroid of both point sets and translate the point set to the origin.
- Compute the optimal rotation (R) that aligns both point sets using SVD.
- Compute the translation (t) based on rotation (R).

The described algorithm uses 3 points to compute $[R|t]$ transformation. In order to improve the robustness of the method the method is extended with four points estimation (A, B, C and D) where the point set is group in four possible triangles (ABC, BCD, CDA, DAB). Each triangle output a rotation R and a translation t that are compared in order to select the best solution. For more details see [31].

3.3.2.2 Perspective from Orthography and Scaling with ITerations (POSIT)

The POSIT algorithm [32] is another approach for pose estimation. Based on correspondence points ($M_i \longleftrightarrow m_i$) where the minimum number of correspondences is four as in the previous approach. The first step of the algorithm computes an approximate solution of the rotation R and translation t by solving a linear system defined by the DLT algorithm (See Sec. 3.3.1). In a second phase, a weight is assigned to each feature based on the computed $[R|t]$. Each point coordinates are transformed based on their weight. Iteratively, a new estimation is computed from the weighted point coordinates until convergence. The drawback of this approach is high sensitivity to noise and the impossibility that the method has for only use coplanar points.

3.3.2.3 Robust Pose Estimation

In pose estimation, the appearance of erroneous points (outliers) due false positive in feature detection methods is a very common problem. Including outliers in pose estimation algorithms lead to severe pose estimation errors. Both types of algorithm performance can be highly improved with the combination of robust estimation mechanism.

One of the best-known algorithms for robust estimation is RANSAC (RANdom Sample Consensus). Introduced in [26], this iterative algorithm estimates the parameters of a mathematical model based on observed noisy data that contains *outliers* (erroneous points). The course of the method starts with a random selection of the minimum data subset. The size of the minimum data subset is determine by the minimum number of data elements with which it is possible to compute the model's parameters. The model's parameters are matched with the rest of the data elements. An error function determines if a data element suits the determined model's parameters (*inliers*) or otherwise (*outliers*).

If the number of *inliers* is below the defined threshold, the model's parameters are rejected and a new data subset is randomly selected. In case that the number of *inliers* is above the defined threshold, the model's parameters are considered valid and the outliers are removed from the data set. Finally, a new model's parameters estimation is computed but in this case only using the inlier elements. In the PnP estimation, RANSAC algorithm is used selecting the minimum number of correspondences that are used to compute a $[R|t]$ transformation (4 correspondences). For each point, the computed transformation is applied to the data points and the error between the transformed point projection and its expected location on the image plane is computed using:

$$d(m_i - R \cdot M_i + t), \tag{3.17}$$

where d represents the Euclidean distance between the transformed M_i point and the m_i point. The distance error is used to classified each point as an *inlier* or *outlier* based on a defined distance threshold.

3.3.3 Iterative Closest Point (ICP)

The ICP algorithm is a mathematical tool to minimize the differences between two datasets:

$$X = x_1, \ldots, x_n \tag{3.18}$$

$$Y = y_1, \ldots, y_k \tag{3.19}$$

This algorithm can be used to solve many different problems and input dataset can represent a wide variety of information. Particularly, in computer vision the input data commonly are represented by correspondences, that define a 2D image coordinates and a 3D world coordinates, or by multiple point clouds that are composed by a collection of 3D points.

The concept of the ICP algorithm is to iteratively estimate the rotation R and translation t that minimize the distance error between both datasets.

The algorithm is divided in the following steps:

- The elements of one of the dataset (X) are associated with the elements of the second dataset (Y). The association is based on the distance between points and it is computed using the nearest neighbor algorithm [33].
- Based on the associated elements, a transformation $[R|t]$ is estimated. The computation is carried out with a mean square cost function:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|x_i - Ry_i - t\|^2, \tag{3.20}$$

  where x_i and y_i are corresponding points. This transformation aligns both dataset minimizing the distance error between associated points.
- The positions of one of the dataset's elements are updated base on the estimated transformation.
- The estimated transformation error is computed with and error function and evaluated based on a stopping criterion. If stop conditions are not fulfilled, the algorithm returns to the first step. Otherwise, the final $[R|t]$ transformation is given as output.

One of the disadvantages of the ICP algorithm is related with the initial poses of the datasets with respect to each other. If the initial poses are very different, the element association could lead to an incorrect transformation that suffers from local minima.

3.3.4 Particle Filter (PF)

Particle filter algorithms are used to estimate the hidden state of a system. The particle filter method is divided in two different steps. In the first step,

the particle filter computes the system's hidden state evolution, which is represented by a state vector. The state vector x_k, represents the system state at any moment in time. The system's state evolution is defined by:

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3.21}$$

where f_k is a nonlinear function that defines the evolution of the state vector x_k based on the previous x_{k-1} estimated state and the estimated noise vector v_{k-1}.

The second step of the PF algorithm computes the x_k estimated state error z_k using the following function named validation function:

$$z_k = h_k(x_k, n_k) \tag{3.22}$$

where h_k is a nonlinear function and n_k is the measurement noise vector. The function (3.22) aims to estimate the error x_k based on the set of all available measurements $z_{1:k} = \{z_i, i = 1, \ldots, k\}$ up to k time.

The prediction of a state space model at time $k + 1$ can be obtained based on the current state x_k and the available observations $z_{1:k}$. System state estimation with particle filter is based on the recursive Bayesian approach, which recursively calculates an estimation of the state x_k at time k, taking different values and given the data $z_{1:k}$ up to time k. Many different approaches have been proposed that incorporate different evolution function or validation function [34, 35].

In object pose estimation, PF hidden state is represented by six dimensions vector that defines the object 3D location. Three parameters (x, y, z) define the translation and three parameters (pitch, roll, yaw) represent the rotation around axis X, Y and Z. The $[R|t]$ transformation matrix can be easily obtain from the estimated state vector. The evolution function and the validation function is defined according to the type of element that we are tracking and the type of features that we are using for the pose estimation.

3.3.5 Pose Estimation Summary

The presented mathematical tools have a direct application to the pose estimation problem. Some of these methods use temporal information and therefore, suitable for tracking system. Some other methods do not use temporal information, hence, they are used in pose detection systems. The Perspective-n-Point (PnP) problem is a pure detection methods, meanwhile ICP can be used as a detection method but can be improved if temporal information is added such as object pose trace or object acceleration information. In the other hand, PF is a pure tracking method since its estimation is base on previous states estimation.

Chapter 4

Rigid Object Pose Estimation

In chapter 3, we have introduced the state-of-the-art in computer vision object tracking and detection. In the current chapter, we focus on the implementation of the state-of-the-art. In particular, we define the diverse methodology for object modeling, feature extraction and pose estimation related with the rigid object pose detection.

Pose detection is a specific class of pose estimation methods that does not use temporal information in its estimation process (See Sec. 3.3).

Firstly we start with an overview of the state-of-the-art approaches in computer vision rigid pose estimation. After the literature summary, a generic architecture is defined for object pose estimation. The generic architecture states the basis of our work and defines the environment where experiments are carried out. Also in this chapter, we define the tools used in the system's implementation. In order to test the system's performance, we develop a benchmark that defines different challenging scenarios and allows a qualitative measurement. Finally, we combine the pose detection imple-

mentation with pose tracking methods that creates a robust and precise combined system.

4.1 State of the Art

Based on the computer vision's tools and features explained in chapter 3, in this section, we continue the review of the literature related with rigid object pose estimation.

Estimating the six DOFs (three translation and three rotation) pose of rigid objects is critical for robotic applications involving object grasping and manipulation, and also for activity interpretation. In many situations, models of the object of interest can be obtained off-line, or on-line in an exploratory stage. Since it is such an important ability of robotic systems, a wide variety of methods have been proposed in the past. We only provide a brief overview of the major classes of methods here. The most popular methods match expected to observed edges by projecting a 3D wireframe model in the image [36]. Many extensions have been proposed that exploit also texture information and particle filtering in order to reduce sensitivity to background clutter and noise [37, 38]. Detection approaches on the other hand can recover the pose without requiring an initial estimate by using sparse keypoints and descriptors with wide-baseline matching [4, 37]. A different class of approaches relies on level-set methods to maximize the discrimination between statistical foreground and background appearance models [39]. These methods can include additional cues such as optical flow and sparse keypoints, but at a large computational cost [40]. Most of the above-mentioned approaches can exploit multi-view information, but dense depth information is rarely used. Recently, due to the prevalence of cheap depth sensors, depth information is being applied with great success in various related problems, such as on-line scene modeling [41] and articulated body pose estimation [42]. In this implementation we focus on the detention methods since these methods are robust and do not require external method for initialization or recover from failures. We are inspired in the detection

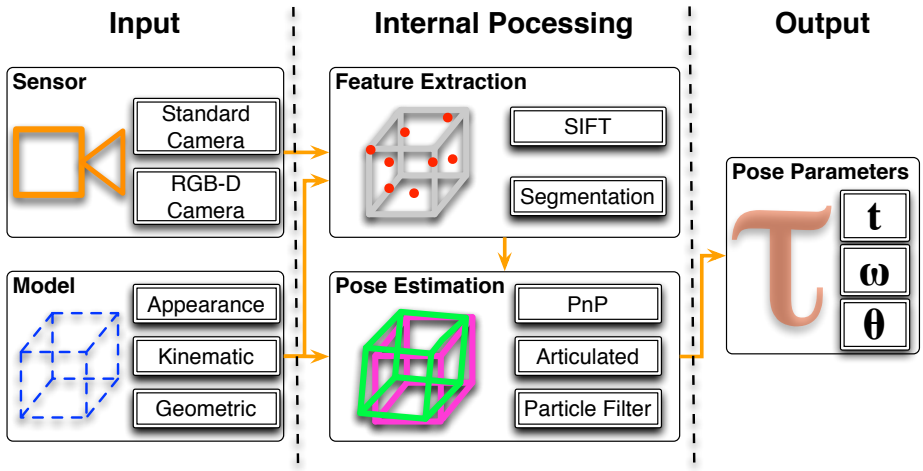


Figure 4.1: Generic dissertation architecture. It is divide in five main blocks. The blocks configuration defines the pose estimation system purpose.

approach presented in [37] that implement a referent method in computer pose detection system.

4.2 Generic Architecture

In this section, we define a generic architecture that describes the estimation system's structure that is used throughout this dissertation. This generic architecture allows modifications and improvements, advantages that are used in the following chapters.

Fig. 4.1 shows the generic architecture diagram. The structure is divided in five different blocks:

Input:

The state-of-the-art indicates multiple types of input data used by the

pose estimation systems. The commonly used input data is captured by standard camera sensors that yields monocular images of the scene. Along this dissertation we also address the depth information as input data that is obtained by stereo camera set up or RGB-D cameras.

Model:

Adding model information simplifies the pose estimation computation. These models can be generated in an off-line process or, in case of tracking systems, on-line in an exploration stage at the beginning of the tracking sequence. Beside the model generation process, there are a wide variety of models that can represent multiple types of elements such as objects, planes, scene information, etc. Among other, models can represent:

- Geometric models, which could be defined with edges, contour, geometry shape, etc.
- Appearance models, that codify the object texture information by feature descriptors. These descriptors are generated using interest point methods such as SIFT, SURF, KLT, etc.
- Kinematic models, represent the mechanic behavior describing the object's motion without references to the forces that cause the motion. This description brings extra information to the pose estimation systems. Elements such as articulated objects require a kinematic model (skeleton) that describes every object's independent part motion and constrains.

Features Extraction:

Features extraction is one of the basic blocks in the generic scheme, which defines the types of features used. Computer vision features state the feature extraction algorithm. The system's input has to be consistent with the type of used features. Throughout this dissertation we design different architecture configurations based on different feature classes, such as SIFT features or segmentation features.

Pose Estimation:

The main core of computer vision pose estimation lies on the pose estimation method. The type of method defines the system's character, outlining the system as a temporal approach (tracking system) or as a single-frame approach (detection system). Therefore, the sensor type, the object's model and the feature class, have to be consistent with the system's temporal orientation.

Output:

The final stage defines the output yields by the estimation system. The output describes the state of the object, which represent its 3D scene's location. In general, for rigid objects, the object state τ is described by its translation $t = (x, y, z)$ and by its orientation $\omega = (roll, pitch, yaw)$ that are Euler angles that represent X, Y and Z axis rotation. In some scenarios with more complex object, some extra parameters may be required. This is the case of articulated objects where their internal articulations have to be described in the object's state. We use θ as a vector of parameters that describes the articulated object internal configuration (See Sec. 5.3.1).

4.3 Pose Estimation Architecture for Rigid Objects

An adaptation of the generic architecture is done for rigid object detection. Based on the literature review in Sec. 4.1, we implements a rigid object pose detection that follows the scheme represented in Fig. 4.2.

Highlighted color boxes stress the modules configuration for this implementation. The scheme defines standard camera data as the input to the system. Appearance and geometric object's models are used; and pose detection methods specific for object detection are explained in the following sections.

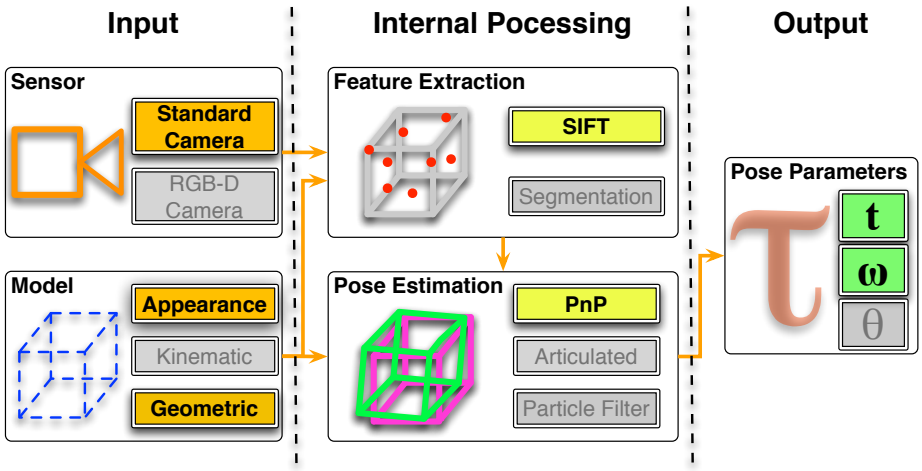


Figure 4.2: Rigid object architecture. The figure shows the generic architecture adapted for rigid object pose detection. The highlighted color boxes represent the configuration for each part of the scheme. The use of SIFT features and P*n*P pose estimation, define single-frame approach and therefore a detection system.

4.3.1 Object Models

Pose estimation algorithms required an object model that eases the pose estimation task. The object model generation is an initial process that can be generated in an off-line process or in an on-line process at the exploration stage.

In rigid object pose estimation, the object model can represent any type of object's information and every author defines and uses the object model in different ways. Most commonly used object models can be defined by its geometrical structure used in algorithms base on edge and contour detection or by its appearance information that is extracted from the object by appearance methods such as SIFT, SURT, KTL, etc.

For rigid object detection, the used models represent a combination of geometric information and appearance information. In the course of this dissertation, the object representation has evolved from simple geometric shape representation to complex shape representation.

4.3.1.1 Geometric Object Representation

The initial object representation was based on simple geometric objects that their shapes were easy to model through manual vertex coordinates definition. Each vertex defines a 3D coordinate in the model coordinate system that is related with the real object dimensions.

The defined geometry is combined with the appearance information of the object. Inspired in rendering engines, we define a single or multiple texture images that contain the appearance information. Geometric vertices have an associated texture image coordinate that creates a link between the geometry and the appearance. See Fig. 4.3.

The main disadvantage of this representation process lies on the manual definition of every vertex of the model (3D coordinates and associated texture coordinates). The manual vertices definition is not a problem in object with simple geometric shapes but extending the tracked object's set to more complex objects, the process become unbearable.

4.3.1.2 Generic Object Representation

Due to the previous representation limitations, a generic modeling process was proposed. This representation is based on commercial modeling tools that generate standard 3D computer graphic models.

4.3.1.2.1 Blender Blender is an open source tool used for 3D modeling [43]. This tool allows complex object modeling through a dynamic interface that also allows texture object definition. Fig. 4.4 shows an example of an object designed in blender and its associated texture image.

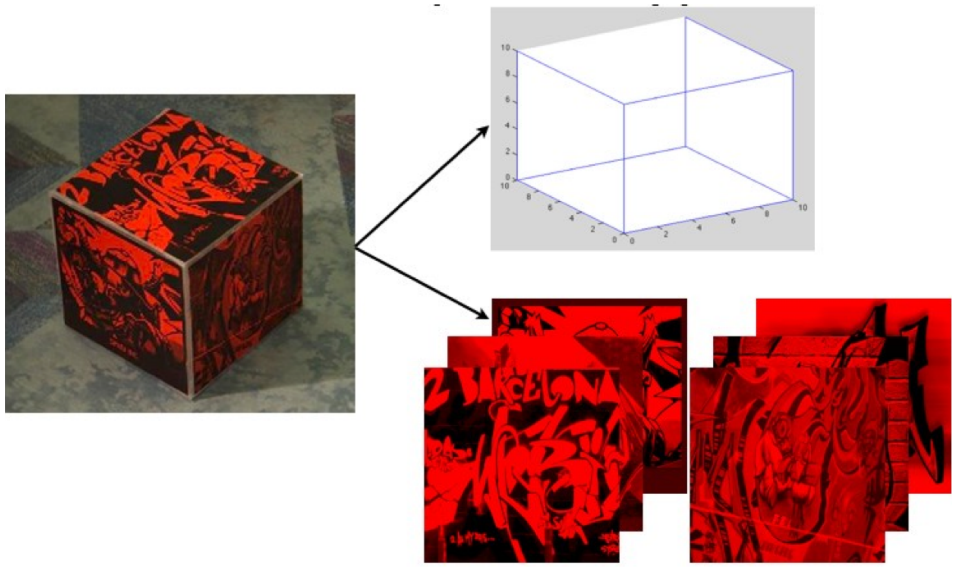


Figure 4.3: Example of a simple geometric object and its representations. Left image, picture of the real-world object. Right image, object's decomposition in geometric model and appearance model that is defined by six texture images.

Blender also provides mechanism to describe object's kinematic definition. For instance, articulated object behavior is defined with an associated *skeleton* (*armature* in Blender) that allows a description of the moving parts of an object (See Chap. 5). Generated object models contain the geometry, texture and *skeleton* information (if defined) that can be process by standard rendering engines. This tool allows the modeling of any kind of object since we do not have any geometry limitations as in the previous approach.

4.3.1.2.2 123D Modeling Modeling tools, such as Blender, eliminate the geometric restriction's barrier, which allows any type of random shape modeling. On the other side, this mechanism may result problematic modeling natural objects in certain situations, e.g. when the model has to be

Figure 4.4: Blender Object. Left image shows the designed 3D model with its texture. Right image, shows the associated texture image.

realistic and accurate. In order to overcome this restriction, we address 123D free software that is a standard tool for objects reconstruction [44]. The tool uses as input a loop of several pictures in small increments about the subject (between 20 and 30 pictures). Based on the object features and the environment features, a registration of the 3D shape and texture is created by the software (Fig. 4.5).

The reconstruction software is based on appearance features. Therefore, restrictions as non-shining objects or untextured objects are required in order to obtain an accurate and realistic model. These restrictions are not problematic for our system since the pose detection is based on appearance information and the same requirement must be applied for the detected object.

As in the previous approaches, the generated object model contains the geometry and texture information that can be process by standard rendering engines. In comparison with Blender models, no articulation can

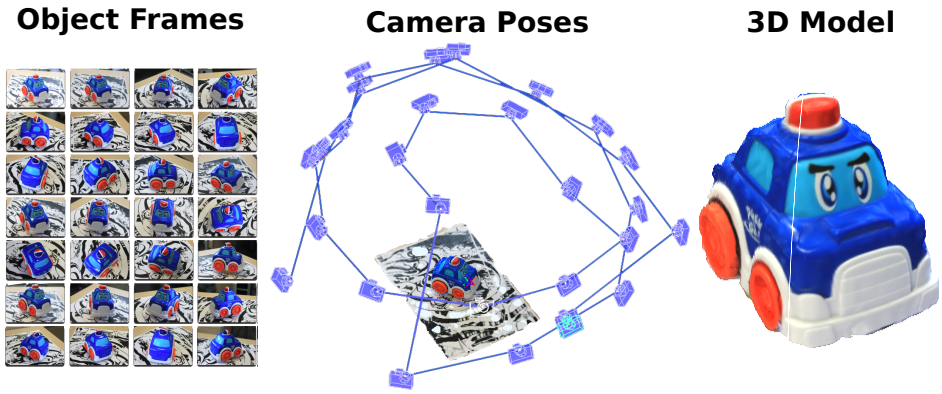


Figure 4.5: 123D Modeling Process. The left image shows the 28 pictures taken around the object. Center, shows the reconstruction of the camera poses for every picture. Right image shows the generated 3D model.

be defined with this tool but the model can be exported to Blender where the associated skeleton can be defined.

4.3.1.3 SIFT Feature Model

The implemented pose estimation approach uses computer vision SIFT features. We select SIFT features over the other appearance methods since they do not require temporal information as in KLT and the descriptor codification (based on 128 elements) are more robust and reliable that SURF descriptor (based on 64 elements) (See Sec. 3.2.2.1.2). In order to have a real-time performance, we use SiftGPU library that uses the GPU architecture to extract the SIFT features [45].

The SIFT methods rely on object appearance information that is defined by its texture. The object's texture is computed and the relevant keypoints are extracted and converted into feature descriptor. Depending on the type of model the appearance information is represented by a single/multiple texture images or by a 3D textured object.

4.3.1.3.1 Single/Multiple Texture Images In this case, a single or multiple flat image planes compose the object's texture. SIFT descriptors are extracted from the images and a 3D model coordinate is associated to each descriptor. Coordinates are interpolated base on the surrounding vertices texture location.

4.3.1.3.2 3D Texture Object In more complex object, the object associated texture image may present deformations with respect to the actual object's appearance. In this case, texture is only visible in its 3D form. For instance, object models generated with 123D modeling tool define texture images with deformations. Since the SIFT algorithm only works on images, we need a rendering engine that creates synthetic images of the objet from different views. The exact location of every image's pixel in the 3D world coordinates is known for every synthetic image. The 3D coordinates of every pixel is compute with the Z-buffer that contains the depth information for every pixel. We use the equation (4.1) to compute the 3D feature position.

$$P_w(X, Y, Z) = (\frac{xZ}{f}, \frac{yZ}{f}, Z) \tag{4.1}$$

Where $P_w(X, Y, Z)$ represents the feature 3D world coordinates, x and y are the feature 2D image coordinates, Z represents the pixel depth information and f the camera focal length.

Knowing the exact object orientation $[R|t]$ in every synthetic frame and the 3D feature coordinates, the features location can be reprojected into the model coordinates system with the following equation:

$$P_m = R^{-1}(P_w - t) \tag{4.2}$$

where P_w is the feature point in 3D world coordinates and P_m is the reprojection of the feature in the 3D model coordinates. The feature reprojection is done with each feature in every synthetic view of the object, resulting in a SIFT feature model where every descriptor has an associated 3D position in the model coordinate system. The described process is graphically explained in Fig. 4.6.

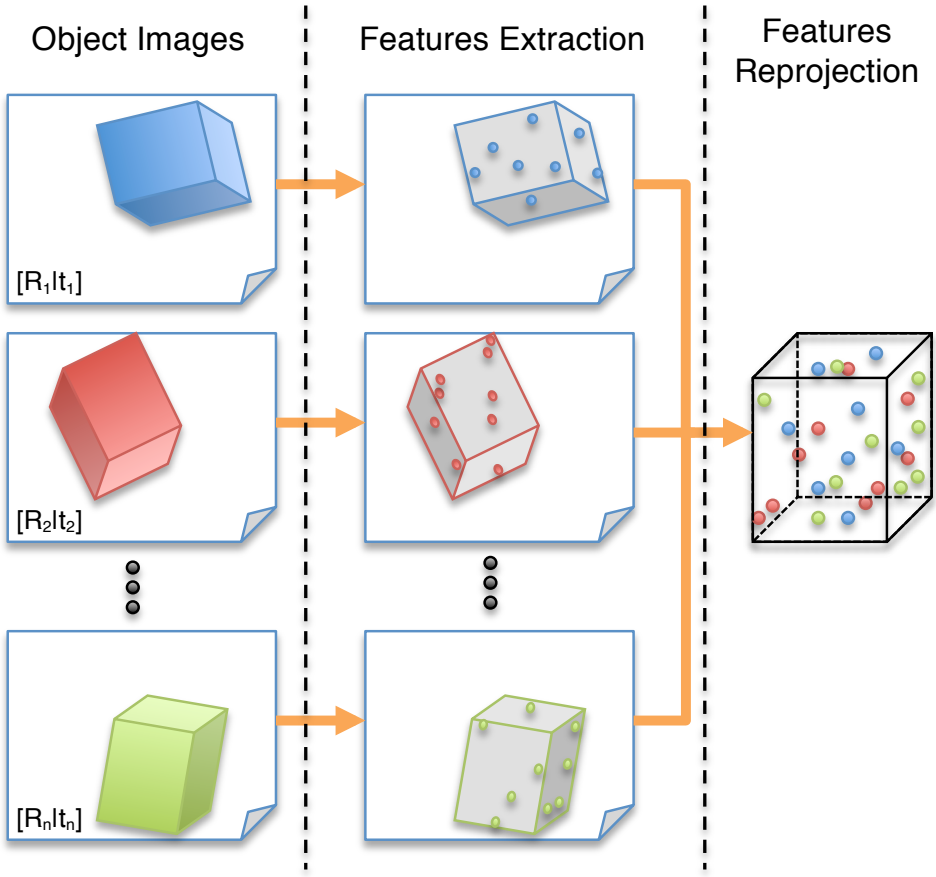


Figure 4.6: SIFT model generation process. Multiple synthetic views with an associated transformation ($[R|t]$) are defined. The SIFT features are extracted from every view. A final stage reprojects the extracted features into the model coordinate system, thus, generating the SIFT model.

For sake of reducing the number of synthetic frames, the rotation between successive frames is limited to 30 degrees. This ensures a lower number of model's features without losing information [37]. The SIFT feature computation are processed using the SiftGPU implementation [45].

4.3.2 Rigid Pose Detection

After the initial modeling phase described in the previous section, the object can be detected due to its appearance model. Regardless of the model generation process, the final object representation is defined with a SIFT feature model (Sec. 4.3.1.3) that associates descriptor with 3D model coordinates. Detecting the pose of a rigid object, according to the generic architecture, is a task divided in two main *internal processing* steps:

4.3.2.1 Feature Extraction and Matching

The image that represents the scene is processed by the SIFT method that extracts the scene's features. The scene's features and the SIFT model's features are compared in order to find the matches between the two feature sets. The scene's features extraction and the feature matching are carried out by the SiftGPU implementation [45].

Fig. 4.7 shows an example where the scene feature are matched to the SIFT model. For sake of a simplify demonstration, the figure shows the different object's texture sides and their matches association. Matches between both feature sets contain the associations between 2D image coordinates (image features) and 3D model coordinates (SIFT model features).

4.3.2.2 Pose Estimation Computation

The second step of the *internal processing* is the pose computation based on the detected feature matches. The 2D image location and the 3D model location of the matches represent the input for the PnP pose estimation

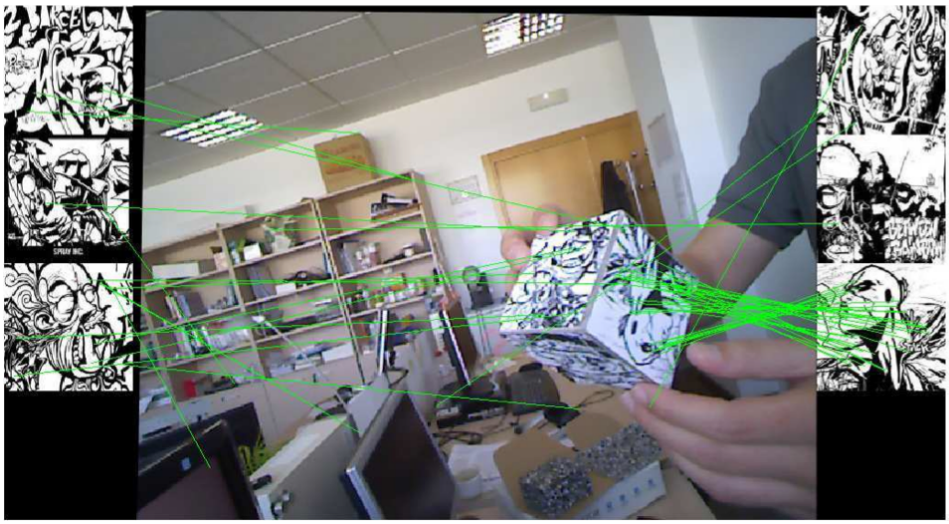


Figure 4.7: SIFT features matching. Both sets matches are defined by green lines that connect the scene's features to the model's features.

algorithm (See Sec. 3.3.2). In the example shown in Fig. 4.7, we can notice errors occurrence where is possible to detect some mismatched features. Therefore, with the aim of robust pose estimation, it is crucial to combine the PnP detection algorithm with RANSAC method (See Sec. 3.3.2.3). This combination creates a system robust to outliers. The PnP method combined with RANSAC is computed using the OpenCV library [46].

4.4 Detecting and Tracking Methods Combination

In the TOMSY project framework, the describe detection system was combined with a tracking system [47].

The tracking system, designed by Karl Pauwels [47], combines dense motion (optical flow) with stereo cues (depth). The visual cues are ex-

tracted from a stereo video stream and combined with the object's model information to estimate the 6 DOFs pose. In turn, model information is fed back to facilitate the cue extraction itself.

The tracking system use 3D models of arbitrary shape and appearance to track. Dense motion and stereo cues are obtained using modified coarse-to-fine GPU-accelerated phase-based algorithms.

The dense stereo cues are extracted with algorithms that perform a limited disparity range and have difficulties detecting fine structures. These problems are overcome by feeding the tracked object pose obtained in the previous frame back as a prior in the stereo algorithm.

The optical flow algorithm integrates the temporal phase gradient across different orientations and also uses a coarse-to-fine scheme to increase the dynamic range. This optical flow is improved through a synthetic optical flow (Augmented Reality (AR) flow) that includes feedback from the model reprojection.

The method incorporates the improved dense motion and stereo cues into a fast variant of the ICP algorithm to allow all the dense cues to simultaneously and robustly minimize the pose error.

The goal of merging this two pose estimation methods is to combine the benefits from the two different approaches (presented detection method and a tracking system). Detection methods have a robust performance in clutter scenarios, object occlusions and provide a great advantage by allowing traces reset in case of system failure. On the other hand, tracking methods provide an enhanced performance in pose estimation accuracy and noisy scene. The aim of this combination is to create a system that outperforms the state-of-the-art pose estimation systems.

The combined system carried out is shown in Fig. 4.8. Stereo video is received as input. Stereo cues, optical flow, augmented reality flow and keypoints (SIFT features for the detection system) are extracted from the scene. The model pose estimation is computed an fed to the system in order to improve the features extraction.

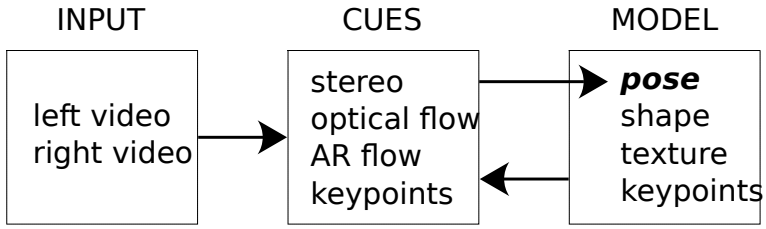


Figure 4.8: Combined system diagram. Stereo video input. The cues are combined and model information is fed back to facilitate the cue extraction.

The combined system computes the pose estimation from the detection and the tracking methods. Both poses are virtually reprojected into the scene and their pose errors are computed base on the same reprojection error used in the tracking system [47]. We select between poses based on the estimated error. The selected pose is assumed as correct and is fed to the tracking system for the next frame computation.

The tracking method implementation and both systems combination, was developed by Karl Pauwels (See [47] for further details).

4.4.1 Synthetic Benchmark

In the collaboration framework produced by TOMSY, a synthetic benchmark was also developed, which allows an accurate comparison with the state-of-the-art methods. Taking advantages of OpenGL rendering engine, camera distortions and background recreations were allows. We developed multiple sequences with different object with a defined object's trace that allows knowing the object's ground-truth for each frame.

Fig. 4.9 shows the different objects used in this benchmark and Fig. 4.10 shows the trace used for every sequence.

Figure 4.9: Rigid object models used.

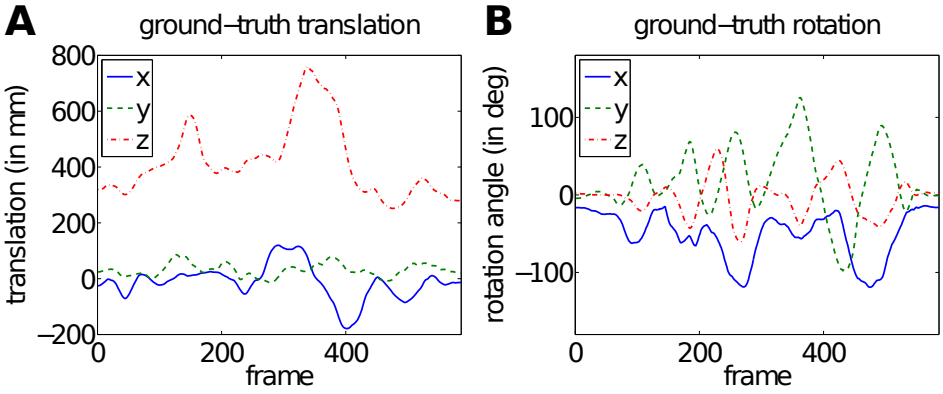


Figure 4.10: Ground-truth object trace in synthetic sequences.

4.4.1.1 Noise and Occluder

The synthetic scenario created with OpenGL allows the design of challenging scene, where different sequences are created corrupted either by noise or an occluding object. For the noisy sequences, Gaussian noise ($\sigma = 0.1$ intensity) is added separately to each color channel, frame, and stereo image (Fig. 4.11B). To obtain realistic occlusion (with meaningful motion and stereo cues), we added a randomly bouncing 3D teddy bear object to the

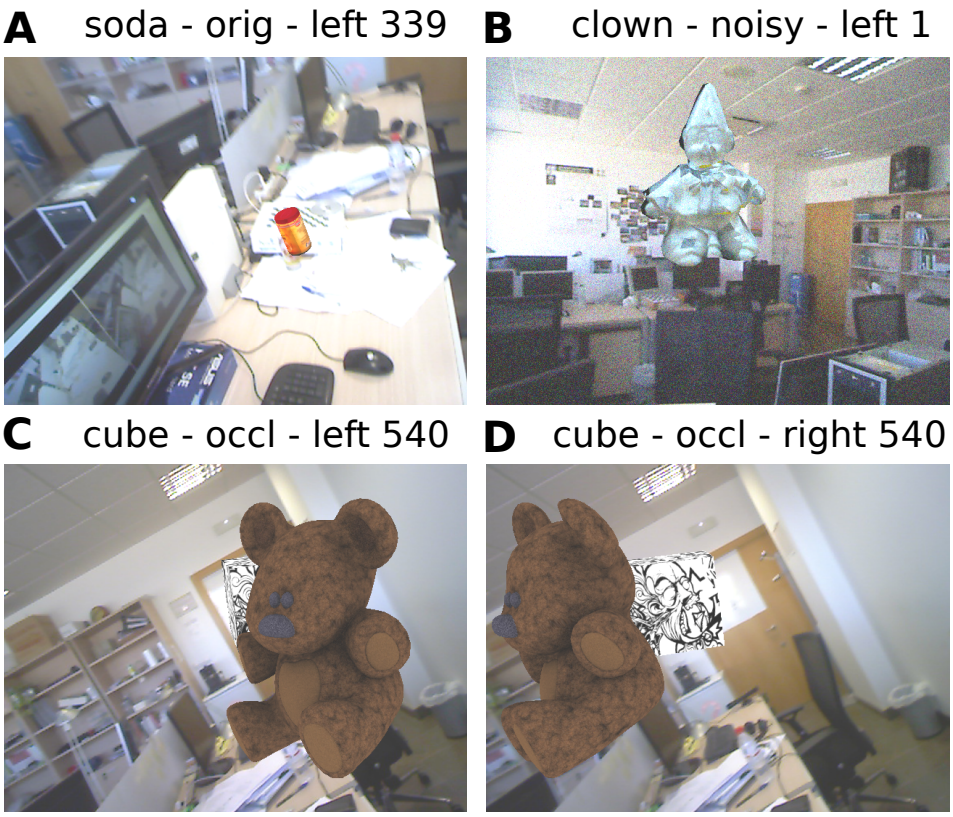


Figure 4.11: Benchmark example frames. A shows an original frame (noise free) where the tracked object (soda) is located relatively far from the camera. B shows a frame from a noisy sequence. C-D show occlusion situation from two different camera poses.

sequence (Fig. 4.11C,D). The occlusion proportion of the cube object over the sequence is shown in Fig. 4.12 for the left and right camera pose.

In this benchmark, every object is used in three specific sequences: original, noisy ($\sigma = 0.1$ intensity) and occluded sequence. Each sequence is generated from two camera positions (Fig. 4.11C,D) that simulates stereo

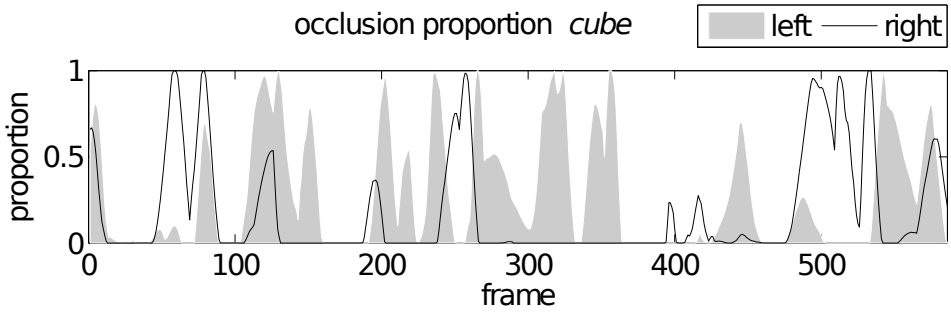


Figure 4.12: Proportion occlusion in the cube sequence.

vision, which is required by the tracking system. In total, the six objects benchmark is composed by 18 sequences.

The developed benchmark has been released and it is now available at *http://www.leonardorubio.com/*.

4.4.2 Error Evaluation

The synthetic benchmark allows pose estimation comparison with respect to the ground-truth. To compute the error, we use the estimated $[R|t]$ to reproject the geometric model into the 3D scene and compare its location with the ground-truth geometric model reprojection defined by $[R_{gt}|t_{gt}]$. Both reprojections define 3D locations for the vertices of each geometric model. The larges distance between corresponding vertices v_j is used as the estimated pose error e_p and computed with the following equation:

$$e_p = \max_j \|(Rv_j + t) - (R_{gt}v_j + t_{gt})\|, \tag{4.3}$$

The error measurement presented in (4.3) is a novel error evaluation introduced in [47].

4.4.3 System Comparison

Using the synthetic benchmark and the error function defined above, we compare the combined pose estimation system with two state-of-the-art methods: BLORT [37] and PWP3D [39].

For the comparison, we use the 18 benchmark's sequences. Defining an error threshold of 10 mm that classify the pose estimation, we compute the percentage of correct estimated frames for each sequence. The computed percentage for the 18 sequences and different methods are shown in Table.4.1.

Table 4.1: Tracking success rate(%) - orig = noise free; occl = occluded.

	soda			soup			clown			candy			cube			edge		
	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl
Combined mthd.	99	97	**68**	98	**99**	**80**	**100**	98	77	**100**	**100**	**81**	**100**	**100**	76	**98**	98	57
Tracking mthd.	**100**	**98**	67	**100**	99	74	100	**100**	70	100	100	75	100	100	71	98	**99**	57
Detection mthd.	61	37	44	93	74	77	92	71	74	96	91	80	98	96	**79**	0	0	0
BLORT 10k particles	76	65	54	77	66	63	88	82	76	77	76	64	93	94	76	72	91	**68**
PWP3D	84	84	44	96	96	44	96	89	44	84	84	39	84	74	38	63	63	50
BLORT 200 particles	58	60	45	47	54	40	56	62	48	46	49	41	53	54	39	63	63	50

In this table, the detection system presents a robust behavior in sequences with textured objects and dramatically fails when there is no texture at all, such as the edge object. In occlusion sequences, all the methods reduce their performance in comparison with corresponding original sequence. In the detection system, the percentage reduction is less affected that the alternative methods. This occurs due to the SIFT features used by the detection method, which are not affected by partial occlusions. On the other side, the detection system has a disadvantage when the sequence object is relatively small and it is located at a considerable distance from the camera. The region occupied by object in the frame is drastically reduced and so the number of correspondences between the object model and the scene's frame. A reduced number of correspondences is translated into an unstable PnP estimation.

This is the case of the *soda* sequences (see Fig. 4.11A) where the object's dimension is smaller in comparison with the rest of the object and the detection system has an inferior performance.

As shown in the table, the combined method confirms the perfect synergy of both modules. The combined system retains the excellent performance of the tracking method with greatly improved robustness to occlusions due to the detection system estimation. The combined system outperforms the state-of-the-art methods or even the tracking method and the detection method individually.

4.5 Real-world Scenario

The synthetic scenario is a controlled environment that allows us to test the pose estimation results. In the other hand, the disadvantage of these sequences is the recreation of a non-entirely realistic environment that does not represent a real scenario's problems where the pose estimation becomes a real challenge. Problems such as manipulator occlusion, illumination changes, realistic noise, etc; are issues that cannot be easily represented in a synthetic pictures. Therefore, the implemented method was tested in different scenarios with different objects. Fig. 4.13 shows an early pose estimation approach where the object model was generated manually and pose estimation was performed by the detection system without combination. In the Fig. 4.13, we can appreciate its accurate estimation although it shows some difficulties with large occlusions and when the object is relatively far from the camera.

Along enhancements were applied, the system's performance was improved. Enhancements such as complex objects with complicated texture were used, multiple objects tracking or the combined system; lead to remarkable pose estimation results. Fig. 4.14 shows an example of multiple objects detection.

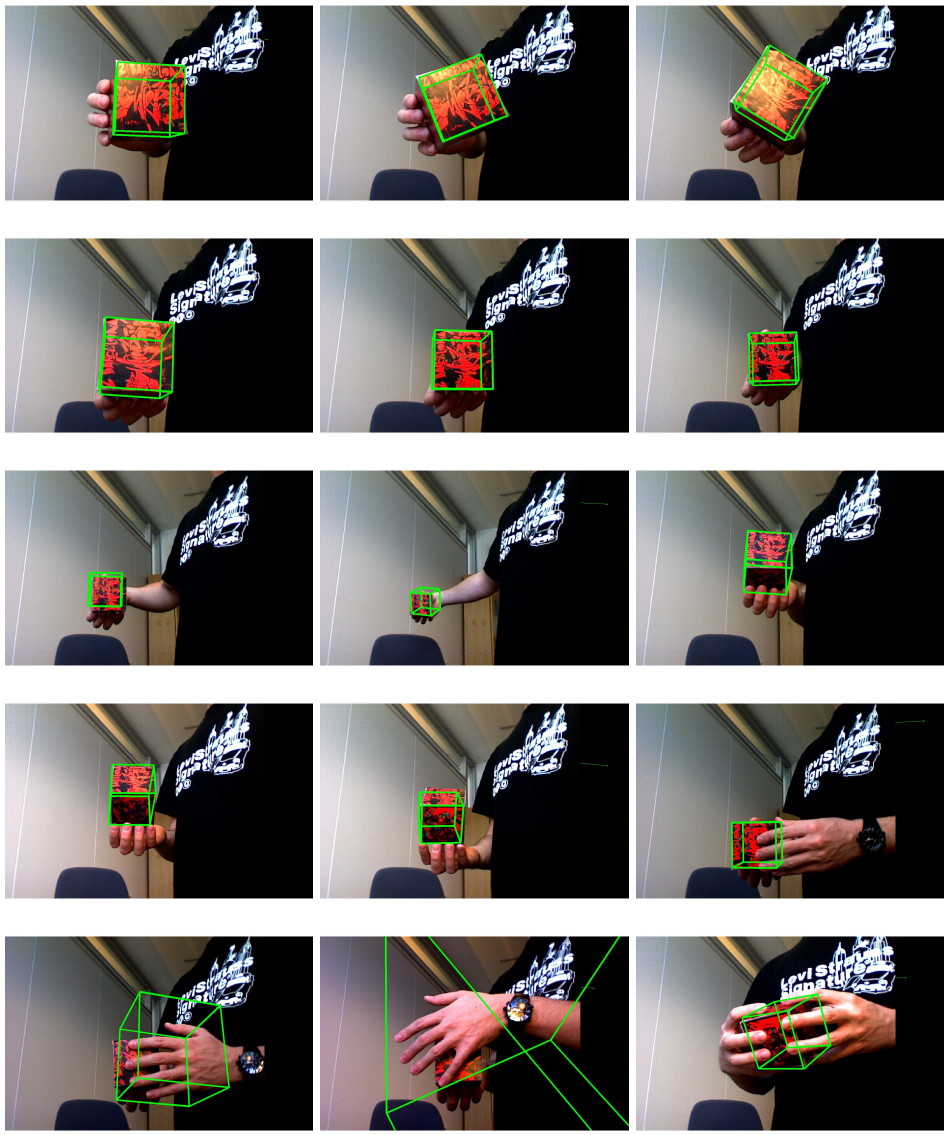


Figure 4.13: Early Pose Estimation Approach in Real-world Scenarios. The object's model was generated through a manual process.

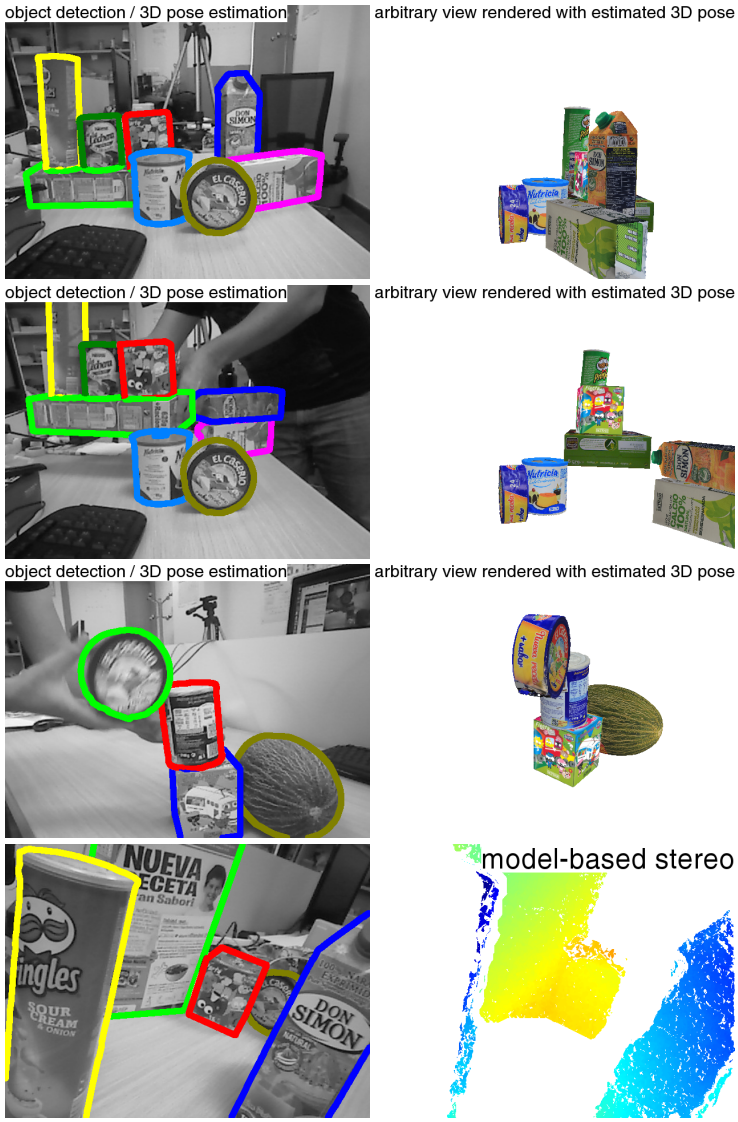


Figure 4.14: Multiple objects in real-world scenarios. The 3 first rows, the left side shows different real-world objects and their estimated poses. On the right, an artificial scene was rendered based on the estimated pose location. The forth row also shows real-world objects and their estimated poses. On the right side, a depth image is generated based on the object's locations.

4.6 Conclusions

We are focused on rigid pose detection since it is crucial for object pose estimation and it can be easily improved by combining the method with tracking methods as shown in this chapter. The detection method is based on appearance information provided by the SIFT algorithm. SIFT features are robust to illumination changes, occlusions, clutter scenarios, orientation, etc. These features are computationally efficiency and do not constraint the geometric shape of the detected object as in contour or edges based methods. The only restriction of using SIFT features is the requirement of textured on the tracked objects. This restriction is not a major problem since common daily objects, natural object, etc; are textured objects.

In order to estimate the object pose base on appearance features, we use the PnP approach since it is widely used in the state-of-the-art with outstanding results. We combine the PnP algorithm with a RANSAC implementation for performance improvement.

In the TOMSY project framework and thanks to the collaboration with Karl Pauwels, a novel method for rigid object pose estimation has been developed. The novel method perfectly combines detection and tracking algorithms. The results outperform literature methods and yield a remarkable performance in real-world scenarios.

This chapter has a great importance in the development process of this thesis. The defined generic architecture states the structure for pose estimation system and creates a developing environment whereby contributions are developed. According to Fig. 1.3 diagram, chapters 5 and 6 expose the main contribution lines based on the generic architecture.

Chapter 5

Articulated Object Pose Estimation

The implementation of a rigid pose estimation system (chapter 4) defines a starting point for improvements. According to Fig. 1.3, in this chapter,we present a novel contribution in the field of object detection, introducing a novel system for articulated object detection.

The detection and tracking of real-world objects based on visual information is an important task in several domains. In robotics, an understanding of the surrounding elements and their distribution in the environment is required in many applications such as grasping, manipulation, activity interpretation, etc. This research field has been very active in the last decades as far as rigid objects are concerned, and as a result impressive results both performance- and precision-wise have been reported on rigid pose estimation [47, 48].

Articulated objects are composed of multiple rigid parts connected by joints that allow rotational or translational motion, or a combination of both. Each joint thus introduces one or more degrees of freedom (DOF) in addition to the six rigid DOFs that determine the object's location and ori-

entation. Articulated pose estimation can be considered a natural extension of rigid pose estimation, and a wide variety of methods have been proposed for this problem as well. In the next section we present an overview of the major method classes. As a result of the large number of degrees of freedom, most articulated pose estimation methods are *tracking* methods that rely on temporal information and can only handle small frame-to-frame motion. In addition, once tracking is lost (as a result of occlusions or fast motion) these systems can only recover with the help of an external method that detects the pose. We present such a novel articulated pose *detection* method in this chapter.

5.1 State of the Art

Rigid object pose detection methods estimate the six DOFs (three for translation and three for rotation) that define the state of a rigid object. The state of articulated objects can be decomposed in terms of the six rigid DOFs that define the object's location and orientation and the DOFs that describe each internal joint. Consider for example a robot arm composed of different rigid parts connected by joints in a hierarchical manner. The base of the robot arm constitutes the root of the hierarchy, and each joint constrains the physical configuration of the articulated object.

Articulated pose estimation is a notoriously difficult problem. The main problem is the high-dimensionality that results from the large number of DOFs and the corresponding parameters that need to be estimated. All these DOFs can support complex configurations of articulated objects, the projection of which can result in a large variety of shapes with many self-occlusions. In addition, if we also consider uncontrolled environments, allow for rapid object motion, and require high performance, it is clear that articulated pose estimation is a very challenging problem [49]. A large number of approaches have been proposed in the past for articulated object pose estimation, as will be described in this section.

A first distinction can be made between *global* and *partial* pose estimation methods. Global methods estimate all the object parameters while ensuring that all the constraints are satisfied. Partial methods, on the other hand, do not enforce the constraints [50]. Instead they approach the problem by considering every rigid part of the articulated object as independent, each with its own six DOFs. These methods do not always aim at estimating all the object parts but instead often focus only on the most representative parts of the articulated object, such as the end effector of a robotic arm.

Another distinction can be made between methods that use temporal information and those that don't. The first group is referred to as *tracking* methods. They refine an initial articulated pose estimate (usually obtained on the previous frame) under the assumption that the object undergoes a small motion. Some of these methods are based on low-level motion features such as provided by the Kanade-Lucas-Tomasi feature tracker [51, 52, 53], edges [54, 55, 56], silhouettes [57], color blobs [58], etc. If the motion is sufficiently small, these systems can provide highly accurate estimates even for a large number of DOFs. They are however unable to recover from failures and need to be initialized by an external method or by hand. In an uncontrolled articulated object pose detection scenario, such an initial object pose may be difficult to obtain and tracking loss will occur frequently due to occlusions, mismatches or due to the object leaving the camera view. For this reason, a second group of methods exists that does not rely on temporal information. These articulated pose *detection* methods operate on a single image frame and are typically less accurate than tracking methods. This is however compensated by their capacity to recover from failures, a critical ability of robust systems.

Some of these articulated pose detection methods divide the problem in two stages [59, 60]. The first stage estimates the six DOFs of the base of the object (considering it a rigid object) and the second stage estimates the remaining internal joint hierarchy parameters. The largest and most visible part is usually chosen as the base of the model. For instance, in the case of an articulated robot arm, the actual base of the robot could be considered

as the base of the model. The object's edges are used in [59] to estimate the joint parameters and refined using an iterative Newton-Raphson approach. An exploration approach is used in [60] where all the possible poses are reprojected on the scene and weighted according to how well the boundary of the object matches the projection of the 3D shape. Both methods detect the pose parameters iteratively through the hierarchical structure.

Although the vast majority of methods only use visual cues from a single camera, there are some approaches that combine multiple cameras to reduce the number of occlusions or to provide stereo information to the detection methods. A probabilistic pose estimation is presented in [61, 62] that estimates the pose using the 3D position of special markers that are located on the object and registered by an (eight camera) motion capture system [12]. In recent years, with the introduction of commercial RGB-D cameras, depth information has been considered in many computer vision problems. Such compact RGB-D cameras provide precise depth information in real-time [11] and can be introduced in many scenarios in a non-intrusive manner. For example, in [42] the 3D point cloud provided by an RGB-D camera is matched to a database of articulated objects (generated in a training stage). This database only allows detecting a specific type of object though (human poses) and limits the joint motion in order to reduce the solution space.

Although many contributions have been made on this topic, the problem is still far from a stable and accurate solution. Therefore novel solutions such as the presented work are required to support the large number of computer vision applications.

5.2 Pose Estimation Architecture for Articulated Objects

Based on the remarkable system described in Chap. 4 for rigid object pose estimation, we present a novel approach for (single-frame) articulated object pose *detection* that is invariant to the motion and shape of the object,

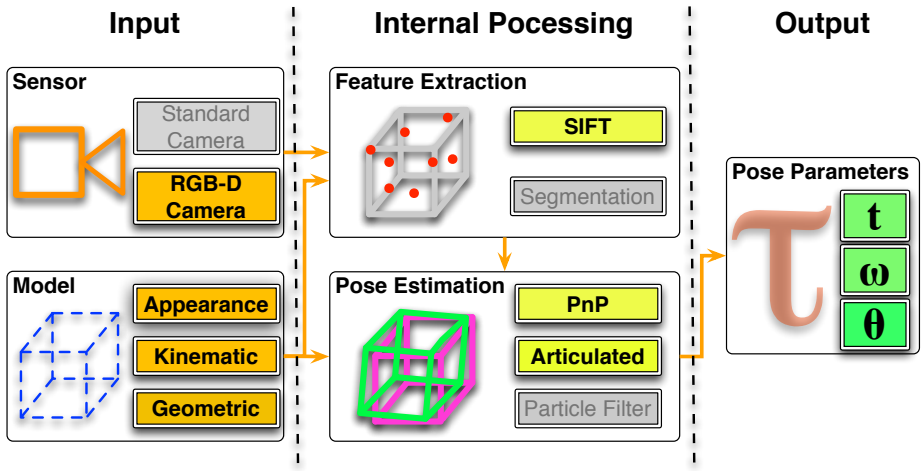


Figure 5.1: Articulated architecture configuration.

and that does not require temporal information. The method performs highly accurate *global pose estimation* resulting in a precise 3D location with millimeter accuracy in real-time. The proposed method is not limited to a particular type of articulated object, such as hands or bodies, but can instead handle arbitrary objects.

Using the defined generic architecture, we adapt its modules for articulated object detection (Fig. 5.1).

The method is model-based in that it relies on a 3D model of the articulated object. Alike rigid object models, we use the same modeling process defined in Sec. 4.3.1 to build an object model that defines the geometry and appearance of the object. In case of articulated objects, it is required to describe the kinematic information that defines the motion behavior of the articulated object. Kinematic information is described in details in Sec. 5.3. This does not limit the types of object, but it helps predicting poses in situations where the object is occluded [55].

Sensor information data used in this system is provided by an RGB-D camera [11]. The RGB-D data is used to compute the sparse appearance and depth information used by the proposed system.

As mentioned, feature extraction module used sparse appearance features. These features are extracted using the SIFT algorithm. As in rigid object SIFT features, they are computed using the SiftGPU implementation but in this case, the features information is combined with depth information extracted from the RGB-D camera.

The pose estimation stage, introduce a novel algorithm that computed articulated object pose using the extracted features and depth. The method combines a PnP pose estimation method with a novel algorithm that computes the articulated parts parameters. The pose estimation method is described in detail in Sec. 5.3.

We define the state of an articulated object as $\tau = (t, \omega, \theta)$, where t and ω define the translation and orientation and θ describe the state of the internal configuration of the object's kinematic chain (see Sec. 5.3).

In the presented work, unlike most methods, whose performance decreases as the number of joints increases, the complexity of the problem is not affected by the number of joints in our method. This is due to the novel way in which we decouple the rigid and non-rigid components of the pose in our estimation. Unlike partial pose estimation techniques, the proposed decoupling allows using the entire object (not just the base) to determine the rigid pose components. Using an extensive comparative evaluation, we show how the proposed method outperforms state-of-the-art alternative methods. Finally, we demonstrate precise pose detection in challenging real-world scenarios.

5.3 Proposed Method

The proposed method extends methods for rigid object pose detection, described in chapter. 4, to articulated objects. The method operates on

color and depth images obtained from an RGB-D camera and requires a 3D articulated model of the object under consideration. An overview of the proposed method is shown in Fig. 6.2. The articulated model and the four stages of the algorithm are discussed in more detail in the next sections.

5.3.1 Articulated Model

The model describes the object's appearance, geometry, and the different types of joints organized as a kinematic chain. An example model for a simple chain-like object is shown in Fig. 5.3. We consider an articulated object to be composed of n moving links and one base link. Both *revolute* and *prismatic* joints (each with one DOF) are supported between links. More complicated joints (cylinder, sphere, helix, plane) can be expressed as a composition of revolute and prismatic joints with zero link length. An example of the different joint types are shown in Fig. 5.4.

The articulated object thus consists of n joints with n DOFs. We define the internal configuration of the object's kinematic chain by $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\} \ , \tag{5.1}$$

where θ_i represents the parameter value for the i^{th}-joint. We use the same notation for *revolute* and *prismatic* joints. The object's base link on the other hand is unconstrained and can thus have an arbitrary location and orientation in the world. Its parameters consist of a translation vector $\boldsymbol{t} = [t_x, t_y, t_z]$ and a rotation axis vector $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$. Together they define the complete pose of an articulated object:

$$\boldsymbol{\tau} = \{\boldsymbol{t}, \boldsymbol{\omega}, \boldsymbol{\theta}\} \ . \tag{5.2}$$

The 3D articulated models are created using the open source tool Blender [43]. It allows the creation of objects with arbitrary geometric shape and texture. The vertices of each link are associated with its joint. The allowable motion of each link is constrained by the joints.

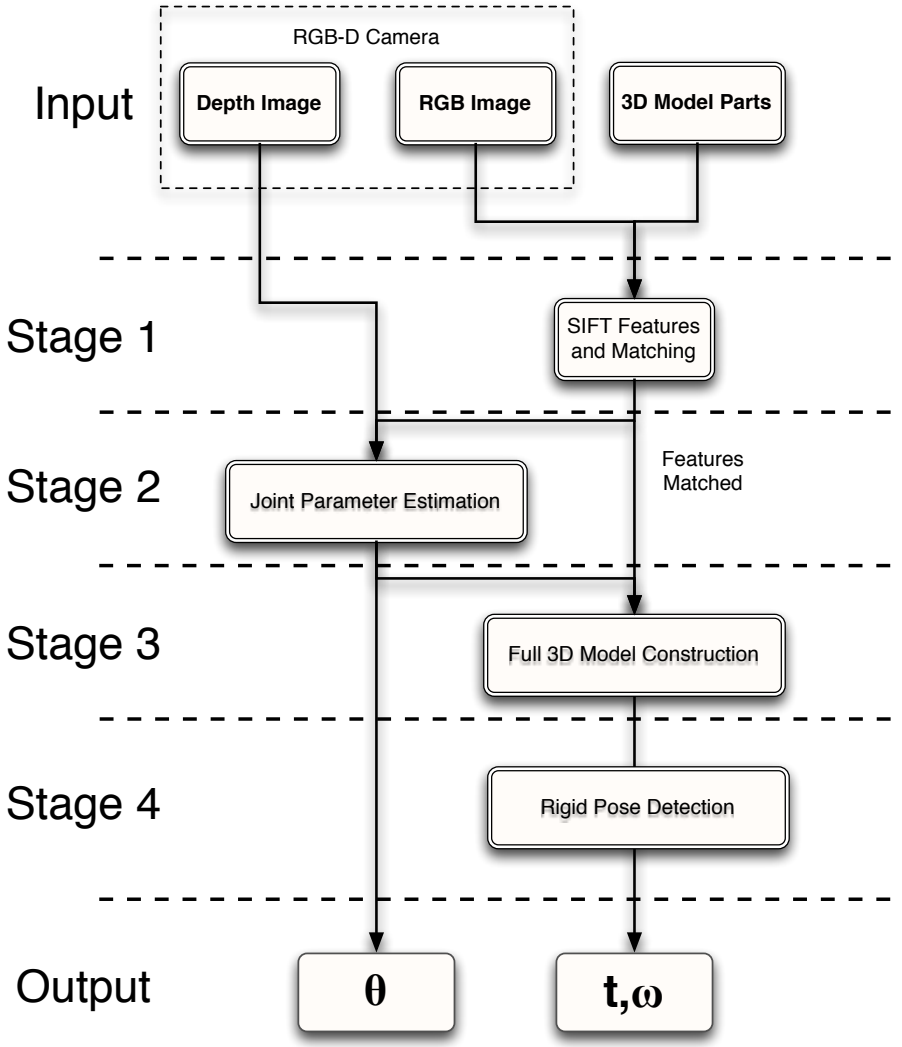


Figure 5.2: Proposed method overview. The method estimates the articulated object pose parameters by combining color and depth images with a 3D model. See Section 5.3 for a discussion of the different stages.

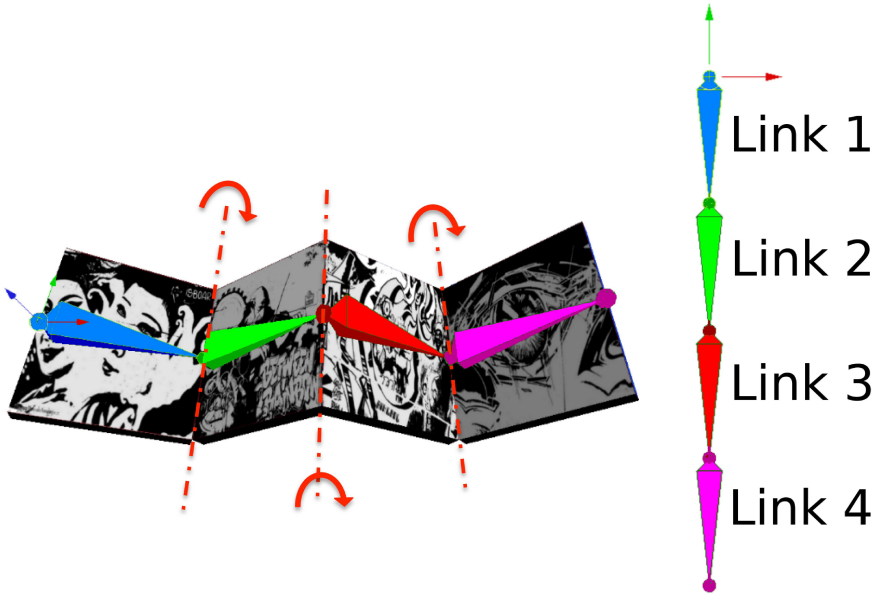


Figure 5.3: Articulated model with links hierarchy. The object's joints are connected by links that are organized in a hierarchy (right). Each link is associated with a rigid part of the object (left). Link one (blue) is the root of the hierarchy and its transformation represents the unconstrained 6DOF rigid translation and rotation of the object as a whole. In this particular example, links two to four each have one DOF (red arrows) and are attached to their parent links.

5.3.2 SIFT Features and Matching

The first stage of the algorithm (Fig. 6.2) is responsible for extracting the visual cues. This system extracts SIFT features (Scale-Invariant Features Transform) [22] from the grayscale image, and matches these with the feature codebook of the 3D model. This codebook is generated in a training stage where features are extracted from keyframes of rotated versions of the model (30° separation), and mapped to the surface model. The SIFT feature extraction from the image, the model codebook construction, and the correspondence matching between both are all performed using a fast

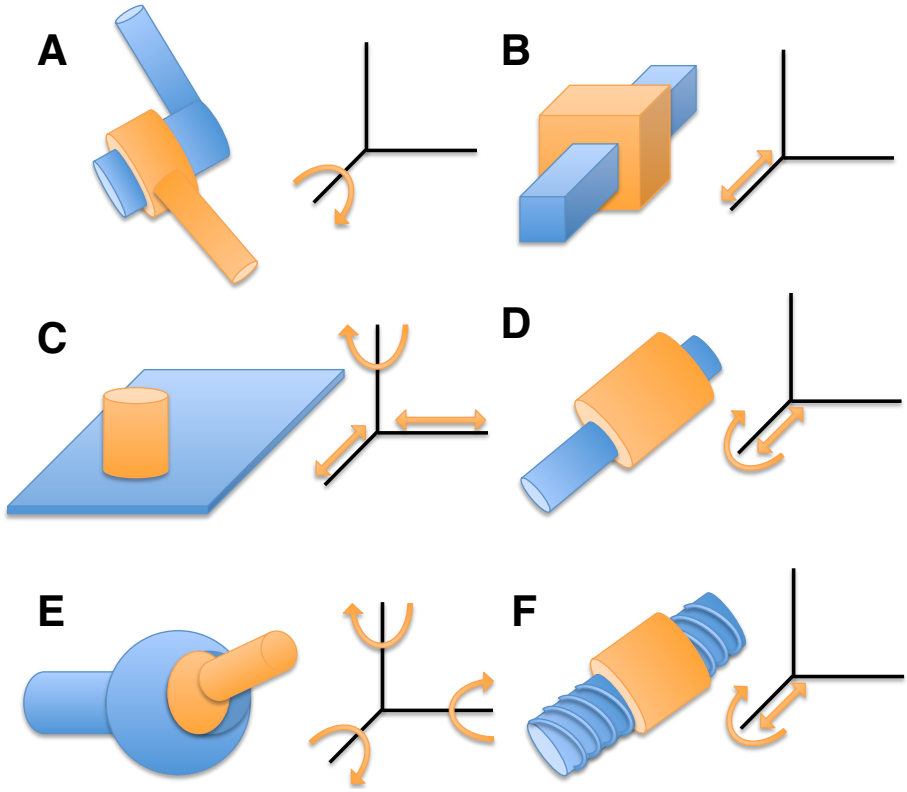


Figure 5.4: Types of joints: A revolute, B prismatic, C plane, D cylinder, E sphere, F helix.

GPU library [45]. As a result of this stage, each image SIFT feature is either assigned to a specific part of the model (as shown in Fig. 5.5) or discarded.

5.3.3 Joint Parameter Estimation

The algorithm's second stage estimates the internal parameters $\boldsymbol{\theta}$ of the kinematic chain that define the transformation of the joints of the artic-

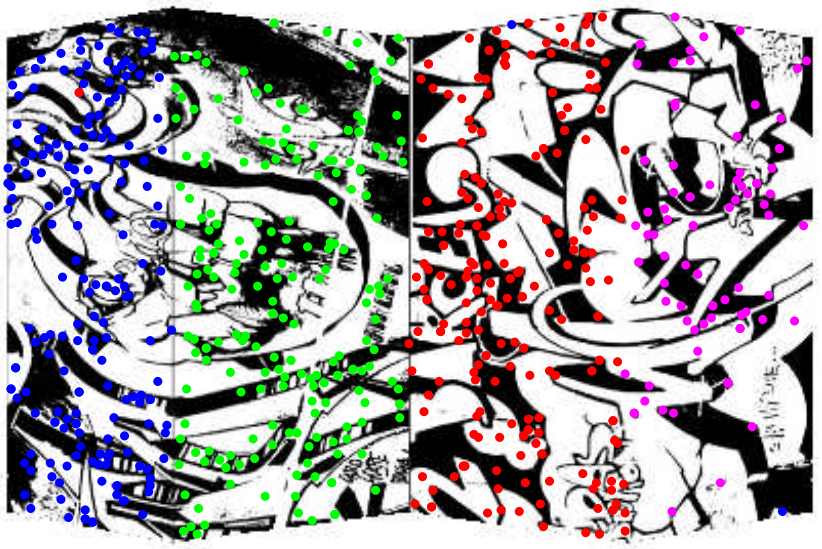


Figure 5.5: SIFT features matched between the image and the 3D model's codebook. The matches are colored depending on the part they belong to. The matches in blue and green will be used to compute the first joint's angle, the matches in green and red to compute the second joint's angle, and the red and magenta matches to compute the third joint's angle.

ulated object. The novelty of the proposed method lies in its ability to extract the parameters of a specific joint independently from the rest of the kinematic chain. As input, the method requires image SIFT features that are matched to the joint's parent and child links of the model. Figure 5.5 shows an example of matched features, colored according to the link they belong to.

Every SIFT feature that is successfully matched to the model can be associated to a 2D location in the image, a 3D location in the model's coordinate system, and (using the depth camera image) also a real-world 3D location. Consider the two features $\boldsymbol{p}$ and $\boldsymbol{q}$ in Fig. 5.6A that belong

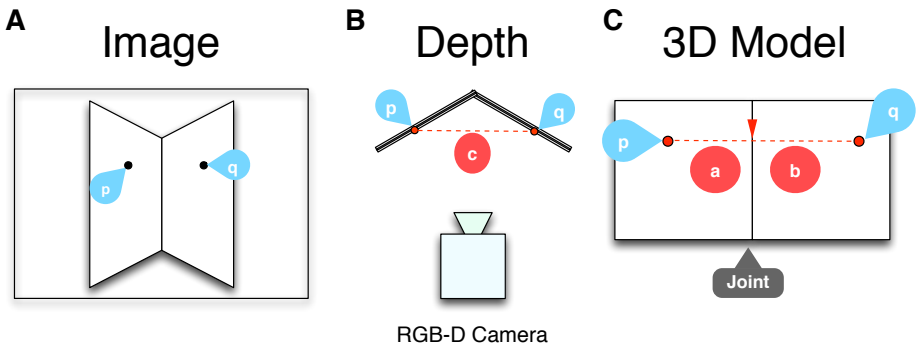


Figure 5.6: Representation of features in image space (A), depth space (B) and 3D model space (C). $\boldsymbol{p}$ and $\boldsymbol{q}$ are features of the articulated object's left and right parts, respectively. The real-world distance between the features, as measured by the RGB-D camera, is represented by c. The distances between the features and the included joint, as measured in the model, are represented by a and b.

to two parts connected by a joint. The proposed method derives the joint parameters by comparing distances between such feature pairs in world coordinates (c in Fig. 5.6B) to distances between these same features and the included joint as obtained from the model (a and b in Fig. 5.7C).

Formally, let $\boldsymbol{p} = (p_x, p_y, p_z)$ belong to link l_{k-1} and $\boldsymbol{q} = (q_x, q_y, q_z)$ to link l_k (Fig. 5.7). The Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ is equal to c. In order to have the same parameter value between the pair of points and the joint, we project the points onto the joint plane, XZ plane that contains $\boldsymbol{q}$ in Fig. 5.7 example. Considering $\boldsymbol{q}$ in the XZ plane and $\boldsymbol{p'}$ the projection of $\boldsymbol{p}$ onto the XZ plane, the distance c' between $\boldsymbol{q}$ and $\boldsymbol{p'}$ is computed as follows:

$$c' = \sqrt{c^2 - (p_y - q_y)^2} \ . \tag{5.3}$$

At this stage each joint type requires a specific approach.

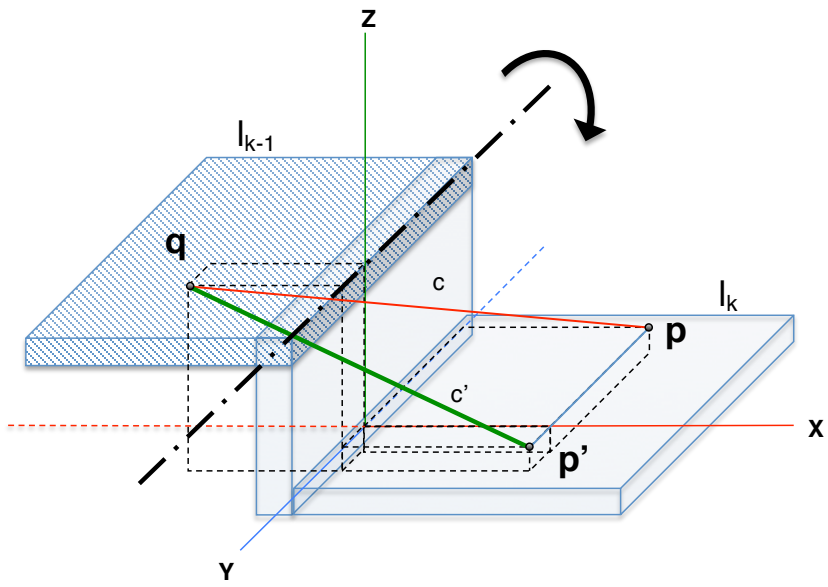


Figure 5.7: Articulated object with two links (solid color and striped color) and one included revolute joint. c is the distance between $\boldsymbol{q}$ and $\boldsymbol{p}$ and c' the projection of this same distance onto the XZ plane that contains $\boldsymbol{q}$.

5.3.3.1 Revolute Joint

We first consider the 3D articulated model in its initial (reference) configuration with $\theta_i = 0, \forall i$ (Fig. 5.8). Given two points $\boldsymbol{q}$ and $\boldsymbol{p'}$, c'_m represents the distance between both points and a and b represent the distances between the points and the included joint. All these distances are measured in the model according to its initial configuration. The reference angle α_m for this revolute joint (k) and this particular point pair is then obtained

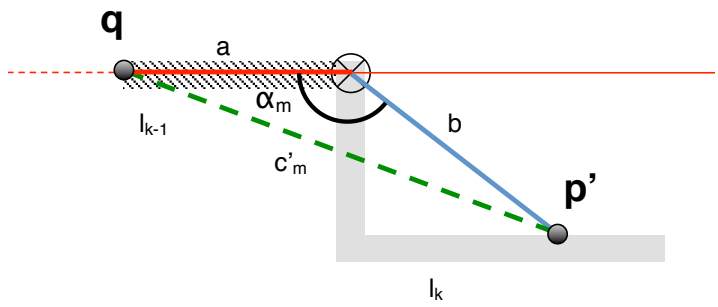


Figure 5.8: Initial configuration of an articulated object (revolute joint). α_m is the reference angle between points $\boldsymbol{q}$ and $\boldsymbol{p'}$.

using the law of cosines:

$$\alpha_m = \arccos\left(\frac{a^2 - b^2 - {c'_m}^2}{-2\,b\,c'_m}\right) , \tag{5.4}$$

only considering the solution in the range $[0, \pi]$. Note that this angle is not necessarily equal to zero in the initial configuration. Instead its value depends on the shape of the parts.

Considering now the observed scene, we can compute this same angle but now using the 'real' distance between $\boldsymbol{q}$ and $\boldsymbol{p'}$, c'_r, as measured by the depth camera. Replacing c'_m with c'_r in (5.4) then leads to two possible joint configurations as shown in Fig. 5.9. The joint parameter θ_k is expressed with respect to this observed angle, α_r, and therefore also has two possible solutions:

$$\theta_k = \{\beta_1, \beta_2\} , \tag{5.5}$$
$$\beta_1 = \alpha_m - \alpha_r , \tag{5.6}$$
$$\beta_2 = 2\pi - \alpha_m - \alpha_r . \tag{5.7}$$

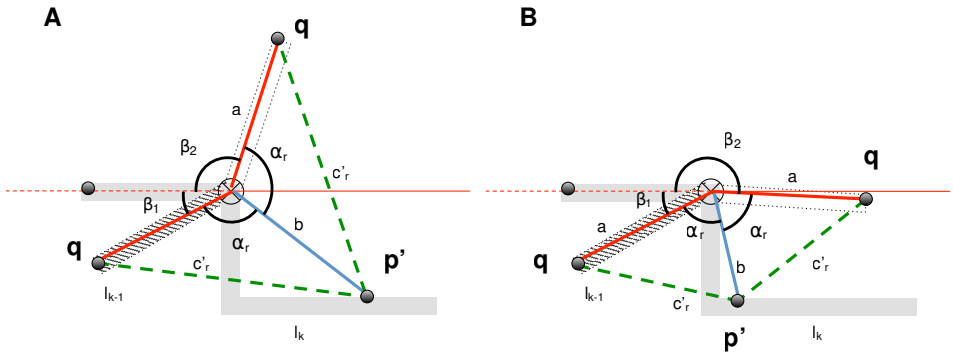


Figure 5.9: Two different point-pairs $\{\boldsymbol{p}, \boldsymbol{q}\}$ illustrating how α_r is used to obtain the two possible solutions for the joint parameters $\{\beta_1,\beta_2\}$. The reference configuration is shown in solid gray (*cf.* Fig. 5.8) and the striped area denotes the correct orientation of the link l_{k-1}.

Every point-pair provides two such possible solutions $\{\beta_1,\beta_2\}$, and the correct angle has to be decided upon. Figure 5.9 illustrates this situation for two different point-pairs and the same joint parameter θ_k. Note how the β_1's are identical in both cases and the β_2's are different. This occurs for all point-pairs. Figure 5.10 shows an example of possible joint parameters encountered in a real-world scenario for various point-pairs. To account for noise, the median $(\tilde{\beta}_1, \tilde{\beta}_2)$ and the standard error of each set of angles are computed and the median of the set with the smallest standard error is selected as the final θ_k value ($\tilde{\beta}_1$ in Fig. 5.10).

5.3.3.2 Prismatic Joint

A similar approach can be used for prismatic joints (see Fig. 5.4). Given two points $\boldsymbol{q}$ and $\boldsymbol{p'}$ (with $\boldsymbol{p'}$ the result of a projection as in Fig. 5.7), the reference distance d_m can be obtained from the distance between these points, c'_m, in the model's reference configuration as illustrated in Fig. 5.11. Considering now the distance d_r between the points $\boldsymbol{q}$ and $\boldsymbol{p'}$ as observed

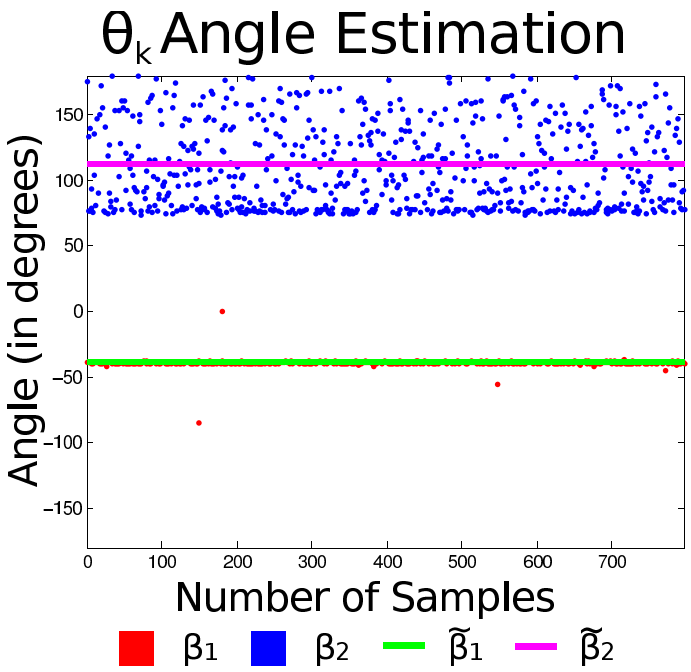


Figure 5.10: Possible joint parameters $\{\beta_1, \beta_2\}$ encountered for multiple point-pairs in a real-world scenario. The green line represent the median value for β_1 and the magenta line the median value for β_2.Note how the deviation from the median is much smaller for the correct parameter ($\tilde{\beta}_1$) than for the incorrect parameter ($\tilde{\beta}_2$).

in the scene (Fig. 5.12), the joint parameter θ_k is defined as:

$$\theta_k = d_r - d_m \ . \tag{5.8}$$

In this stage of the algorithm, regardless of joint type, if there is not enough information to determine the θ_k parameter of a joint (*e.g.* due to a part being occluded), its value is set to the initial configuration ($\theta_k = 0$).

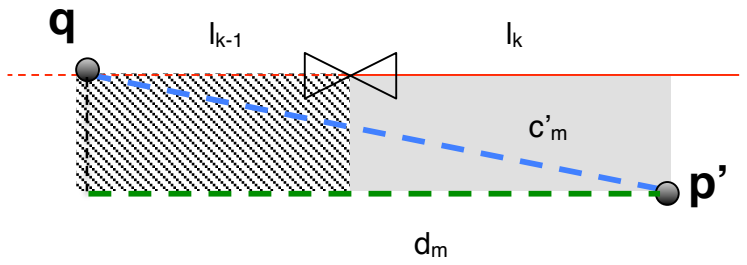


Figure 5.11: Initial configuration of an articulated object with a prismatic joint. d_m is the reference distance between points $\boldsymbol{q}$ and $\boldsymbol{p'}$.

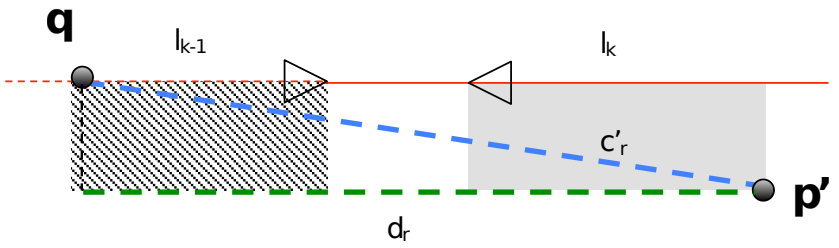


Figure 5.12: The observed joint distance d_r as derived from the distance between the two points $\boldsymbol{q}$ and $\boldsymbol{p'}$.

5.3.4 Full 3D Model Construction

Once all the internal configuration parameters $\boldsymbol{\theta}$ have been determined, they are applied to the model's kinematic chain (stage 3 in Fig. 6.2) and the articulated model is geometrically transformed from its initial (reference) configuration into the observed configuration by updating the 3D locations of the model codebook features. The goal of this stage is to change the shape of the model according to the observations. Note that the absolute location of the model is still unknown. This is the aim of the next stage.

5.3.5 Rigid Pose Detection

In this final stage, the articulated model (with shape updated according to the observations) is treated as a single rigid object. The updated codebook locations now allow estimation of the rigid pose parameters $\{\boldsymbol{t}, \boldsymbol{\omega}\}$ using standard perspective-n-point algorithms [46, 48].

The proposed articulated object pose detection framework thus computes the pose in a very similar manner as a rigid object pose detector (with identical first and fourth stages) but adds intermediate stages that adapt the shape of the articulated object to fit the rigid estimation scheme.

5.4 Experiments

We use a synthetic benchmarking dataset to compare the proposed method to a number of state-of-the-art methods.

5.4.1 Alternative Methods for Comparison

In this section we briefly describe alternative method that we use to compare with the proposed model.

5.4.1.1 Articulated Iterative Closest Point (AICP)

The AICP-algorithm [63] is an extension of the standard iterative closest point algorithm for rigid pose detection. The object's base location needs to be initialized reasonably accurate. This is done either manually or using an external method [63]. We consider the object's base as a rigid object and use standard rigid pose estimation to estimate it's location and orientation [46, 48]. With the base known, the algorithm then randomly iterates over the different joints estimating and applying the minimal error transformation that satisfies the joint's constraints. The algorithm uses a 3D object model and the scene depth extracted from the RGB-D camera.

5.4.1.2 Hierarchical Recognition (HR)

The HR-method estimates the articulated pose by evaluating a number of pose hypothesis against visual cues [60]. For the purpose of the comparison we adapt this method to the depth cues used here. As in AICP, the method is initialized with the estimated base of the kinematic chain. It then iterates through the hierarchy of links and creates hypotheses of possible poses that satisfy the joints' constraints. Each hypothesis is evaluated based on the consistency with the depth information and the best one selected.

5.4.1.3 Post-Imposed Constraints (PIC)

Proposed systems are frequently compared to a naive unconstrained approach where every object part is considered an independent rigid object with full six DOFs. We apply that idea here to construct a simplified approach of the proposed method. The PIC-method is a natural extension of the rigid object pose estimation method. It is composed of three stages. In the first stage every link is treated as an independent rigid object and no constraints are enforced between the links. Therefore the articulated model has six DOFs for each link, which are estimated using a rigid pose estimator that processes each part as a separate rigid object. The next stage then

enforces the joint constraints by computing the relative transformations between connected parts and restricting the DOFs. The joint parameters $\boldsymbol{\theta}$ that result from this operation are then used to modify the codebook of the object (*cf.* stage 3 in Fig. 6.2). Finally, a rigid pose estimator is applied to the updated model to estimate the rigid pose parameters $\{\boldsymbol{t}, \boldsymbol{\omega}\}$ as in the fourth stage of our method.

5.4.2 Synthetic Benchmark Dataset

To compare the different approaches we have generated a synthetic sequence using OpenGL. As described in Sec. 4.4.1, the rendering engine allows camera distortion representation together with mechanisms, such as *shaders* [64], that essay articulated object rendering. The sequence is 300 frames long and involves compressing and expanding the object from Fig. 5.3, which is located 50 cm from the camera. The object is located far from the camera in order to complicate SIFT feature detection for the appearance-based methods. Since the ground-truth of these scenes is known, the different methods can be compared with high precision. To evaluate the robustness of the different algorithms, additional sequences are created by corrupting the generated sequences either with Gaussian noise or an occluding object. In the grayscale images, the noise is directly applied to the pixels' intensity. The depth information on the other hand is distorted as a function of the distance to the RGB-D camera [65]:

$$Z' = \frac{f \cdot b}{d + \epsilon} \ , \tag{5.9}$$

where ϵ is the noise with standard deviation σ, f and b are the focal length and baseline of the sensor, and d is the disparity, defined as follows:

$$d = \frac{f \cdot b}{Z} \ , \tag{5.10}$$

with Z the ground-truth depth. Applying noise in the disparity rather than the depth domain results in a more realistic situation where near objects

are less affected than far objects. To explore the methods' robustness, we introduce different levels of noise from $\sigma = 0.0$ to $\sigma = 3.0$. Fig. 5.13 shows the effects of image and depth noise.

For the occluded sequences we introduce occluding objects that obscure a specific proportion of the articulated object (0%, 25%, 50%, and 75%). This occluder is located 5 cm closer to the camera than the articulated object. Some example frames with varying levels of occlusion are shown in Fig. 5.14.

Note that in this synthetic benchmark, we pursuit the creation of a controlled environment that tests the performance degradation against noise and occlusion. Since we combine appearance and depth information, any mismatch produced by a clutter background will be removed from the data by algorithm based on its depth. Therefore, in this test scenario, we did not focus on developing a realistic background, as in the previous chapter.

The developed benchmark has been released and it is now available at *http://www.leonardorubio.com/*.

5.5 Results

We first compare the four different methods on the synthetic benchmark with known ground-truth. Next, we demonstrate the reliability of the proposed method on challenging real-world scenarios.

5.5.1 Synthetic Dataset

Fig. 5.15 shows some typical results obtained with the four different methods on the original (A), noisy (B), and occluded (C) sequences. Each image of Fig. 5.15 contains a closeup of the area of interest, given the original frames of Fig. 5.13 and Fig. 5.14. Although representative, this is only a snapshot of the results.

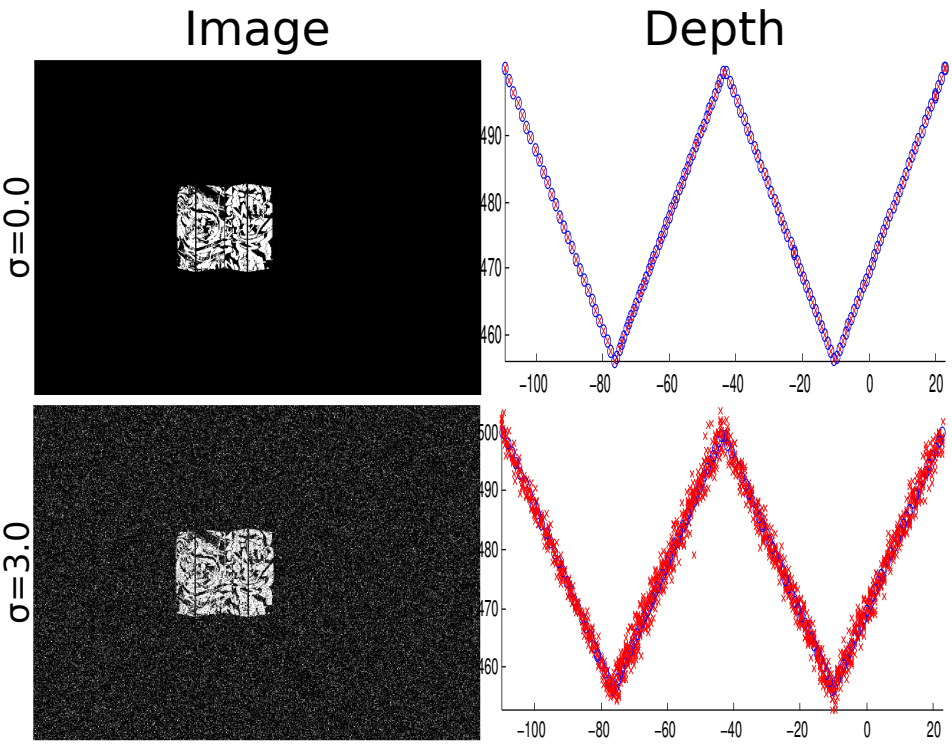


Figure 5.13: Example frames of the synthetic test sequences (Noise). The left column shows the effect of noise on the grayscale image. The left column shows the noise added to the depth information (the noise-free and noisy depth are shown in blue and red, respectively).

To quantitatively evaluate the performance of the different methods we compare the estimated articulated pose parameters $\boldsymbol{\tau}$ to the ground-truth pose $\boldsymbol{\tau}_{\text{gt}}$ across the complete set of synthetic sequences. The different pose parameters influence the object's location in different ways. Instead of directly comparing their values it is therefore more meaningful to measure the effect they have on the 3D model. Both $\boldsymbol{\tau}$ and $\boldsymbol{\tau}_{\text{gt}}$ are applied to the model and the maximum Euclidean distance of corresponding model vertices is used as the articulated pose error [47]. The validity of the estimated poses

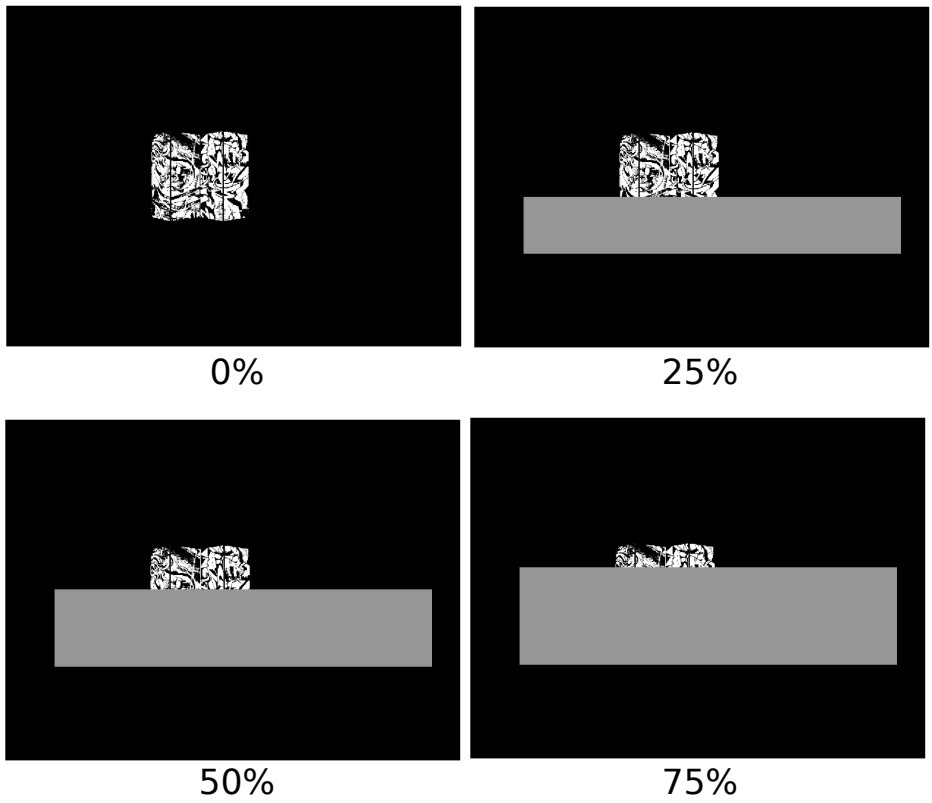


Figure 5.14: Example frames of the synthetic test sequences (Occlusion). The different levels of occlusion are shown in the different images.

is then determined by thresholding this maximum distance and the proportion of valid frames is used to summarize the performance on an entire sequence. Figure 5.16 shows the percentage of valid frames as a function of the distance threshold for each method for the original (A), 50% occluded (B) and noisy with $\sigma = 3$ (C) sequences. After fixing the distance threshold to 10 mm (red vertical lines in Fig. 5.16), the performance on all the noisy and occluded sequences can be represented more compactly as in Fig. 5.17.

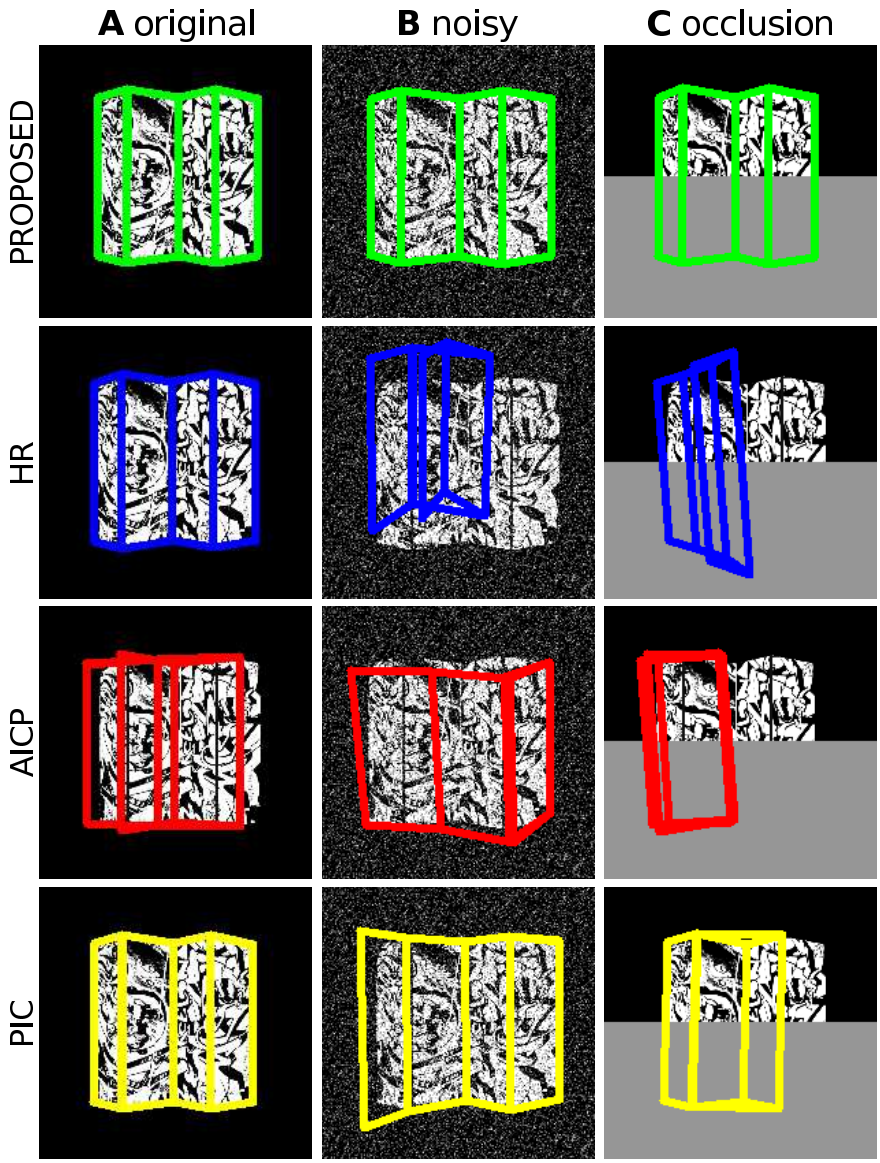


Figure 5.15: Example results obtained by the four methods in three different sequences. The images are closeups of the area of interest. (A) original sequence, (B) noisy sequence with $\sigma = 3$, and (C) occlusion sequence with 50% occlusion.

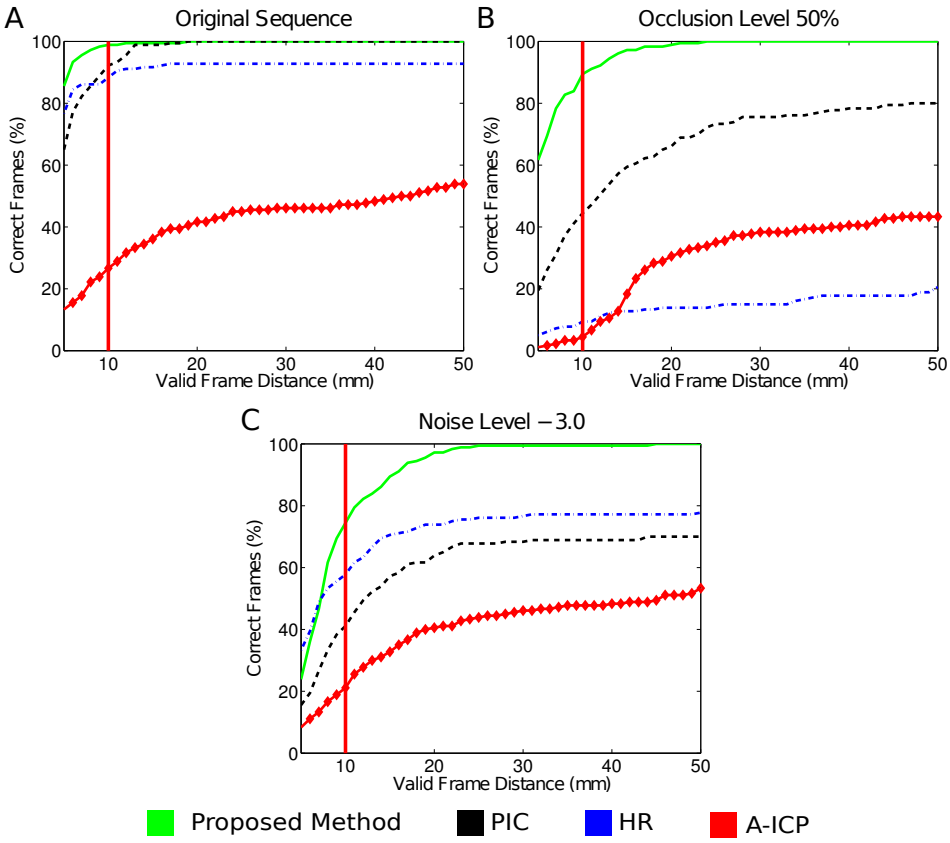


Figure 5.16: Comparative evaluation of the different articulated pose estimation methods on the synthetic dataset. A–C show how the performance changes as a function of the distance threshold. The vertical red lines represent the results for threshold fixed to 10 mm.

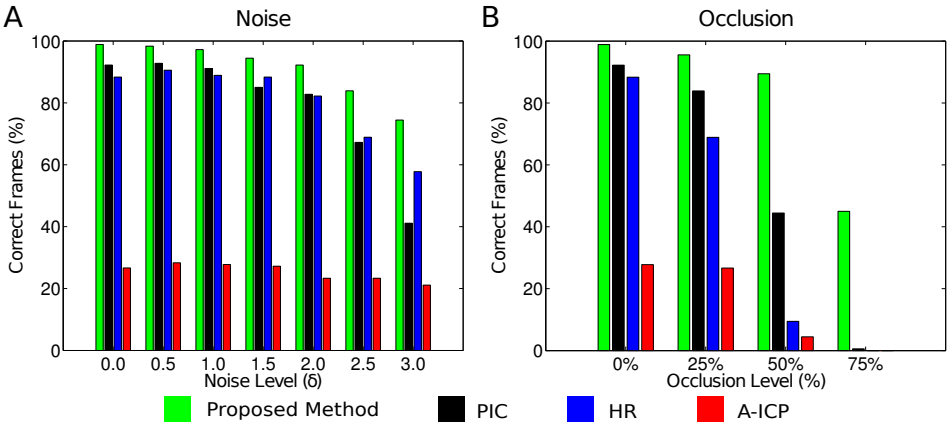


Figure 5.17: Comparative evaluation of the different articulated pose estimation methods on the synthetic dataset. A and B summarize the results obtained on all sequences with the distance threshold fixed to 10 mm.

The proposed method significantly outperforms the alternative methods in the noisy scenario (Fig. 5.17A), and keeps performing well even at very high noise levels. The HR and PIC methods also perform very well at low and medium noise levels but suffer at high noise levels. AICP fails dramatically regardless of noise level. Even greater improvements are achieved by the proposed method in the occlusion scenarios (Fig. 5.17B). At 50% occlusion the proposed method is only mildly affected whereas both HR and PIC suffer significantly more. Even at 75% occlusion the proposed method can still correctly estimate the pose on half the sequence. Also notable in this scenario is that PIC performs better than HR. As in the noisy situation, AICP performs very badly at all levels of occlusion.

5.5.2 Real-World Scenarios

We next evaluate the proposed method in a variety of challenging real-world situations. We create 30 different poses for each object. We stress again that the system is not provided with an initial position, nor does it rely

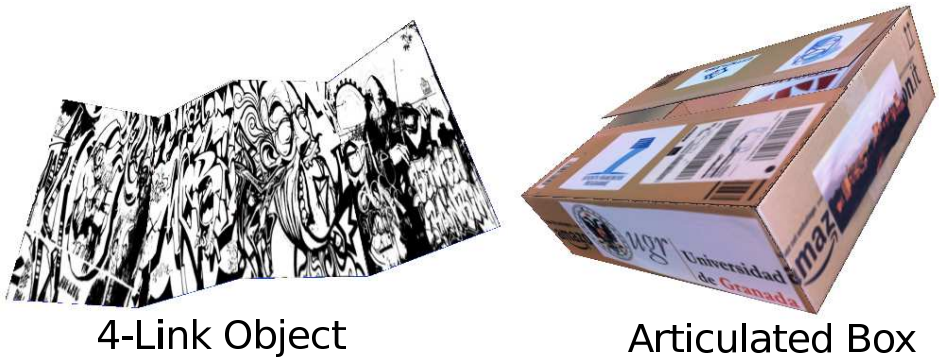


Figure 5.18: Articulated models of the objects used in the real-world evaluation.

on temporal information (such as the previous frame's pose) during pose estimation. The 3D model of the objects used in this experiment are shown in Fig. 5.18.

Some example results with a cluttered scene, occlusion, motion blur and/or extreme poses are shown in Fig. 5.19 for the *4-link object* and in Fig. 5.20 for the *articulated box*. The system performs adequately across a wide variety of articulation and environmental states. Fig. 5.19 shows the performance of the method in multiple challenging real-world situations involving strong deformation (A,B), self-occlusion (B), manipulator occlusion (C) and third object occlusion (D–F). In Fig. 5.20 different object poses are shown with strong deformations and occlusions. In general, the different parts of the object are estimated correctly but some problems occur when locating parts that are close to the image edges (Fig. 5.20B,F).

5.6 Discussion

The performance of AICP has been disappointing throughout the evaluation. This is mainly due to two reasons. Firstly, since AICP suffers from local minima, the initial pose needs to be sufficiently accurate. In this

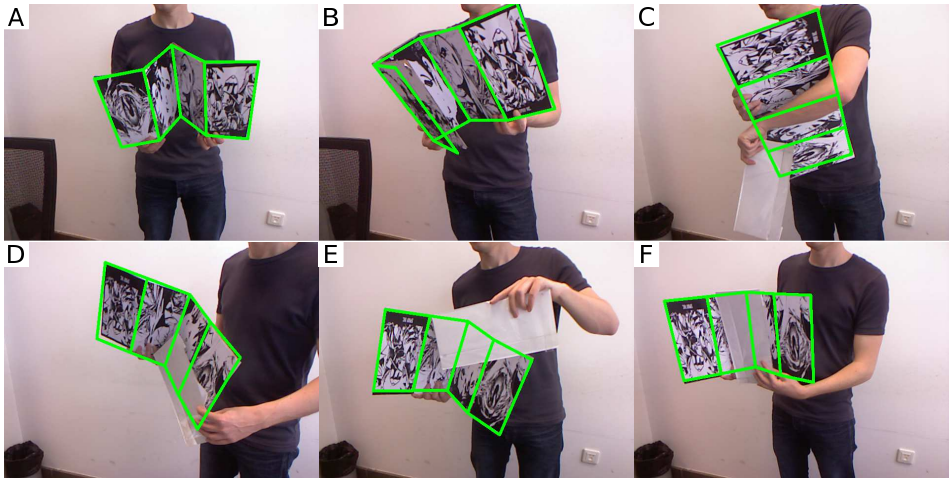


Figure 5.19: Real-world frames with the projection of the estimated pose in green for the *4-link object*. The frames show different scenarios: strong deformation (A,B), self-occlusion (B), manipulator occlusion (C) and third object occlusion (D–F).

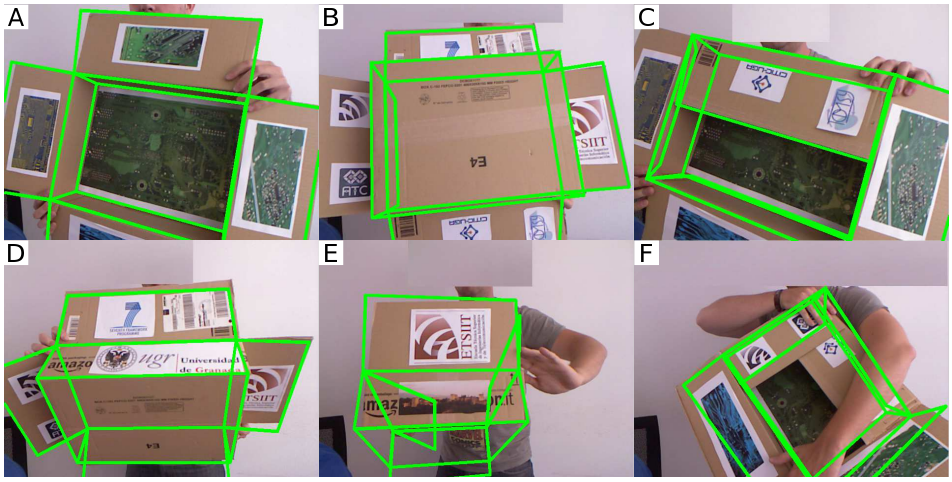


Figure 5.20: Real-world frames with the projection of the model according to the estimated pose in green for the *articulated box*. The frames show different scenarios: strong deformation, self-occlusion, and manipulator occlusion.

sense AICP is more a tracking than a detection method. In comparison, our method does not rely on such an initial configuration that can affect the method's behavior. Secondly, since the errors are accumulated through the hierarchical chain, the performance is lower for parts deeper in the hierarchy. On the other hand, AICP's performance is unaffected by the number of children of a particular joint since the errors of the different children are all independent of each other. AICP also has some strong points. The accuracy of the base location does not need to be as high as in HR for example, since AICP is able to refine the initial base pose. It also performs quite well in the presence of noise. In Fig. 5.17A the (albeit very low) AICP performance is almost unaffected by noise. Provided the initial pose is sufficiently close to the correct pose, the algorithm performs very well here.

As in AICP, HR is affected by the nature of the object's kinematic chain. Due to the accumulation of errors, a lower performance is achieved on parts deeper in the hierarchy. HR also depends much more strongly than AICP on the initial pose of the objects' base. A very good initial localization of the base is critical since the method does not refine it. As a result the error accumulation that starts at the base is more severe than in the other methods. The proposed method estimates each joint's parameter independently and is therefore not affected by the kinematic chain. On the positive side, HR does not suffer from local minima since the algorithm performs an exploration over the entire range of values. The number of values that can be evaluated is of course limited by the computational budget available to the system. Both AICP and HR are very sensitive to occlusions which is problematic in real-world applications, where the object can be occluded by hands, robot end effectors, or by itself.

The PIC method outperforms both AICP and HR but suffers in challenging situations where an insufficient number of image/model SIFT correspondences are available.

The proposed method is much less sensitive to this for two reasons. Firstly, it relies on depth information in addition to SIFT features. But

secondly and more importantly, by decoupling the estimation of the joint parameters from the rigid pose parameters, it requires fewer correspondences. For example, one point-pair is sufficient to estimate a joint angle, whereas PIC estimates the full 6DOF of both parts and thus requires at least four matches in each part.

One of the drawbacks of the SIFT-based methods (PIC and proposed method) is the requirement of textured objects in order to extract feature information. This is not required by the AICP and HR methods.

In terms of computational complexity, the proposed method has significant advantages over the other methods considered here. HR's computation time depends strongly on the depth of the object's kinematic chain (as does its performance). Due to the domain discretization, the computational complexity scales exponentially with chain depth. AICP is less strongly affected, but its iterative nature also increases the execution time drastically. With an increasing number of parts (regardless of chain depth) the complexity increases superlinearly since the joints need to be visited multiple times to achieve accurate alignments. The computational complexity of both the proposed and the PIC approach scales linearly with number of parts. However, since the PIC approach performs a full 6DOF pose estimation for each part of the object, its scale factor is much larger, whereas we only need to estimate the joint parameters.

5.7 Conclusions

This chapter describes a novel methods for articulated object detection that extents the rigid object pose detection system described in chapter 4.

The novel approach for articulated pose detection combines sparse SIFT features with depth information. The method is not restricted in terms of object shape or number of object parts and is highly accurate and robust to occlusions and noise. Our method outperforms other alternative methods in highly noisy scenarios and in the presence of large occlusions. The accuracy and robustness are achieved by decoupling the estimation of the

joint parameters from the rigid part of the articulated object's pose. This makes the proposed method computationally much simpler and suitable for parallel computing architectures. Since the existing articulated object benchmarks are oriented towards body pose estimation, we have introduced a novel benchmark dataset and ground-truth data in order to specifically evaluate our model in noisy and occluded scenarios. Using this benchmark we have shown how the proposed method greatly outperforms alternative methods. Finally, the methods merit has been demonstrated on complex real-world problems.

Chapter 6

Improved Pose Estimation

The second scientific contribution presented in this dissertation aims an improvement in pose estimation systems by reducing its requirements without decreasing performance at the same time. Using as starting point, the rigid object pose estimation system, presented in Chap. 4, we design specific enhancements that aim to reduce the tracking task requirements.

According to (Fig. 1.3), in this chapter we focus on object tracking systems that is a complex task in computer vision that has been widely addressed in the recent years since it is useful in a large number of applications (intelligent robots, monitoring surveillance, human-computer interfaces, vehicle tracking, biomedical image analysis, etc.). Being able to understand the environment in an efficient and robust manner, provides the ability of a correct interaction with the scene performing tasks such as planes tracking, object interaction, etc. These tasks require robust and efficient scene understanding [66, 67].

6.1 State of the Art

Following the general overview of the literature about pose estimation described in Chap. 3, in this section we continue the state-of-the-art review. In particular, we focus on 3D object tracking systems. Different approaches have been proposed for 3D object tracking. A number of methods operate on 2D image sequences and establish temporal correspondences between frames, using for example Kanade-Lucas-Tomasi feature tracking [24]. In [36], a 3D wireframe model of the object is reprojected on the image and the object pose is updated to minimize the differences between projected and observed edges. However, object tracking methods that operate on 2D images suffer from a low precision with regard to object movement in depth [68].

More recently, due to the prevalence of cheap RGB-D cameras, many methods have been introduced that exploit 3D data for object tracking. Most of these incorporate some variant of the Iterative Closest Point (ICP) algorithm [47, 69, 70]. ICP algorithms however are very sensitive to local minima. This can result in error accumulation and greatly reduce the tracker's robustness [71]. Model-based techniques that combine appearance and depth information yield more accurate estimations [47], but require detailed appearance and shape information of the tracked object.

Methods based on particle filtering (PF) can overcome these issues [35, 38, 72, 73], but are limited by the number of degrees-of-freedom (DOF) required to describe the tracked object's motion. The computation time typically scales very badly with increased DOFs [74]. To counter this, many approaches have been proposed to reduce the problem solution space. Some methods constrain the range of movement [75, 76] whereas others assume a static camera [74, 77, 78]. Although these strategies are effective in reducing the solution domain, they also very strongly limit the object's allowable motion, and consequently, they cannot be used in more general real-world scenarios.

Hybrid search methods on the other hand estimate the parameters in multiple steps, where the parameter values calculated in a previous step constrain the solution space for the subsequent set of parameters [74, 79, 80]. Such a hierarchical search can greatly reduce computational complexity.

6.2 Improved Pose Estimation Architecture

In this chapter, we present a novel object tracking method that uses a hybrid search strategy to separate the tracking problem into two different phases: scene plane tracking and particle filtering. In domestic environments, objects are frequently located on planes such as tables, floors, walls, shelves, etc. In the proposed method, the tracked object is associated with the environment plane on which it is located. This associated plane is detected and tracked through the sequence and its location is used to constrain the solution space of the subsequent particle filtering phase. The system provides highly precise 6DOF rigid object pose estimates in real-time on the basis of depth information supplied by RGB-D cameras at 30 Hz [81]. We also introduce a novel benchmark dataset consisting of real-world sequences involving different tracking scenarios. Using it, we demonstrate improved performance of the proposed method as compared to some state-of-the-art methods.

According to this description, we create a modification that adapts the generic architecture to the described system.

Fig. 6.1 shows the blocks configuration adopted for this chapter. The input data is provided by an RGB-D camera sensor (see Sec. 6.3.1). The models used are pure geometric models. In contrast to the appearance models used in Chap. 4 and Chap. 5, the model used for the improved pose estimation are simplified and only require 3D geometric shape information (see Sec. 6.3.3). Therefore, no appearance features are extracted in this chapter. Instead, the 3D scene in segmented in meaningful regions such as planes and objects (see. Sec. 6.3.2). Since we are working with a tracker, we define a new pose estimation method base on particle filter (Sec. 6.3.5).

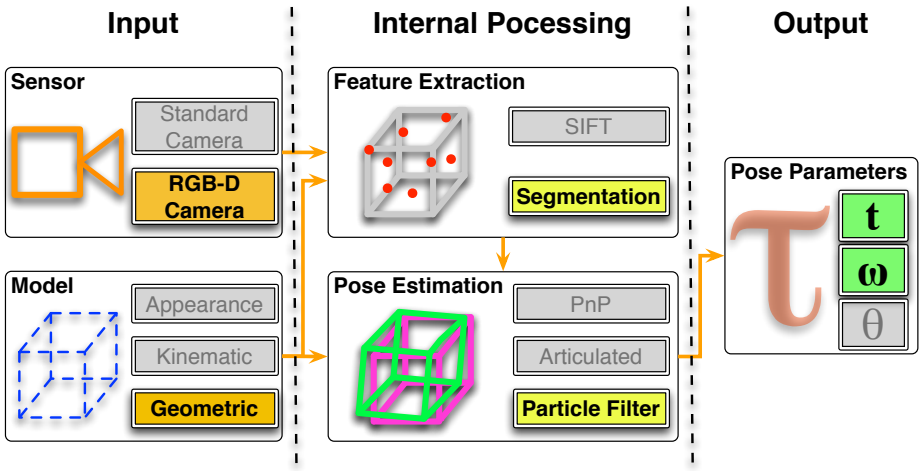


Figure 6.1: Improved Pose Estimation Architecture.

With this system, we are dealing with rigid object tracking, hence, the object state τ is defined with a translation t and a orientation ω.

6.3 Proposed Method

The proposed algorithm uses a hybrid search strategy to combine particle filtering with plane tracking. Tracking the planes that compose a scene provides environmental information to the particle filter method. Decomposing the scene in its different planes drastically reduces the solution space of the particle filter.

The method operates by relating an *associated plane* to the tracked object (e.g. ground plane) and tracking both throughout the sequence. Figure 6.2 contains a schematic overview of the proposed method. The method can be divided in six stages, which are discussed in detail in the next sections.

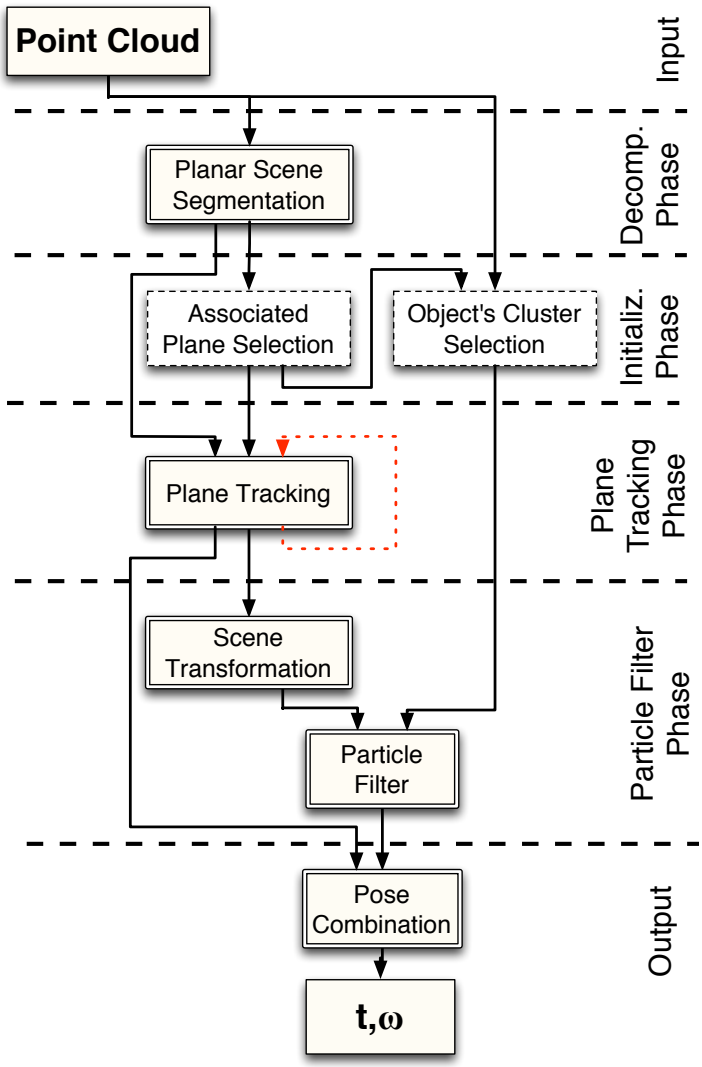


Figure 6.2: Proposed method overview. The input point cloud is decomposed into planes and used in each step of the algorithm. The steps represented as dashed blocks in the Initialization Phase are only executed at the beginning of the tracking process. The red dashed line represents the tracked plane information that will be used in the next frame's plane tracking phase.

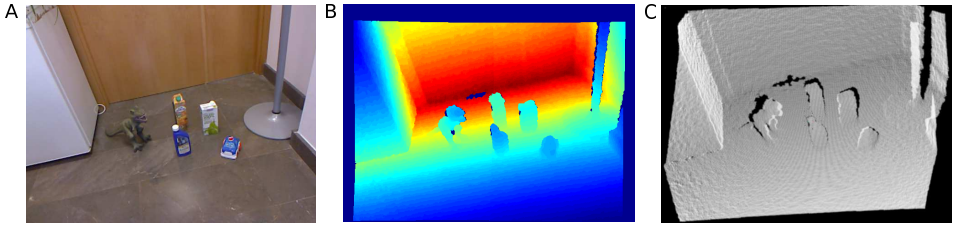


Figure 6.3: RGB-D camera data. Color (A) and depth (B) images at 640×480 resolution together with the point cloud (C) generated from the depth image (307,200 points). Due to the amount of data and the image size, individual points are difficult to distinguish in the point cloud.

6.3.1 Input

The RGB-D camera provides color images (Fig. 6.3A) and scene depth measurements (Fig. 6.3B). The depth data consists of a point cloud (Fig. 6.3C) composed of scattered 3D points that represent the scene's depth information. The proposed method only relies on point cloud data as input and uses it in every step of the algorithm. The input point cloud is constructed from a depth image of 640×480 pixel resolution (307,200 points). To improve efficiency, we downsample the input depth image to 320×240 pixel resolution (76,800 points) (see Section 6.5.1).

6.3.2 Decomposition Phase

The proposed method extracts a simplified planar representation of the environment and uses it together with the input point cloud in every phase of the algorithm. This *Decomposition Phase* is performed at the beginning of the algorithm and repeated for every input frame. The *Planar Scene Segmentation* segments the point cloud into different scene planes based on the normals associated with the points. Groups with more than 100 coplanar points (allowing for up to 3° orientation difference in the normals) are considered scene planes. These specific parameter settings result in the optimal segmentation for the environments considered in this work.

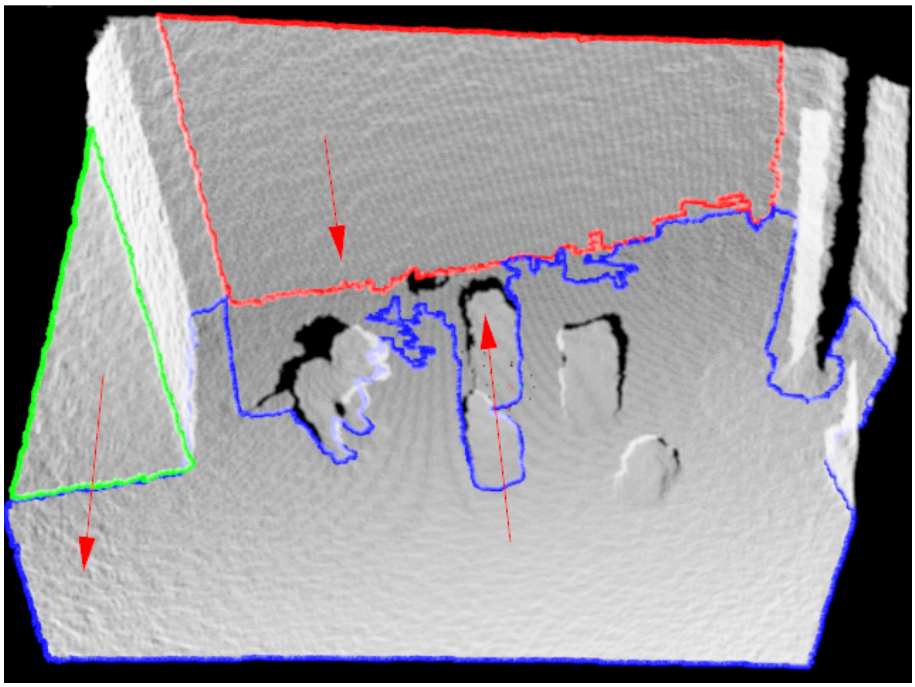


Figure 6.4: Planar scene segmentation of the input shown in Fig. 6.3. Three planes are detected in this scene. The different colors represent their approximate boundaries and the red arrows represent each plane's normal. Small planar regions and regions with excessive variability in the normals are not considered planes and therefore not labeled in this phase.

Figure 6.4 shows a scene composed of three planes. The boundaries of the planes are shown in color and their normals are depicted by the red arrows. The output of this *Decomposition Phase* consists of the detected plane regions and their normals.

6.3.3 Initialization phase

To maintain general applicability, the proposed system does not rely on predefined object models or scene information. Rather, the tracked object and its associated plane need to be selected by the user in an *Initialization Phase*. This phase is executed once, at the beginning of tracking, and relies on the point cloud together with the planes extracted in the *Decomposition Phase*. The associated plane of interest is selected among the possible planes through the user interface. Next, the points above this plane (using the normal as a direction indicator) are subtracted from the point cloud and grouped according to the Euclidean distance. A threshold equal to two centimeters is used to determine whether a point belongs to a cluster or not. This threshold correctly groups objects of the sizes considered in this work. Figure 6.5 shows the same input scene as in Fig. 6.3 with the three detected planes now represented in pastel color. The mustard-colored plane is the selected associated plane. On top of this plane we can see six different groups of point clouds. The group corresponding to the object of interest is selected through a specifically designed user interface that allows for manually selecting any object and plane in the scene. This selection task could be automated as well.

Once the associated plane and object point group are selected, the *Initialization Phase* is complete.

6.3.4 Plane Tracking Phase

The *Plane Tracking Phase* is responsible for tracking the associated plane throughout the sequence. The region corresponding to this selected plane is allowed to move freely with six DOFs to accommodate for camera motion.

We define a plane as a set of n 3D points:

$$P = \{M_0, \ldots, M_n\}\,, \tag{6.1}$$

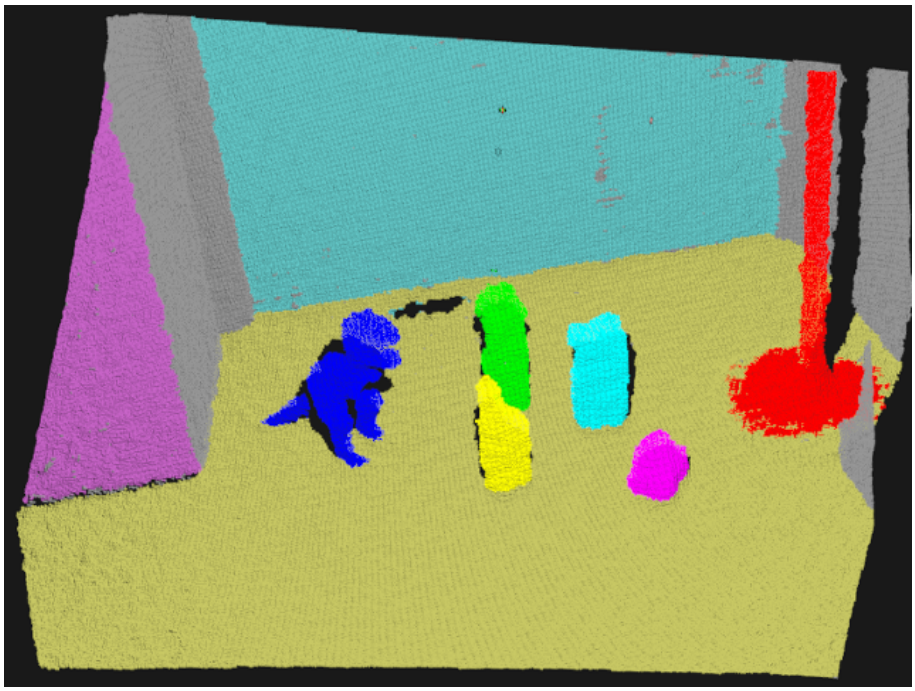


Figure 6.5: Scene point cloud with three detected planes represented in pastel color. The selected associated plane is shown in mustard color. The points above this plane are grouped together and shown in bright colors. A total of six different object groups are detected here.

At time t, the new planar scene segmentation S is defined as:

$$S = \{S_0, \ldots, S_k\}\,, \tag{6.2}$$

where k defines the number of scene's planes. The plane region selected at the previous time $t-1$ is defined as P_{t-1}. The scene segmentation S and the selected plane P_{t-1}, are used as input. Initially, at time t_0, the selected plane is provided through the *Initialization Phase*.

Both the newly detected scene planes at time t and the selected plane at time $t-1$ are first transformed into the 2D coordinates of the image plane using:

$$m_i = KM_i, \tag{6.3}$$

where m_i is a 2D point in the image plane, M_i is a 3D point of the scene and K is the reprojection matrix. The reprojection of the selected plane P'_{t-1} and the scene segmentation S' are compare in order to find the tracked plane.

The amount of overlapping pixels between new and old plane regions are then determined and the newly detected scene plane that maximally overlaps with the plane selected at the previous time is computed using:

$$P_t = \max_{i=0:k} f(P'_{t-1}, S'_i), \tag{6.4}$$

where f is the function that determine the number of overlapping pixels between two planes and P_t becomes the new associated plane at time t. At the next time instance $t+1$, this new associated plane will then be used in the same way to further update the associated plane region.

The selected plane is then determined by its normal $n = (n_x, n_y, n_z)$ and a point on the plane $X_0 = (x_0, y_0, z_0)$. In the remainder, all the coordinate system transformations are described in relation to a camera system with X-axis rightward, Y-axis upward and Z-axis backward. The normal (n) of the selected plane P_t is align with the Y-axis of the camera coordinate system $(u = (0, 1, 0))$. We rotation between vector by the quaternion $Q = (w, x, y, z)$ that is defined by:

$$Q = \left[cos\left(\frac{\alpha}{2}\right), sin\left(\frac{\alpha}{2}\right) v_x, sin\left(\frac{\alpha}{2}\right) v_y, sin\left(\frac{\alpha}{2}\right) v_z \right], \tag{6.5}$$

where the rotation axis v is defined by $v = n \times u$ and the angle between vector is computed with:

$$\alpha = acos\left(\frac{n \cdot u}{\|n\| \, \|u\|}\right). \tag{6.6}$$

The quaternion Q is transformed into a rotation matrix using (6.7).

$$R = \begin{pmatrix} 1 - 2y^2 - 2z^2 & 2xy + 2wz & 2xz - 2wy \\ 2xy - 2wz & 1 - 2x^2 - 2z^2 & 2yz + 2wx \\ 2xz + 2wy & 2yz - 2wx & 1 - 2x^2 - 2y^2 \end{pmatrix} \tag{6.7}$$

After the rotation transformation R, we align the two planes (P_t and XZ-plane) with a translation defined by $t = (0, -y_0, 0)$. The final plane transformation is represented by:

$$T_{cp} = [R|t]. \tag{6.8}$$

The points above the selected plane at time t now represent the spatial domain in which the tracked object is assumed to be situated. The points in this region are then extracted from the scene's point cloud and, together with the plane's region at time t and its transformation, constitute the output of the *Plane Tracking Phase*.

6.3.5 Particle Filter Phase

The proposed method links the tracked object to its associated plane. The location and orientation of this plane guides the object search by constraining its solution space.

The *Scene Transformation* step aligns the region-of-interest obtained from the *Plane Tracking Phase* with the camera coordinate system using the associated plane's transformation T_{cp}. After this transformation, the associated plane is aligned with the XZ-plane and the tracked object is situated on top of this XZ-plane.

We use a particle filtering technique to track the object pose. Particle filters can be used to predict the hidden state of a system based on temporal

information and sensor measurement of the system. The probability density of the state is represented by a set of particles with associated weights. Each particle represents a different state hypothesis and its weight corresponds to the current level of confidence associated with this hypothesis. At each time step, the particles are updated according to the sensor measurements and an evolution model that incorporates historical information. The state can be estimated by combining all the particles according to their weights. See [82] for more details.

In the case of object tracking, the particle filter's state vector $\boldsymbol{\tau}$:

$$\boldsymbol{\tau} = \{\boldsymbol{t}, \boldsymbol{\omega}\}, \tag{6.9}$$

consists of a translation vector $\boldsymbol{t} = [t_x, t_y, t_z]$ and a rotation axis vector $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$. The six parameters of the state vector $\boldsymbol{\tau}$ correspond to the six DOFs of the object.

In the proposed method, the position and orientation of the associated plane provides initial information about the object's location. By transforming the scene's region-of-interest according to the plane's transformation T_{cp}, the object's DOFs are reduced to three (x and z location on the XZ plane and Y axis rotation). In this spatial domain, the object state vector is defined by only three parameters. The particle hypotheses are thus also generated for a three parameter state vector.

To determine the particle accuracy, the points associated to the object's group as defined in Section 6.3.3 are reprojected into the scene. The particle's weight is then obtained by (6.10) that sums the Euclidean distances between every point of the object's group (M_i) and their nearest point in the point cloud (n_i) which is computed using a fast k-d-tree algorithm [83].

$$e = \sum_{i=1:k} d(M_i, n_i). \tag{6.10}$$

The particle filter combination leads to the optimal system state estimation ($\tau = [t_x, t_z, \omega_y]$). The ω_y rotation is transformed into its quaternion form:

$$Q = \left[cos\left(\frac{\omega_y}{2}\right), 0, sin\left(\frac{\omega_y}{2}\right), 0\right], \tag{6.11}$$

and then, into a rotation matrix (R) using (6.7). The translation is defined by $t = [t_x, 0, t_z]$ and the transformation matrix between the tracked object and the selected plane by:

$$T_{po} = [R|t]. \tag{6.12}$$

T_{po} is the output of the *Particle Filter Phase*.

6.3.6 Output

In the final phase the tracked object pose is computed by combining the plane orientation with the object location. The plane orientation defines the transformation matrix $T_{cam-plane}$ between the camera coordinate system and the plane coordinate system. The object location on the associated plane defines the transformation matrix $T_{plane-obj}$ between the object and the plane coordinate system. The final transformation between the object and the camera coordinate system is computed with:

$$T_{cam-obj} = T_{cam-plane} \cdot T_{plane-obj}. \tag{6.13}$$

The transformation matrix $T_{cam-obj}$ defined in (6.13) contains the 6 DOFs parameters $\{\boldsymbol{t}, \boldsymbol{\omega}\}$ that define the tracked object pose in the 3D space.

6.4 Experiments

We introduce a novel real-world benchmarking data set to compare the proposed method to a number of alternative state-of-the-art methods.

6.4.1 Alternative Methods

6.4.1.1 Iterative Closest Point (ICP)

ICP algorithms iteratively minimize a distance error function between the scene point cloud and the object's point cluster. Since correspondences are unknown, at each iteration, each point of the object's cluster is associated with the nearest point in the scene's point cloud. Using these hypothesized correspondences, the algorithm then estimates and applies the cluster transformation that minimizes the distances between these corresponding points. The correspondences are then re-established and the process is repeated a number of iterations. To test the ICP algorithm with point cloud data, we use the implementation included in the point cloud library [70].

6.4.1.2 Six Degrees-of-Freedom particle filter (6D-PF)

In this approach, a standard particle filtering technique is used, without domain reduction. This method thus estimates a 6DOF state vector and the generated particles are the full six-dimensional problem domain. The 6D-PF implementation using point cloud data is also implemented in the point cloud library [70].

6.4.2 Real-World Benchmark Dataset

To compare the different approaches we have recorded a total of nine sequences in different real-world scenarios. Each sequence is between 400 and 800 frames in length, and was recorded at 30 frames per second (FPS) using an in-hand RGB-D camera[1].

For six of these sequences, we have used the *Augmented Reality Toolkit* (ARToolkit) [13] to obtain ground-truth pose data. ARToolkit estimates the position and orientation of physical markers in relation to the camera

[1]The benchmark sequences are available at `http://www.leonardorubio.com/`

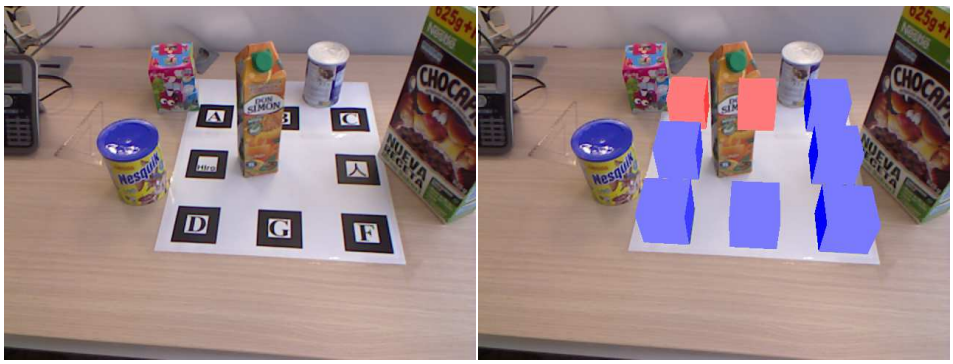


Figure 6.6: Pose estimation using ARToolkit. The left image shows the eight physical markers used and the right image shows the estimated pose of the markers. Detected markers are shown in blue and non-detected markers are shown in red.

(see Fig. 6.6). The marker's poses can be estimated with millimeter accuracy [13], which is sufficient for our purposes. ARTookit and the proposed method have different coordinate systems. Both are calibrated with respect to each other in the *Initialization Phase* (Section 6.3.3) when the object-of-interest is selected. In the six sequences recorded with ARToolkit we move the camera rather than the object since the object needs to maintain the same distance to the markers on the plane (Fig. 6.6) throughout the sequence. Multiple objects, differing in shape and material properties, feature in these sequences. The objects used are shown in Fig. 6.7. Example frames of these ARToolkit sequences are shown in Fig. 6.8A–F.

An additional three sequences were recorded with more challenging scenarios involving high-speed camera motion, independent object motion, occlusion, etc. Since the objects move independently in these sequences, ARToolkit could not be used to provide the ground-truth pose. Example frames of these sequences are shown in Fig. 6.8G–I.

Figure 6.7: Different objects used in the real-world benchmark sequences. The objects differ in terms of shape and material properties. From left to right: *dino*, *car*, *cleaning*, *milk*, *juice*.

6.5 Results

We first analyze the efficiency of the different methods and highlight the factors responsible for their performances. We then quantitatively evaluate the methods' accuracies using the sequences with ground-truth pose, and qualitatively compare the methods on the more complex sequences.

6.5.1 Computation Complexity

For particle filtering, each object point's nearest neighbor in the scene point cloud needs to be determined to evaluate the pose error. This needs to repeated for each particle in order to compute its weight. With M and n the number of object and scene points respectively, N_p the number of particles, and given that we use a fast k-d-tree algorithm of complexity $O\left(log(n)\right)$[83], the complexity of the particle filter used in the 6D-PF model is then $O\left(N_p M log(n)\right)$.

The proposed method requires plane tracking as an additional step before the particle filtering. This stage has $O\left(n\right)$ complexity [84], so the final

Figure 6.8: Example frames of the different benchmark sequences. (A) *car* (517 frames), (B) *dino* (516 frames), (C) *juice* (404 frames), (D) *milk* (786 frames), (E) multiple objects (840 frames), (F) occlusions (464 frames), (G) independent *car* object motion (319 frames), (H) complex scenario with high-speed camera motion and occlusions (512 frames), (I) camera motion sequence with high-speed motion and occlusions (512 frames). (A)–(F) have ground-truth obtained through ARToolkit. (G)–(I) are more complex but without ground-truth.

complexity for the proposed method is then $O(n) + O\left(N_p M log(s)\right)$ where s is a subset of the scene's point cloud.

ICP's complexity is equal to $O\left(N_i M log(n)\right)$ [85] where N_i the number of ICP iteration, and M and n the number object and scene points as before. As in the particle filter, the k-d-tree algorithm, with complexity $O\left(log(n)\right)$, is used to find each object point's nearest neighbor in the scene's point cloud.

6.5.1.1 Parameter Settings

Since the number of scene points n directly affects the performance, we down-sample the scene data to 76,800 points. With n fixed, the computation times depend on N_p, N_i and M. The parameters of each system are tuned so that each can operate in real-time at the framerate of the RGB-D camera, which is equal to 30 Hz. The same Intel i7-2600 3.4GHz with 8GB RAM was used for all algorithms.

For the ICP method, we allow up to 20 iterations (N_i) in order to assure the algorithm has converged [86]. Given $n = 76,800$ and $N_i = 20$, up to 250 object points (M) can be used in order to achieve 30 Hz. The proposed method requires associated plane tracking and particle filtering. With $n = 76,800$ and $M = 250$, a maximum of 130 particles (N_p) can be used to achieve 30 Hz operation (plane tracking requires 10 ms and particle filtering 25 ms). With the same n and M configuration, the 6D-PF system on the other hand, can only accommodate up to 50 particles for 30 Hz operation. The reason for this is that in the 6D-PF system the particle filter needs to operate on the entire point cloud, as opposed to the much smaller region-of-interest used by the proposed method.

The efficiency parameters discussed in this section allow adapting the proposed system to alternative platforms where the availability of resources may be limited, such as mobile phones.

6.5.2 Quantitative Comparison

To quantitatively evaluate the performance of the different methods we compare the estimated object pose $\boldsymbol{\tau}$ to the ground-truth pose $\boldsymbol{\tau}_{gt}$. The different pose parameters influence the object's location in different ways. Instead of directly comparing their values it is more meaningful to measure the effect they have on the 3D object points. Both the estimated $\boldsymbol{\tau}$ and ground-truth pose $\boldsymbol{\tau}_{gt}$ are applied to the object points and the mean Euclidean distance between corresponding points is used as the pose error [47]. The estimated pose is determined correct if this distance error is below a

defined threshold. The proportion of correct frames can then be used to summarize the performance of an entire sequence into a single value.

As discussed in Section 6.5.1, we use 250 object points and allow for 20 ICP iterations. For 6D-PF and the proposed method the maximal number of particles (efficiency-wise) are 50 for 6D-PF method and 130 for the proposed method. Since a lower number of particles decreases performance we allow both methods to use 130 particles. In this way the comparison is more interesting, but note that 6D-PF cannot process this many particles in real-time. Figure 6.9 contains pose estimates obtained by all three algorithms for representative frames of the benchmark sequences.

We use a 15 mm threshold on the distance error between the estimated τ and ground-truth τ_{gt} pose to determine the correctness of each frame's estimate. Based on our experience, the particle filter methods are more likely to recover from failures than the ICP method when the pose estimation error is large. Since this greatly decreases the proportion correct frames for the ICP method, we have cut the sequences once the ICP error becomes too large. The sequences described in Fig. 6.8 are thus delimited either by the length of the sequence itself or by the first frame where the ICP error exceeds 10 cm.

The proportion correct frames obtained by the three methods on the benchmark sequences are summarized in Fig. 6.10. The proposed method significantly outperforms the alternative methods in all the scenarios. The ICP method presents a substantially lower performance and is severely affected by small objects, clutter, and occlusions. In every scenario, the 6D-PF also performs worse than the proposed method even when using the same number of particles. We would like to remind that this amount of particles implies a much higher execution time for the 6D-PF method.

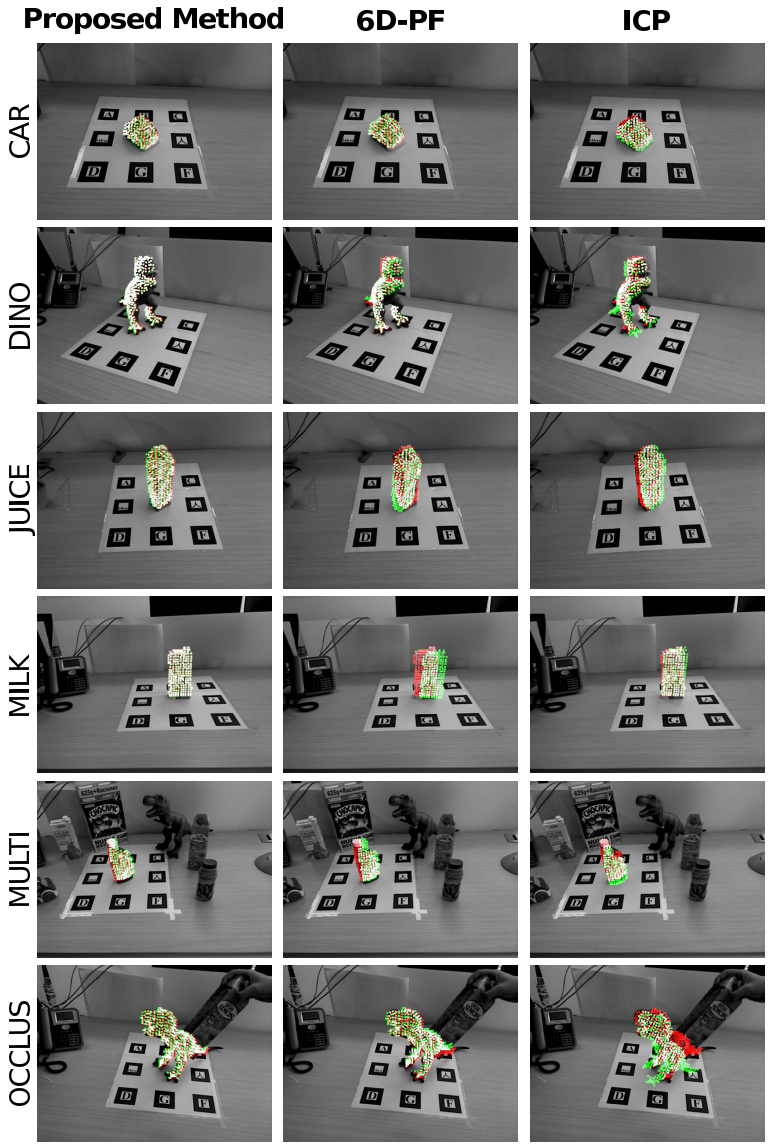


Figure 6.9: Representative frames of the benchmark sequences. Each row shows the pose estimates obtained by each method on the same frame. The object was transformed according to the estimated (green dots) and ground-truth pose (red dots) and projected in the image. In case the estimated and ground-truth dots overlap, they are colored white.

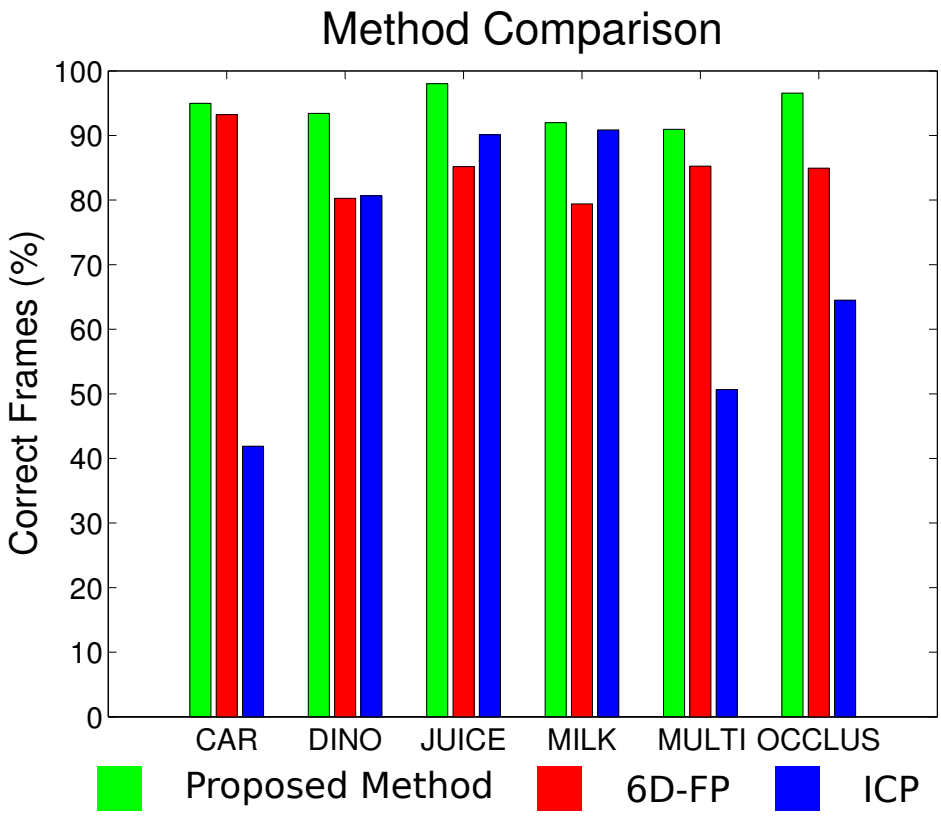


Figure 6.10: Quantitative comparison between the different methods. The bars show the percentage of correctly tracked frames in the different sequences.

6.5.3 Qualitative Comparison

We now qualitatively compare the methods in more challenging scenarios. Ground-truth could not be obtained here using the same procedure since the object's are moving in the scene.

Fig. 6.11 shows two representative frames of the *object's motion* (Fig. 6.8G) and *complex scenario* (Fig. 6.8H) sequences together with the pose esti-

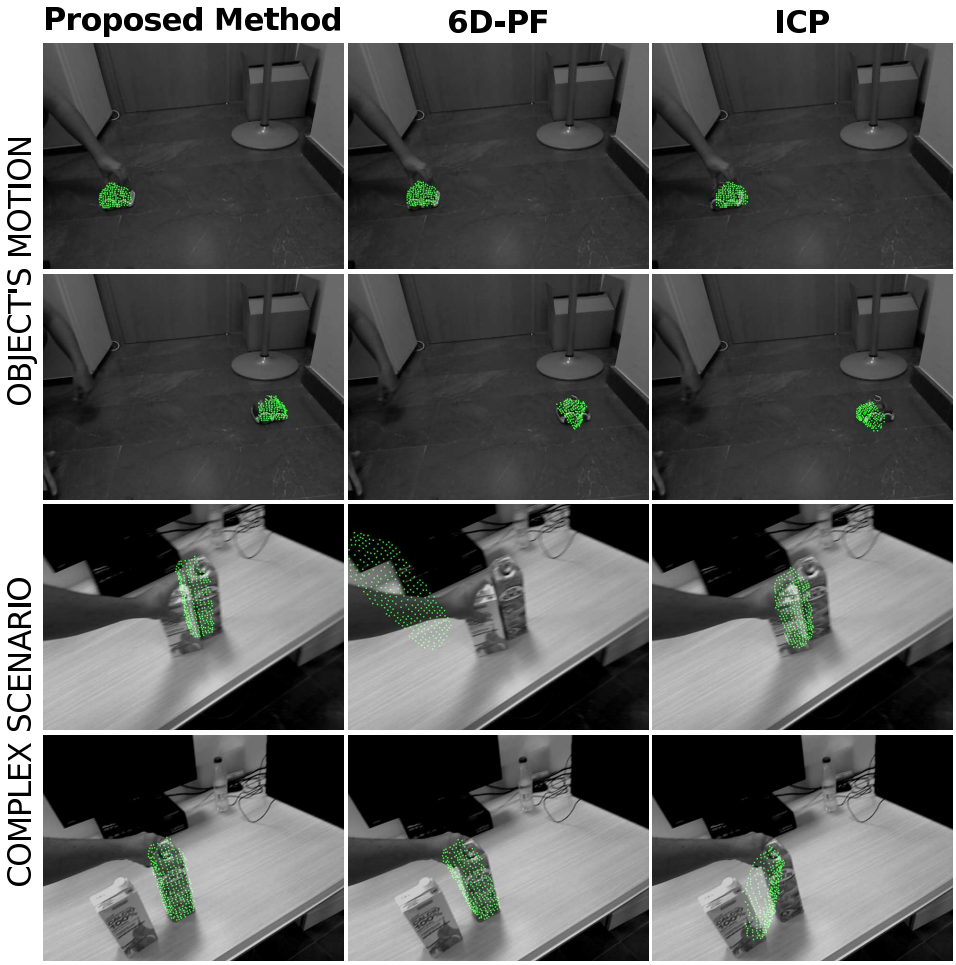


Figure 6.11: Pose estimates obtained in the more challenging scenarios. Two representative frames are shown from each sequence. The green dots represent the estimated poses reprojected on the image.

mates of each method. In the *object's motion* sequence, the object is initially occluded and then moves at high speed from the left to the right side of the floor. The proposed method is unaffected by this high-speed motion of the object or its occlusion. Even though the ICP and 6D-PF methods do not suffer much from the initial partial occlusion, they have more problems with the object's fast motion. The second scene presents a complex scenario with multiple moving objects, occlusions between the objects, and an independently moving camera. All methods are affected by the relative movement between the camera and the object. Based on a visual inspection of the model point cloud reprojected in the image according to the estimated pose, our method is more accurate than the other methods (see Fig. 6.11).

6.5.4 Camera Motion Comparison

The proposed system can be used in various situations. The *camera motion* (Fig. 6.8I) sequences present different scenarios where the camera rather than the object pose is estimated with respect to its environment. The furniture of the scene now represents the tracked object and its associated plane is the floor. Figure 6.12 contains a number of frames of multiple sequences where the camera is moving in the scene. In these scenarios, the number of tracked object points (scene elements) needs to be increased in order to avoid the loss of shape details. With 3,500 tracked object points and 20 iterations, the ICP method achieves 4 Hz on the hardware mentioned above. For fair comparison, we increase the number of particles to 800 for the proposed method in order to match the 4 Hz frame rate.

In this scenario, for simplicity's sake, we do not show the results related to the 6D-PF method. Base on the experiments carried out, the method has a lower performance than the proposed system. This is due to the use of less particles (400 particles in order to match the 4 Hz frame rate) and for not applying any domain reduction technique. We also test the 6D-PF method with 800 particles, as in the proposed method, and the results also showed a lower performance than the proposed method.

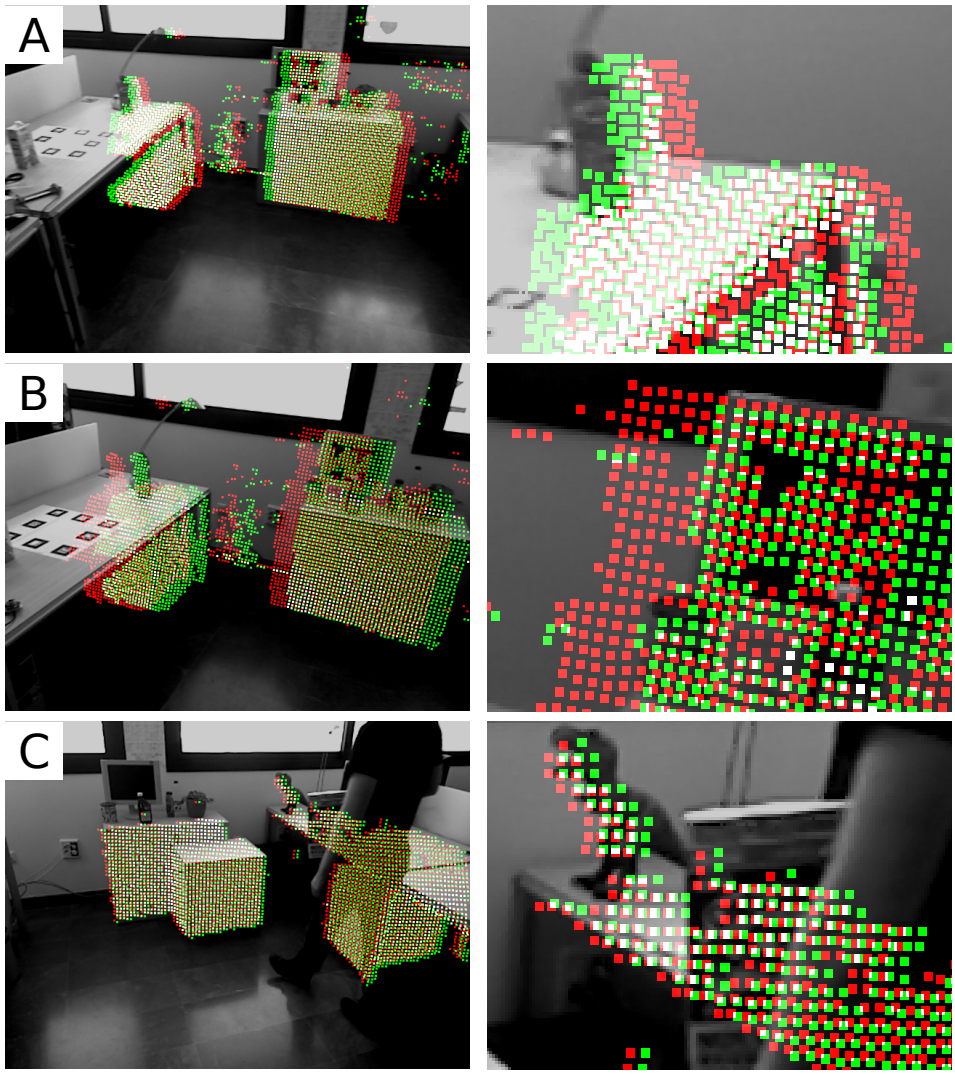


Figure 6.12: Illustrative frames of the *camera motion* sequences. The complete frames are shown in the left column and zoomed-in regions are shown in the right column: (A) high-speed camera motion, (B) camera translation, (C) scene occlusion. The red dots correspond to ICP's pose estimates and the green dots to the proposed method's estimates. The image areas where the reprojection of both methods overlaps present the combination of the two colors (yellow).

Figure 6.12 shows some representative frames of the *camera motion*-sequences. The reprojection of the estimated pose is again shown in the frames. The red dots correspond to the ICP method and the green dots to the proposed method. The image areas where the reprojections of both methods match present the combination of the two colors (yellow). This representation facilitates the comparison of both estimated poses. Both methods perform very well on the occluded scene, but experience difficulties in the presence of high-speed camera motion. The proposed method however yields more accurate results than ICP (see Fig. 6.12).

6.6 Discussion

As shown in the experiments, the proposed method achieves high accuracy in real-time in all the proposed scenarios whereas the alternative methods suffer in certain situations.

The ICP algorithm estimates the current frame's pose by initializing the object at its previous frame location, and iteratively matching it to the current frame's scene. This only works with small inter-frame displacements. ICP is also more likely to get stuck in a local minimum since only one pose estimate is maintained. The particle filtering used in the proposed and 6D-PF methods on the other hand, maintains a number of different pose hypotheses. This allows these methods to recover from failures. For these reasons, ICP is more sensitive to occlusions, cluttered scenarios, high-speed camera motion, and almost never recovers from large errors.

Unlike the proposed method, the 6D-PF method does not apply any domain reduction technique that allows concentrating the processing on a region-of-interest. This forces the method to process all the scene data for every particle. The computation time for each particle is therefore higher than in the proposed method, and, as a result, a much smaller number of particles can be processed in real-time. Furthermore, since 6D-PF operates in a six dimensional state space, a much higher number of particles are required to obtain similar accuracy as the proposed method. Together,

this higher number of particles and the increased execution time for each particle render the 6D-PF not suitable for real-world scenarios involving rapid motion, for which 30 Hz tracking is required.

One limitation of the proposed and alternative methods results from the shape of the objects. The three algorithms are unable to correctly estimate the pose of an object if a rotation of the object does not change the perceived depth. This occurs with cylindrical or spherical objects, such as cans or balls. This problem can be solved by including object appearance information (color, texture, etc.) in the pose estimation process.

In the *camera motion*-sequences the performance of the ICP method was much better, and closer to that of the proposed method, than in the object tracking sequences. This is most likely due to the much larger number of object points and the wider field-of-view that they cover. Note that this large number of points also drastically increased the execution time of both methods. The accuracy of both methods was very high even with occlusion since the occluded part was relatively small compared to the object tracking sequences. Nevertheless, the proposed method still outperformed ICP on frames with high-speed camera motion.

6.7 Conclusions

The presented chapter extensively described one of the main contributions of this thesis. We have presented a novel approach for object tracking that combines plane tracking and particle filtering in real-time. The method assumes that the object motion occurs on top of a plane (as is the case in many movements, such as cars, etc.). This greatly simplifies the problem and allows for reducing the pose parameters from six to three with respect to the reference plane. The estimation of the plane and the estimation of the object on the plane are separated, which allows for higher efficiency when tracking the object on the plane. Since the method does not rely on object appearance or on certain shape primitives, a wide variety of objects are supported.

We have demonstrated the algorithm in different challenging situations, such as high-speed motion, occlusion, cluttered scenarios and camera pose estimation. Since there is no suitable real-world benchmark dataset available with scene depth information and ground-truth, we have developed such a dataset and ground-truth data specifically for this problem. Using this benchmark we have shown that the proposed method outperforms state-of-the-art methods.

Chapter 7

Conclusions and Future Work

This dissertation has presented our scientific contribution to the area of computer vision. In particular, we focused on object pose estimation tasks.

In this final chapter we introduce a general overview of the contributions achieved during the course of this thesis. We also present a proposal for future work. Following, we include a list of publications and related works derived from this dissertation. Finally, a summary of this thesis's main contributions is exposed.

7.1 General Overview

The detection and tracking of real-world objects based on visual information is an important task for several domains. An improvement in this field is relevant in many research areas. With this motivation, the European project TOMSY creates a collaborative environment and allowed this thesis's development.

The first stage of this dissertation had the goal of developing a detection system that achieves similar performance in comparison with current detection systems. The implementation of the latest algorithms, methods and techniques contributed to a deep understanding of the computer vision field. Furthermore, this first stage led to the implementation and design of a generic architecture, which involved the establishment of an experimentation environment for further research.

The generic architecture describes the required steps for a computer vision pose estimation system. The same architecture's modules can be configured for different purposes, however, their configurations have to correspond to a global system pose estimation class.

There are two main pose estimation classes that have been addressed along this dissertation. The basic different between classes lies in the usage of temporal information. The first class refers to detection systems where no temporal information is used by the algorithms or the type of used data. A second class describes tracking methods that rely their estimations in temporal knowledge. Data, methods and algorithms that configured the pose estimation architecture have to fulfill the temporal restriction defined by the system temporal orientation (detector or tracking system).

The adopted architecture is defined by five steps that have to be configured:

- **Sensors:** Type of sensors that capture the scene's data. In this dissertation, we use standard cameras (RGB image) and depth sensors (RGB image and depth data).

- **Models:** Pose estimation task require a model of the tracked object. In this work, we have developed different techniques for object modeling. Thereby, we use different model representations with different complexity levels. Starting with manual representation models and continue with automatic representations of complex shapes and textures. We also explore off-line and on-line modeling processes.

- **Features Extraction:** The extraction of features represents the transformation of the scene's data into meaningful information. Distinctive feature's types were used along this work. Starting with appearance information based on SIFT features, we have extended the scene understanding to data segmentation and classification. This approach classifies the information into meaningful parts that represent planes and objects in the scene.

- **Pose Estimation:** Pose estimation methods directly work with scene's abstract information. Models and features are used by defined pose estimation process for object's state computation. Different methods were implemented according to the features and temporal restrictions. Methods such as a novel method for articulated object computation or a novel improved PF were implemented, described and compared in this thesis.

- **Pose parameters:** Finally, the pose parameters are the last part of the architecture that has to be defined. This configuration is related with the type of objects that we are tracking. Therefore, the object's state τ is adapted to fulfill the system requirements. In this work, we define a translation (t) and rotation (ω) vector for rigid object representation. Moreover, we extend the representation with an internal parameter chain vector (θ) that defines articulate object's configurations.

Therefore, the contribution of this work can be classified in two main goals:

- **Articulated object extension:** Based on the implemented state-of-the-art system for rigid object pose detection, we improve the defined system to articulated object pose detection. This achievement expands the type of detectable object to articulated elements. We create synthetic scenarios that simulate occlusion and noise situations and where the exact error measurement is feasible. The synthetic

sequences and the error measurement function, allow the comparison of the proposed method with the state-of-the-art methods. The proposed system was also tested in challenging real-world scenarios. The final results show that the proposed system outperforms the state-of-the-art methods and aims remarkable results in real-world scenes.

- **Improved pose estimation system:** The second main achievement describes the enhancements perform for the rigid pose estimation system. One of these enhancements refers to simplifications in the tracked object modeling process. A reduction of the information contained by the object's model is achieved. At the beginning of this work, we focused on the used of complex appearance model. This type of representation required an off-line exploration stage to generate the geometry and appearance information model. In this thesis, we define a tracking system that uses basic geometric information in its estimation process. The simplification on the system's models enables a modification on the exploration stage from an off-line model generation to an on-line model acquisition. An additional improvement incorporated to the pose estimation system, refers to including environmental information to the pose tracking methods. Information such as surrounding scene's planes decomposition, significantly improves the pose estimation process results. This additional information involves an adaptation of the pose estimation methods to the new data type. Both enhancement were implemented in a pose tracking system and compared with state-of-the-art methods. Accurate comparison was doable thanks to an augmented reality tool (ARToolKit). A benchmark with many different scenarios was developed. The proposed method performs an accurate and robust pose estimation that outperforms the alternative methods. The developed pose tracking framework is extended to diverse tracking situations such as camera pose estimation tasks.

7.2 Future Work

The generic architecture can be extended in many diverse ways. The natural extension from rigid objects to articulated objects could be continued to deformable objects. The deformable object representation could be defined as an articulated object that is composed by a large number of articulated parts, which are connected and constrained between each other.

Following the steps presented in the combined rigid pose estimation system, we could develop a system that combines the proposed articulated detection method with articulated tracking methods. As in the rigid approach, this combination will lead to an articulated combined system that groups the advantages from both method's classes.

The proposed improvement for the rigid object tracking could be applied to the articulated object tracking, where the articulated object's models are simplified and environmental information added to the pose estimation process.

Independent to the object pose estimation task, an alternative task was introduced in the last chapter regarding camera pose estimation. This topic refers to the compute vision ego-motion task. Ego-motion is responsible for computing the camera motion (direction vector) in a frame sequence. Combining and comparing ego-motion methods with the camera pose estimation could improves the understanding of the camera trace and therefore, enhancing the computer vision tasks that based their computation in the direction vector, such as independent moving objects (IMOs) detection.

7.3 Publications

The published (or submitted) works related to this research are the followings:

- M. Vanegas, **L. Rubio**, M. Tomasi, J. Diaz and E. Ros, "On-chip ego-motion estimation based on optical flow". 7th International Sym-

posium on Applied Reconfigurable Computing (**ARC 2011**). Belfast (United Kingdom).

- **L. Rubio** "Interfaces Gestuales", Book's Chapter "Perifericos Avanzados" A. Prieto, GARCETA GRUPO EDITORIAL, **2012**.

- **L. Rubio** and R. B. Rusu, "Point Cloud Library Tracking". IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop (**IROS 2012**). Vilamoura, Algarve (Portugal).

- K. Pauwels, **L. Rubio**, J. Diaz and E.Ros. "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues". Computer Vision Pattern Recognition (**CVPR 2013**). Portland (United States).

- E.J. Fernandez-Sanchez, **L. Rubio**, J. Diaz and E. Ros. "Background subtraction model based on color and depth cues". Journal of Machine Vision and Applications (**JMVA 2013**).

- **L. Rubio**, K. Pauwels, J. Diaz, E. Ros and R. Rusu. "Hybrid search of Particle Filter and Plane Tracking for Rigid Object Pose Estimation in Real-Time". **SUBMITTED**.

- **L. Rubio**, K. Pauwels, J. Diaz and E. Ros. "Articulated Object Pose Detection from Depth and Appearance by Decoupling Rigid and Non-rigid Pose Components". **SUBMITTED** to Image and Vision Computing (**IVC**).

- K. Pauwels, **L. Rubio** and E. Ros."Real-time Pose Estimation and Tracking of Hundreds of Objects". **SUBMITTED** to International Journal of Robotics Research (**IJRR**).

- K. Pauwels, **L. Rubio** and E. Ros."Real-time Model-based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints". **SUBMITTED** to Computer Vision Pattern Recognition (**CVPR 2014**).

7.4 Main Contributions

We include here a summary of the main contributions achieved in this dissertation.

- Benchmark sequences (available at *http://www.leonardorubio.com/*)
  - A *synthetic benchmark* for rigid object pose estimation was developed. The rendering engine allows to know the ground-truth object's poses. The benchmark contains 18 sequences with 6 types of object in the presence of noise and occlusion. The created sequences were rendered from two camera positions that allow disparity computation. Along with the frame data, ground-truth depth is provided.
  - A *synthetic benchmark* for articulated object pose estimation was developed. The ground-truth of the articulated object pose is known by the rendering engine. The benchmark contains 8 sequences with different levels of occlusion and noise. Along with the frame data, ground-truth depth is provided.
  - A *real-world benchmark* for rigid object pose estimation was developed. Due to an external augmented reality tool (ARToolKit), that provides accurate pose estimation, an approximated ground-truth pose is computed. Using RGB-D cameras, the scene's depth is measured. These two mechanisms allow the creation of a real-world benchmark for rigid objects. We provide 6 sequences between 400 and 800 frames long, and with 5 different objects. Image frames and depth information is provided in the benchmark.
- Object Modeling
  - We have incorporated techniques for real-world object modeling and we have implemented methods that adapt the standard models intro pose estimation models. The adaptation implies

appearance information generation and kinematic information definition.

- Pose Estimation Systems
  - Collaboration in the integration of a novel system that combines detection and tracking methods for rigid objects was implemented. The proposed system is able to work in real-time and has a remarkable perform in real-world scenarios. This system outperforms the state-of-the-art methods.
  - A novel system for articulated object pose detection was implemented. Based on the rigid object detection system, we have implemented a novel method that outperforms the state-of-the-art methods.
  - A novel system that reduces the requirement and outperforms the state-of-the-art system in rigid object tracking was implemented. Improvements such as object's model simplification or environmental information were added to improve the rigid object pose tracking methods.

Capítulo 8

Conclusiones y Trabajo Futuro (Español)

En esta tesis se han presentado contribuciones científicas en el área de la visión artificial. En particular, nos hemos centrado en las tareas de estimación de la pose de objetos.

En este último capítulo presentamos una visión general de los aportes obtenidos a lo largo de esta tesis. También presentamos una propuesta de trabajo futuro. Después, se incluye una lista de publicaciones y trabajos derivados de esta tesis. Por último, se expone un resumen de las principales aportaciones de esta tesis doctoral.

8.1 Discusión General

La detección y el seguimiento de objetos del mundo real basándonos en información visual es una tarea importante en varios dominios. Obtener una mejora en este campo es de relevante importancia para muchas áreas de investigación. Con esta motivación, el proyecto europeo TOMSY, crea un ambiente de colaboración que ha permitido el desarrollo de esta tesis.

La primera etapa de esta tesis doctoral tenía el objetivo de desarrollar un sistema de detección de objetos que logra un rendimiento similar al de los sistemas de detección actuales. La implementación de los últimos algoritmos, métodos y técnicas contribuyó a una profunda comprensión del campo de visión artificial. Además, esta primera etapa condujo a la implementación y al diseño de una arquitectura genérica, que supuso la creación de un entorno de experimentación para el desarrollo de nuevas contribuciones.

La arquitectura genérica describe los pasos necesarios para diseñar un sistema de visión artificial de estimación de pose. Los módulos de la arquitectura pueden ser configurados con diferentes propósitos, sin embargo, su configuración de módulos debe ser consecuente entre sí y corresponde con una clase de metodología de estimación de pose determinada.

Existen dos clases principales de estimación de pose a las que se hacen referencia a lo largo de esta tesis. La diferencia básica entre las distintas clases radica en el uso de la información temporal. La primera clase se refiere a sistemas de detección donde no se hace uso de información temporal en los algoritmos o en el tipo de datos tratados. La segunda clase describe los métodos de seguimiento que basan sus estimaciones en información temporal. Datos, métodos y algoritmos que configuran la arquitectura de estimación de pose tienen que cumplir con la restricción temporal definida para el sistema (detección o seguimiento).

La arquitectura adoptada se define con la configuración de cinco bloques:

- **Sensores:** Son los tipo de sensores que capturan los datos de la escena. En esta tesis, se utilizan cámaras estándar (imagen RGB) y sensores de profundidad (imagen RGB y datos de profundidad).
- **Modelos:** La tarea de estimación de pose requiere un modelo del objeto que se está siguiendo. En este trabajo, se han desarrollado diferentes técnicas de modelado de objetos. De este modo, se utilizan diferentes representaciones de modelos con diversos niveles de

complejidad. Desde modelos de representación manual, hasta modelos generados de forma automática con formas complejas y texturas. También exploramos procesos de modelado off-line y on-line.

- **Extracción de Características:** La extracción de características representa la transformación de los datos de la escena en información significativa. Distintos tipos de características han sido utilizadas a lo largo de este trabajo. Partiendo de la información de apariencia que describen las características SIFT, hemos ampliado la comprensión de la escena con segmentación y clasificación de datos. Este método clasifica la información en partes significativas que representan planos y objetos de la escena.

- **Estimación de Pose:** Los métodos de estimación de pose trabajan directamente con la información abstracta de la escena. Modelos y características son utilizados por el proceso de estimación de pose para el computo del estado del objeto. En este trabajo se implementan diferentes métodos de acuerdo con las restricciones temporales. Métodos tales como un método novel para la estimación de objeto articulado o un filtro de partículas mejorado, se describen y se evalúan en esta tesis.

- **Parámetros de Pose:** Finalmente, los parámetros que definen una pose son la última parte de la arquitectura que tiene que ser definida. Esta configuración está relacionada con el tipo de objetos que estamos siguiendo. Por lo tanto, el estado del objeto τ se ajusta para cumplir con los requisitos del sistema. En este trabajo, definimos un vector translación (t) y rotación (ω) para la representación del estado de los objetos rígidos. Además, extendemos la representación con un vector de cadena de parámetros internos (θ) que definen configuraciones de objetos articulados.

Por lo tanto, las contribuciones de este trabajo se pueden clasificar en dos objetivos principales:

- **Extensión a objetos articulados:** Partiendo de un sistema del estado-de-la-técnica para la detección de la pose de objetos rígidos, realizamos una mejor el sistema definido para la detección de objetos articulados. Esta mejora expande el tipo de objetos detectables a elementos articulados. Creamos escenarios sintéticos que simulan oclusión y ruido, y donde la medición de error de la estimación es factible. Las secuencias sintéticas y la función de medición de error, permiten la comparación del método propuesto con los métodos del estado-de-la-técnica. El sistema propuesto también fue probado en escenarios complejos del mundo real. Los resultados finales muestran que el sistema propuesto supera a los métodos del estado-de-la-técnica y produce resultados notables en escenarios del mundo real.

- **Mejoras del sistema de estimación de pose:** El segundo logro principal se centra en las mejoras realizadas en los sistemas de estimación de pose rígida. Una de estas mejoras se refiere a la simplificación del proceso de modelado de objetos. Se consigue una reducción de la información necesaria de los modelo de los objetos. Al comienzo de este trabajo, nos centramos en modelos de apariencia compleja. Este tipo de representación necesita una etapa de exploración off-line para generar la geometría y el modelo de apariencia. En esta tesis, definimos un sistema de seguimiento que únicamente utiliza información geométrica básica para el proceso de estimación de pose. Una simplificación de los modelos utilizados en el sistema permite la modificación de la etapa de exploración donde pasamos de una generación de modelos off-line a una adquisición de modelos on-line. Adicionalmente incorporamos al sistema de estimación de pose una mejora relacionada con la inclusión de información del entorno al proceso de seguimiento. Información del entorno, como la descomposición de los planos de la escena, mejora significativamente los resultados del proceso de estimación de pose. Esta información adicional implica una adaptación de los métodos de estimación de la posición para este nuevo tipo de información. Ambas mejoras fueron implementadas en un sistema de seguimiento y fueron comparadas con

los métodos del estado-de-la-técnica. Una comparación precisa entre sistemas fue factible gracias a una herramienta de realidad aumentada (ARToolKit). Un banco de pruebas fue desarrollado donde se representan una gran variedad de escenarios. El método propuesto realiza una estimación precisa y robusta lo cual supone una mejora significativa frente a los métodos alternativos. El desarrollo planteado para el seguimiento de objetos rígidos se puede extender a diversas situaciones, como por ejemplo, en tareas de estimación del movimiento de una cámara.

8.2 Trabajo Futuro

La arquitectura genérica puede extenderse de muy diversas maneras. La extensión natural de objetos rígidos a objetos articulados podría continuar con la detección y seguimiento de objetos deformables. La representación de objeto deformable se podría definir como un objeto articulado que está compuesto por un gran número de piezas articuladas, que están conectados entre sí con un movimiento limitado por las articulaciones vecinas.

Siguiendo la metodología empleada en el sistema combinado de estimación de pose rígida, se podría desarrollar un sistema que combina el método propuesto de detección para objetos articulados con métodos de seguimiento de objetos articulados. Esta combinación dará lugar a un sistema combinado articulado que agrupe las ventajas de ambas clases de métodos.

La mejora propuesta para el seguimiento de objetos rígidos se podría aplicar en sistemas de seguimiento de objetos articulados, donde se podría realizar una simplificación de los modelos articulados e incluir información del entorno al proceso de estimación de la posición.

Independientemente del proceso de estimación de pose, hemos explorado aplicaciones alternativas donde poder emplear el sistema desarrollado. En el capítulo anterior, el sistema de seguimiento fue aplicado para el seguimiento de la estimación de pose de cámaras. Este tema está relacionado con el

cálculo de ego-motion. El ego-motion es responsable de calcular la dirección del movimiento de la cámara (vector de dirección) en una secuencia de imágenes. Una combinación y comparación de los métodos de ego-motion con la estimación de pose de la cámara podría mejora la comprensión de su movimiento y por lo tanto, mejorar las tareas de visión artificial que se basan en el vector de dirección, como por ejemplo la detección de objetos con movimiento independiente a la cámara (IMOs).

8.3 Publicaciones

Los trabajos publicados (o enviado para publicación) relacionados con esta investigación son los siguientes:

- M. Vanegas, **L. Rubio**, M. Tomasi, J. Diaz and E. Ros, "On-chip ego-motion estimation based on optical flow". 7th International Symposium on Applied Reconfigurable Computing (**ARC 2011**). Belfast (United Kingdom).

- **L. Rubio** "Interfaces Gestuales", Book's Chapter "Perifericos Avanzados" A. Prieto, GARCETA GRUPO EDITORIAL, **2012**.

- **L. Rubio** and R. B. Rusu, "Point Cloud Library Tracking". IEEE/RSJ International Conference on Intelligent Robots and Systems: Workshop (**IROS 2012**). Villamoura, Algarve (Portugal).

- K. Pauwels, **L. Rubio**, J. Diaz and E.Ros. "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues". Computer Vision Pattern Recognition (**CVPR 2013**). Portland (Estados Unidos).

- E.J. Fernandez-Sanchez, **L. Rubio**, J. Diaz and E. Ros. "Background subtraction model based on color and depth cues". Journal of Machine Vision and Applications (**JMVA 2013**).

- **L. Rubio**, K. Pauwels, J. Diaz, E. Ros and R. Rusu. "Hybrid search of Particle Filter and Plane Tracking for Rigid Object Pose Estimation in Real-Time". **ENVIADO**.

- **L. Rubio**, K. Pauwels, J. Diaz and E. Ros. "Articulated Object Pose Detection from Depth and Appearance by Decoupling Rigid and Non-rigid Pose Components". **ENVIADO** to Image and Vision Computing (**IVC**).

- K. Pauwels, **L. Rubio** and E. Ros."Real-time Pose Estimation and Tracking of Hundreds of Objects". **ENVIADO** to International Journal of Robotics Research (**IJRR**).

- K. Pauwels, **L. Rubio** and E. Ros."Real-time Model-based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints". **ENVIADO** to Computer Vision Pattern Recognition (**CVPR 2014**).

8.4 Contribuciones Principales

Incluimos aquí un resumen de las contribuciones principales logradas en esta tesis.

- Secuencias de Benchmark (disponibles en *http://www.leonardorubio.com/*)

  - Se ha desarrollado un *benchmark sintético* para la estimación de objetos rígidos. Mediante el uso de motores de renderizado podemos conocer la pose real del objeto en la escena. El *benchmark* contiene 18 secuencias con 6 tipos de objeto donde añadimos ruido y oclusiones. Las secuencias son creadas desde dos posiciones de cámara conocidas que permiten el cálculo de disparidad. Junto con la secuencia de imágenes, se proporciona la información de profundidad de la escena.

  - Se ha desarrollado un *benchmark sintético* para la estimación de objetos articulados. La pose real de los objetos articulados es conocida gracias a los motores de renderizado. El benchmark contiene 8 secuencias donde se emplean diferentes niveles de oclusión y ruido. Junto con las secuencias de imágenes, se proporciona la información de profundidad de la escena.
  - Se ha desarrollado un *benchmark* del mundo real para la estimación de objetos rígidos. Gracias a una herramienta externa de realidad aumentada (ARToolKit), podemos estimar una aproximación precisa de la pose real de los objetos en la escena. Usando cámaras RGB-D, podemos medir la profundidad de la escena. Estos dos mecanismos permiten la creación de un *benchmark* del mundo real para objetos rígidos. Proporcionamos 6 secuencias de entre 400 y 800 frames, y con 5 objetos diferentes. El *benchmark* proporciona las secuencias de frames y la información de profundidad para cada secuencia.

- Modelado de Objetos

  - Hemos incorporado técnicas de modelado de objetos del mundo real y se han implementado métodos que adaptan dichos modelos estándar para su uso en sistemas de estimación. La adaptación implica la generación de la información de apariencia y la definición de la información cinemática del modelo.

- Sistemas de Estimación de Pose

  - Colaboración en la implementación de un sistema novel que combina los métodos de detección y seguimiento de objetos rígidos. El sistema propuesto es capaz de trabajar en tiempo real, y presenta una notable mejora en escenarios del mundo real. Este sistema supera a los métodos del estado-de-la-técnica.
  - Se implementó un sistema novel para la detección los objetos articulados. Basado en el sistema de detección de objetos rígidos,

hemos implementado un método novel que supera a los métodos del estado-de-la-técnica.

– Se implementó un sistema novel que reduce los requisitos de los sistemas estándar de seguimiento de objetos y supera las prestaciones de los sistemas del estado-de-la-técnica. Se han añadido mejoras relacionadas con la simplificación de los modelos de objetos o la inclusión de información del entorno en el proceso de estimación de pose.

Bibliography

[1] Hobrough Gilbert Louis. Methods and apparatus for correlating corresponding points in two images. *US Patent*, 2964642, December 1960.

[2] Mauricio Vanegas, Leonardo Rubio, Matteo Tomasi, Javier Diaz, and Eduardo Ros. On-chip ego-motion estimation based on optical flow. In *Proceedings of the 7th international conference on Reconfigurable computing: architectures, tools and applications*, ARC'11, pages 206–217, Berlin, Heidelberg, 2011. Springer-Verlag.

[3] Sara Granados, Francisco Barranco, Sonia Mota, Javier Díaz, and Eduardo Ros. On-chip semidense representation map for dense visual features driven by attention processes. *Journal of Real-Time Image Processing*, pages 1–15, 2013.

[4] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of rigid objects. *Found. Trends. Comput. Graph. Vis.*, 1(1):1–89, January 2005.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[6] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. In Lawrence B. Wolff, Steven A. Shafer, and Glenn Healey,

editors, *Radiometry*, pages 221–244. Jones and Bartlett Publishers, Inc., USA, 1992.

[7] Peter Sturm and Steve Maybank. On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, volume 1, pages 432–437, Fort Collins, États-Unis, 1999. IEEE Computer Society.

[8] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000.

[9] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 1106–, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Barak IL FREEDMAN Binyamina IL, Alexander US SHPUNT Cambridge MA US, and Yoel IL ARIELI Jerusalem IL. Distance-Varying Illumination and Imaging Techniques for Depth Mapping. *US Patent*, 12522176(US 2010/0290698 A1), January 2008.

[11] Microsoft's Kinect. Microsoft's Kinect. http://www.xbox.com/en-us/kinect/, 2013.

[12] MoCap System. Mocap system, carnegie mellon university motion capture database. http://mocap.cs.cmu.edu/, 2003.

[13] H. Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, San Francisco, USA, October 1999.

[14] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[15] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[16] Michel Jourlin and Jean-Charles Pinoli. A model for logarithmic image processing. *Journal of Microscopy*, 149(1):21–35, 1988.

[17] Eric J. Wharton, Karen Panetta, and Sos S. Agaian. Logarithmic edge detection with applications. In *SMC*, pages 3346–3351. IEEE, 2007.

[18] W. Förstner. A Feature Based Correspondence Algorithm for Image Matching. *Int. Arch. of Photogrammetry and Remote Sensing*, 26(3):150–166, 1986.

[19] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[20] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, Ithaca, NY, USA, 1993.

[21] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[22] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[23] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[24] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[25] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[26] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[27] Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision*, 13(3):331–356, December 1994.

[28] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):774–780, August 1999.

[29] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(8):930–943, August 2003.

[30] Shang-Ching Chou and Xiao-Shan Gao. Ritt-wu's decomposition algorithm and geometry theorem proving. In Mark E. Stickel, editor, *CADE*, volume 449 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 1990.

[31] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[32] Daniel F. Dementhon and Larry S. Davis. Model-based object pose in 25 lines of code. *Int. J. Comput. Vision*, 15(1-2):123–141, June 1995.

[33] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, November 1998.

[34] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000.

[35] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50:174–188, 2002.

[36] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):932–946, July 2002.

[37] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)*, 2010.

[38] Changhyun Choi and Henrik I Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. J. Rob. Res.*, 31(4):498–519, April 2012.

[39] Victor Prisacariu and Ian Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision*, pages 1–20, 2012. 10.1007/s11263-011-0514-3.

[40] Thomas Brox, Bodo Rosenhahn, Juergen Gall, and Daniel Cremers. Combined region and motion-based 3d tracking of rigid and articulated objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3):402–415, March 2010.

[41] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011*

10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[42] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

[43] Blender Foundation. Blender project: 3D computer graphics software product. http://www.blender.org/, 1995.

[44] Leslie Predy, Alexander Rice-Khouri, Greg Fowler, Anna Romanovska, and Hans-Frederick Brown. 123d sculpt: designing a mobile 3d modeling application for novice users. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 845–848, New York, NY, USA, 2012. ACM.

[45] Changchang Wu. SiftGPU: A GPU implementation of scale invariant features transform (SIFT). http://cs.unc.edu/ ccwu/siftgpu/, 2007.

[46] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[47] Karl Pauwels, Leonardo Rubio, Javier Díaz, and Eduardo Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *Conf. Computer Vision and Pattern Recognition*, CVPR '13, 2013.

[48] Vincent Lepetit and Pascal Fua. Monocular model-based 3D tracking of rigid objects. *Found. Trends. Comput. Graph. Vis.*, 1(1):1–89, January 2005.

[49] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, October 2007.

[50] Koike H, Sato Y, and Kobayashi Y. Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system. *ACM Trans. Comput.-Hum. Interact.*, 8(4):307–322, December 2001.

[51] Ben Daubney, David P. Gibson, and Neill W. Campbell. Real-time pose estimation of articulated objects using low-level motion. In *Conf. Computer Vision and Pattern Recognition*, CVPR '08, 2008.

[52] Ben Daubney, David P. Gibson, and Neill W. Campbell. Estimating pose of articulated objects using low-level motion. *Computer Vision and Image Understanding*, 116(3):330–346, 2012.

[53] A. Woerner. H. Koehler. Motion-based feature tracking for articulated motion analysis. In *International Conference on Multimodal Interfaces, Workshop on Multimodal Interactions Analysis of Users a Controlled Environment.*, ICMI '08, 2008.

[54] Tom Drummond and Roberto Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proceedings of the International Conference on Computer Vision*, ICCV '01, pages 315–320, 2001.

[55] Andrew I. Comport, Eric Marchand, and François Chaumette. Object-based visual 3D tracking of articulated objects via kinematic sets. In *Conf. Computer Vision and Pattern Recognition Workshop*, CVPRW '04, pages 2–, 2004.

[56] Andrew I. Comport, Eric Marchand, and François Chaumette. Kinematic sets for real-time robust articulated object tracking. *Image Vision Computing*, 25(3):374–391, March 2007.

[57] Quentin Delamarre and Olivier Faugeras. 3D articulated models and multi-view tracking with silhouettes. In *Proceedings of the International Conference on Computer Vision*, ICCV '99, pages 716–, 1999.

[58] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal Computer Vision*, 61(1):55–79, January 2005.

[59] M. Dhome, A. Yassine, and J. M. Lavest. Determination of the pose of an articulated object from a single perspective view. In *British Machine Vision Conference*, BMVC '93, pages 10.1–10.10, 1993.

[60] Alexa Hauck, Stefan Lanser, and Christoph Zierl. Hierarchical recognition of articulated objects from single perspective views. In *Conf. Computer Vision and Pattern Recognition*, CVPR '97, pages 870–883, 1997.

[61] Baihua Li, Qinggang Meng, and Horst Holstein. Reconstruction of segmentally articulated structure in freeform movement with low density feature points. *Image Vision Computing*, 22(10):749–759, 2004.

[62] Baihua Li, Qinggang Meng, and Horst Holstein. Articulated motion reconstruction from feature points. *Conf. Computer Vision Pattern Recognition*, 41(1):418–431, 2008.

[63] S. Pellegrini, K. Schindler, and D. Nardi. A generalization of the icp algorithm for articulated bodies. In M. Everingham and C. Needham, editors, *British Machine Vision Conference*, BMVC '08, 2008.

[64] D. Wolff. *OpenGL 4.0 Shading Language Cookbook: Over 60 Highly Focused, Practical Recipes to Maximize Your Use of the OpenGL Shading Language*. Packt Publishing, 2011.

[65] Chuong V. Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3D reconstruction and tracking. In *3DIMPVT*, pages 524–530, 2012.

[66] Raúl Cabido, Antonio S. Montemayor, and Juan José Pantrigo. High performance memetic algorithm particle filter for multiple object tracking on modern gpus. *Soft Comput.*, 16(2):217–230, 2012.

[67] Zsolt L. Husz, Andrew M. Wallace, and Patrick R. Green. Tracking with a hierarchical partitioned particle filter and movement modelling. *Trans. Sys. Man Cyber. Part B*, 41(6):1571–1584, December 2011.

[68] J. Schmidt, B. Kwolek, and J. Fritsch. Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images. In *Proc. of Automatic Face and Gesture Recognition*, pages 567–572, Southampton, UK, April 2006. IEEE.

[69] F Pomerleau, S Magnenat, F Colas, M Liu, and R Siegwart. Tracking a depth camera: Parameter exploration for fast icp. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[70] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[71] Limin Shang, Piotr Jasiobedzki, and Michael A. Greenspan. Model-based tracking by classification in a tiny discrete pose space. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):976–989, 2007.

[72] J. Deutscher, A. J. Davison, and I. D. Reid. Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Kauai*. IEEE Computer Society Press, December 2001.

[73] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *Best Practice in 3D Perception and Modeling for Mobile Manipulation (in conjunction with ICRA 2010)*, 2010.

[74] J. Deutscher, A. Blake, and I. D. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2126–2133. IEEE Computer Society Press, 2000.

[75] David Hogg. Model-based vision: a program to see a walking person. *Image Vision Comput.*, 1(1):5–20, 1983.

[76] Sourabh A. Niyogi and Edward H. Adelson. Analyzing and recognizing walking figures in xyt. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 469–474. IEEE Computer Society Press, 1994.

[77] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pages 764–, Washington, DC, USA, 1995. IEEE Computer Society.

[78] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 8–, Washington, DC, USA, 1998. IEEE Computer Society.

[79] Jesús Martínez del Rincón, Dimitrios Makris, Carlos Orrite-Uruñuela, and Jean-Christophe Nebel. Tracking human position and lower body parts using kalman and particle filters constrained by human biomechanics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 41(1):26–37, 2011.

[80] Mun Wai Lee, Isaac Cohen, and Soon Ki Jung. Particle filter with analytical inference for human body tracking. In *Proceedings of the Workshop on Motion and Video Computing*, MOTION '02, pages 159–, Washington, DC, USA, 2002. IEEE Computer Society.

[81] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[82] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Trans. Sig. Proc.*, 50(2):174–188, February 2002.

[83] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[84] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-Time Plane Segmentation using RGB-D Cameras. In *Proceedings of the 15th RoboCup International Symposium*, volume 7416 of *Lecture Notes in Computer Science*, pages 307–317, Istanbul, Turkey, July 2011. Springer.

[85] Michael A. Greenspan and Mike Yurick. Approximate k-d tree search for efficient icp. In *3DIM*, pages 442–448. IEEE Computer Society, 2003.

[86] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.