Universitat
Jaume•I

Departament d’Enginyeria i Ciències dels Computadors

Universitat Jaume I

Visual determination, tracking and execution of 2D grasps using a behavior-inspired approach

Ph. D. Thesis

Presented by:

Gabriel Recatalá Ballester

Supervised by:

Pedro José Sanz Valero
Ángel Pascual del Pobil y Ferré
Enric Cervera Mateu

Castelló, 2003

UNIVERSITAT
JAUME·I

Departament d'Enginyeria i Ciències dels Computadors

Universitat Jaume I

Determinación, seguimiento y ejecución visual de agarres 2D usando una aproximación inspirada en comportamientos

Tesis Doctoral

Presentada por:

Gabriel Recatalá Ballester

Supervisada por:

Pedro José Sanz Valero
Ángel Pascual del Pobil y Ferré
Enric Cervera Mateu

Castelló, 2003

To my family,
for the encouragement and support
I have received from them.

Abstract

This thesis focuses on the definition of a task for the determination, tracking and execution of a grasp on an unknown object. In particular, it is considered the case in which the object is ideally planar and the grasp has to be executed with a two-fingered, parallel-jaw gripper using vision as the source of sensor data. Each step in this task is analyzed separately, considering several options in some cases.

For the specification of this task, an architecture is defined that is based on three basic components –virtual sensors, filters, and actuators–, which can be connected to define a control loop. The proposed architecture is highly modular, making easier the development of the control task, and can be used as a framework for the development of other control tasks of a similar degree of complexity.

Within this task, several options are analyzed for the discrimination of the object of interest from the other elements in the observed scene. Procedures for finding the location of stable grasps on the shape of the object and selecting a single one are also provided and several possibilities for the description of the grasp are described. The grasp points corresponding to the selected grasp are directly used as control features for a visual servoing control law. This has led to the development of algorithms for allowing the tracking of the grasp points along a sequence of images and an analysis of the restrictions imposed by each particular tracking algorithm on the control law. Unlike in other works, a procedure is proposed for the automatic computation of the target value of the grasp points to be used by the control law. In addition, a simple algorithm is provided to compute, in an off-line step, some of the reference values required by this procedure. Results show that the tracked grasp points can be successfully used to guide a robot arm towards an unknown object in order to grasp it, even when noisy data are extracted from the images provided by the vision system.

Some of the main contributions of this thesis include: (1) the use of a modular approach to the specification of a control task that provides a basic framework for supporting the concept of behavior; (2) the analysis of several strategies for obtaining a compact representation of the contour of an object; (3) the development of a method for the evaluation and search of a grasp on a planar object for a two-fingered gripper; (4) the specification of different representations of a grasp and the analysis of their use for tracking the grasp between different views of an object; (5) the specification of algorithms for the tracking of a grasp along the views of an object obtained from a sequence of single images and a sequence of stereo images; (6) the definition of parametrized models of the target position of the grasp points and of the feasibility of this target grasp, and of an off-line procedure for the computation of some of the reference values required by this model; and (7) the definition and analysis of a visual servoing control scheme to guide the gripper of a robot arm towards an unknown

object using the grasp points computed for that object as control features.

Resumen

Motivación

Esta tesis aborda el problema de la manipulación de objetos mediante un robot. Un requisito importante en muchas áreas de la robótica, como es el caso de la robótica de servicios, es la capacidad para realizar estos pasos sin depender de un modelo predefinido del objeto –con independencia de que previamente el sistema haya hecho uso o no de un modelo para reconocer dicho objeto.

En general, esta manipulación implica la ejecución de una secuencia de pasos. Entre estos pasos, se encontraría la selección del objeto a agarrar, la determinación de unos puntos de agarre sobre ese objeto, la aproximación de la garra del robot hacia los puntos de agarre y el agarre propiamente dicho del objeto. Ahora bien, en la mayoría de los trabajos sobre manipulación de objetos con robots, estos pasos han sido abordados de forma aislada. Esta tesis aborda la ejecución de los pasos relacionados con la aproximación de la garra al objeto, y propone la definición de una arquitectura para la integración de los mismos en un sistema de control.

Objetivos

El principal objetivo de esta tesis es la definición de una tarea que se encargue de la determinación, seguimiento y ejecución del agarre de un objeto sin requerir para ello un modelo predefinido de dicho objeto. Este objetivo, incluye, entre otros, los siguientes objetivos parciales:

- Definición de un mecanismo para la selección de un objeto de la escena.
- Definición de un mecanismo para la búsqueda de puntos de agarre en un objeto y el seguimiento de los mismos a lo largo de una secuencia de imágenes.
- Definición de un esquema de control para el guiado del robot hacia los puntos de agarre seleccionados sobre el objeto.
- Definición de una arquitectura para la integración de todos los pasos relacionados con la tarea de aproximar la garra del robot hacia el objeto.

Metodología

La tarea de control considerada en esta tesis ha sido tratada como un problema de posicionamiento entre el robot y el objeto. Para abordar este problema, se ha definido un sistema de control visual encargado de mover el robot hacia el objeto utilizando para ello la información extraída de las imágenes proporcionadas por un sistema de visión. Los diferentes pasos a realizar dentro de este sistema de control, desde la extracción de la información visual necesaria hasta el cálculo de las órdenes de movimiento a enviar al robot han sido identificados y estudiados separadamente, considerando diferentes opciones para el desarrollo de algunos de ellos.

Para la integración de los pasos anteriores, se ha propuesto una arquitectura basada en tres componentes básicos –sensores, filtros y actuadores virtuales– los cuales serían conectados entre sí para construir el bucle de control. Este enfoque permite la realización de un diseño modular de la tarea de control, facilitando el desarrollo de una implementación distribuida de la misma. En esta tesis, se propone dicha arquitectura como marco general para el desarrollo de otras tareas de control.

Aportaciones

Entre las principales contribuciones de esta tesis, cabe destacar:

- El uso de un enfoque modular para la especificación de una tarea de control, permitiendo introducir el concepto de comportamiento dentro de dicha especificación.

- El análisis de diversas estrategias para obtener una representación compacta del contorno de un objeto.

- El desarrollo de un método para la evaluación y búsqueda de agarres para una garra de dos dedos plano-paralelos sobre un objeto idealmente plano.

- La especificación de diferentes representaciones de un agarre y el análisis de su aplicabilidad para el seguimiento de dicho agarre entre diferentes vistas de un objeto.

- La especificación de algoritmos para el seguimiento de un agarre a lo largo de las vistas de un objeto obtenidas a partir tanto de una secuencia de imágenes individuales como de una secuencia de imágenes estéreo.

- La definición de modelos parametrizados de la posición final de los puntos de agarre y de la viabilidad de dicho agarre. Se define también un procedimiento, para ser ejecutado como un paso previo, para el cálculo de algunos valores de referencia requeridos por estos modelos.

- La definición y el análisis de un sistema de control visual para guiar el brazo del robot hacia el objeto utilizando los puntos de agarre como datos de entrada de dicho sistema.

Conclusiones y trabajo futuro

Los resultados obtenidos con la aplicación del sistema de control demuestran la validez del seguimiento de los puntos de agarre para proporcionar datos de entrada a dicho sistema, incluso en el caso de que los datos extraídos de las imágenes proporcionadas por el sistema de visión estén bastante afectados por ruido y los valores proporcionados como parámetros a algunos componentes del sistema de control sean aproximaciones poco precisas. Por otra parte, el esquema modular de la tarea de control, establecido por la arquitectura propuesta en esta tesis ha permitido distribuir la complejidad de dicha tarea de control entre sus módulos componentes. Esto ha facilitado no sólo el análisis y desarrollo individual de cada módulo, sino también la prueba en algunos módulos de diferentes algoritmos sin alterar por ello el diseño global de la tarea de control.

Respecto al trabajo futuro, en esta tesis se propone las siguientes líneas:

- En relación con la arquitectura de control:
  - Especificación de los mecanismos necesarios para permitir la ejecución simultánea de varios comportamientos.
  - Especificación de los mecanismos necesarios para la secuenciación de comportamientos o de bloques de comportamientos, permitiendo al sistema encontrarse en un estado u otro en función de los comportamientos activos en cada momento.
  - Complementar el conjunto de componentes básicos de la arquitectura con la especificación de un componente correspondiente a un sensor activo, el cual, a diferencia de los sensores actualmente definidos, completamente pasivos, podría modificar su comportamiento en función de la información proporcionada por otros componentes.
- En relación con la selección del objeto a agarrar:
  - Estudio de mecanismos que permitan una separación robusta del objeto a agarrar del resto de la escena.
- En relación con la evaluación y búsqueda de agarres:
  - Inclusión de mecanismos de aprendizaje para permitir al sistema seleccionar los criterios más adecuados para la evaluación del agarre.
  - Estudio de otros procedimientos para la búsqueda de puntos de agarre.
- En relación con el seguimiento de los puntos de agarre:
  - Estudio de mecanismos para hacer el seguimiento más robusto respecto al ruido en la información visual.
  - Estudio de otros mecanismos de seguimiento que permitan relajar algunas de las restricciones impuestas sobre los grados de libertad controlables por sistema de control.
- En relación con la ejecución del agarre:

- Complementar la información visual con información proporcionada por otros sensores, con el fin de permitir una mayor precisión en el acercamiento de la garra del robot al objeto.

Acknowledgments

This thesis has been developed at the Robotic Intelligence Laboratory of the University Jaume I. First of all, I would like to express my greatest gratitude towards my advisors, Pedro J. Sanz, Ángel Pascual del Pobil, and Enric Cervera, for their guidance and support during the development of this thesis. They provided extremely valuable help in so many difficulties that arose during this period.

I would also like to thank the members of this laboratory, current and former, for their warm welcome and their help and encouragement from the very beginning. I am specially grateful to Antonio Morales, with whom I have shared many hours of work in the early stages of our PhDs. An important part of the work on grasp search and evaluation presented in this thesis is the result of earlier cooperation with him. In addition, the proposed specifications of task and behavior are inspired on wider-scope discussions from this period regarding the design of a behavior-based architecture to be used as a framework for our work in the laboratory. I would also like to specially thank Raül Marín, for his valuable advices during the different stages of my thesis. To Antonio, Raül and Eris, thanks for their friendship.

During these years at the university I have had the privilege to carry on part of my research in cooperation with people from other universities. Among them, I would like to thank Nikos Papanikolopoulos, from the *University of Minnesota* (Minneapolis, USA), for providing me the invitation for my first stay in a university of a foreign country. It was during this stay when I had my first contact with visual servoing. There I also received help from other members of the laboratory, such as Richard Voyles and Maria Gini. I would like to thank Rahul, Sarbjit, Rashmi, and Max for smoothing my first contact with such a different country, and, specially, for their unforgettable friendship.

I am also thankful to Prof. Georg Färber, from the *Technische Universität München* (Munich, Germany), for letting me work in his laboratory. Thanks also to Alexa Hauck and Michael Sorg, who were my first contacts there. The work on grasp tracking presented in this thesis was started during my stays at this university. Hans Oswald and Michael Sorg let me use their software for object tracking based on active contours to develop the first version of the grasp tracking algorithm. Part of my work on 3D grasping, not included in this thesis, was also started my stay at this university. As part of this stay, I had the opportunity to be the co-advisor of the projects developed by some students, which allowed me to test some ideas that I had not time during that period to fully develop by myself. In particular, using Sonja Glas worked on a procedure for the automatic initialization of a B-spline using the proposed strategy –described in section 4.4– for the selection of points on a reference contour based on the analysis of its curvature. And Jan Leupold developed a color-segmentation filter. Thanks also to other friends that I met there, such as Georg, Natalie, Sigy, Annette, Robert and Stefan.

The visual servoing work described in this thesis was actually started during my stay at the *Université Blaise Pascal* of Clermont-Ferrand (France). I am specially thankful to Philippe Martinet and Youcef Mezouar for the time they devoted to introduce me to this field and their help in the development of the control law used in this thesis. Youcef made valuable suggestions regarding the use of the homography for tracking the grasp points, which were applied to the procedures described in section 5.3.3. I would also like to thank people like Alain, Guillaume, Sandra, Mourat, Lucie and many others for making my stay in Clermont-Ferrand such an enjoyable experience.

The work on polygonal approximations using genetic algorithms described in section 4.3 was actually supervised by José Manuel Iñesta during his last period in our laboratory, as well as for some time more when he moved to the *University of Alicante*, where I visited him a couple of times. In the last stages of this work, I counted with the cooperation of V. Javier Traver, who made highly valuable suggestions and helped to improve the initial results.

For the implementation of the whole control task described in this thesis, I used components developed by several other people, some of them working on projects from which I was advisor. Raffaella Carloni, from the *University of Bologna* (Italy), worked on an initial implementation of the virtual sensor corresponding to the acquisition system during her stay in the Robotic Intelligence Laboratory. She worked on the implementation of the strategy proposed in section 5.3.3 for the selection of the correspondences to be used to compute the homography. I am deeply grateful to her for working on this modules and to Claudio Melchiorri, her advisor at the *University of Bologna*, for granting her permission to staying in our laboratory. Antonio Morales provided the initial implementations of the binarization and the contour extraction filters, as well as a template for the implementation of other filters. Marcos Alcántara and José Gómez introduced some improvements in the binarization filter. Hans Oswald provided the software for object tracking with B-splines, which was used to test the grasp tracking based on a grasp description with respect to these curves. Sonja Glas provided not only the implementation of the automatic B-spline setup, but also implemented some curvature-analysis procedures that were used as part of the homography-based grasp tracking. Mario Prats provided me a first implementation of the software to interface the robot arm, which I adapted to make it possible to use it as a virtual actuator with the interface described in this thesis. Finally, Juan Pérez developed some of the data viewers required to view the data produced by the behavior described in this thesis.

People from other laboratories in this university have also provided highly valuable help. Among others, thanks to Lledó Museros and Tere Escrig, Raul Montoliu, Maite Lozano and J. Miguel Sanchiz. I am specially thankful to V. Javier Traver, who provided extremely valuable help in so many issues. Thanks in general to the Department of Engineering and Computer Science of the University Jaume I for providing me framework for carrying on my research and teaching work, and to the national and international institutions that have provide the funds for making this research possible.

Thanks also to all the friends who have made my last years at the university so enjoyable. Among those already mentioned, thanks to Pilar, Ismael, Gustavo, Alex, Miguel Ángel, Ferran, Marisol, Joanvi, Geni, Elena, Maribel, Yuri, Rocío, Benjamín, Delia, Siou Y, José, Sara, Manuel, Eduardo, another Pilar, Juanma, Álvaro, and many others that not for being omitted have been less important to me.

And last but not least, to my family, for their capacity to listen and their wisdom to advice, and, above all, for giving me their unconditional support, regardless of what I was doing with this thing of a planar grasp or what that had to do with the degrees of freedom of a control task.

Gabriel Recatalá Ballester
Castelló, October 2003

Contents

List of Figures

List of Tables

List of Algorithms

Notations

Points, lines and planes

P	Point expressed with respect to a 3D coordinate frame
p	Point expressed with respect to a 2D coordinate frame
p	Point expressed in image space coordinates
P_i, p_i or p_i	Point corresponding to index i in an indexed list of points
(X, Y, Z)	Coordinates of a point with respect to a 3D coordinate frame
(X, Y)	Coordinates of a point with respect to a 2D coordinate frame
(u, v)	Coordinates of a point in image space
$\mathbf{V}$	Vector expressed with respect to a 3D coordinate frame
(V_x, V_y, V_z)	Coordinates of a vector with respect to a 3D coordinate frame
I	Image
$\Re^n$	n-dimensional space of real numbers

Frames and frame transformations

$\mathfrak{F}_c$	Coordinate frame c
${}^{c'}T_c$	Transformation matrix that translates the coordinates of a point from frame c to frame c'
${}^{c'}M_c$	Transformation matrix that translates velocities expressed in frame c to frame c'
${}^{c'}x_c$	Manifold that contains the translation and rotation components of a transformation matrix

Camera parameters

f	Focal length
(u_0, v_0)	Principal point

Matrices

$\mathbb{I}$	Identity matrix
$\mathbb{O}_n$	Null square matrix of dimension n
$\mathbb{H}$	Homography matrix
$[v]_\times$	Anti-symmetric matrix associated to vector v

Operators

$\mathbf{A}^t$	Transpose of matrix **A**
$\mathbf{A}^{-1}$	Inverse of matrix **A**
$\mathbf{A}^{\dagger}$	Pseudo-inverse of matrix **A**

Control law

$\mathcal{T}$	Task space of the robot, $\mathcal{T} \subseteq \Re^m$
$\mathbf{r}$	Pose of the robot, $\mathbf{r} \in \mathcal{T}$
$\dot{\mathbf{r}}$	Velocity screw of the robot
$\mathcal{F}$	Image feature parameter space, $\mathcal{F} \subseteq \Re^n$
$\mathbf{s}$	Image feature parameter vector, $\mathbf{s} \in \mathcal{F}$
$\dot{\mathbf{s}}$	Velocity of the image feature parameters
$\mathbf{L}_{\mathbf{s}}(\mathbf{r})$	Interaction matrix between $\mathcal{F}$ and $\mathcal{T}$ with respect to a vector $\mathbf{s}$ of feature parameters $\mathbf{L}_{\mathbf{s}} \in \Re^{n \times m}$

Chapter 1

Introduction

This section explains the motivation and the goals of this thesis.

1.1 Motivation

The manipulation of objects is a fundamental class of operations in many areas of robotics, such as space, industry, medical and service robotics. This class includes both *fixturing* –restraining an object with the fingers of the robot hand– and *dexterous manipulation* –use of the fingers to change the position of the object within the robot arm– [18]. Most grippers are used for fixturing, and not for dexterous manipulation. In order to simplify the notation, the term *grasping* will be used along this thesis in the sense of fixturing.

In both cases, the ability of the robotic system to deal with modeled, unmodeled and unknown objects is very important. In the latter two cases, the system must be able to use its sensors to detect and extract the necessary information about the object. In general, in those situations in which the robot has to operate in dynamic and loosely structured environments, the required object description must be extracted from sensor data, instead of being already available as a model. This information is then used to decide on the most convenient way to grasp the object and to execute that grasp, possibly taking into account some constraints imposed by the task to perform with it.

Predefined object models for the grasp search have been used by many [12, 32, 56, 110]. Service robots, however, require the object description to be extracted from vision data. In this case, as the search of stable grasps on the object is based on the analysis of the shape of the object, the problem is how to extract such a description from an image or a set of images. In addition, if the relative position between the object and the camera changes, because one of them -or both- is moving, there is, in order to control the approach of the gripper to the object, a need to track not only the object itself but also the points at which it should be grasped. Moreover, the manipulation of objects requires not only a grasp search, but also the execution of a whole sequence of steps, which may change depending of the way the manipulation task is defined. This sequence could include the selection of the object of interest, the selection of a grasp for this object, the positioning of the grasp tool around the object, the grasping of the object, and possibly the execution of some action with the object.

The above problems can be considered as globally unsolved in the current state of the art of the field, specially when considered as a global task. From them, this thesis addresses the required set of steps for the execution of an approximation movement towards the object and proposes an architecture to integrate these steps in a seamless fashion.

With respect to the selection of the object, one of the difficulties is locating it and extracting a useful description of it from sensor data. When vision is the source of these data, images where there is a clear distinction between the object and the rest of the scene have been normally used, in order to extract a description as accurate as possible [130, 149, 172]. Nevertheless, although there has been a lot of effort over the last years to relax the working conditions (including light and type of background) of this segmentation, its precision is still highly dependent on them.

The positioning of the robot with respect to the object has been performed in many classical systems [110]. In systems where sensors –typically vision– have been used in this step, the positioning has been usually performed with respect to features other than the grasp points –such as some marks explicitly set on the object, or some geometric parameter related to its projection. In some cases [84, 203], the grasp search and the computation of the target position between the robot and the object are performed in an off-line step.

Most reported works on object manipulation have focused only on a specific aspect of the task. Some authors [104] have suggested the use of a different control strategy for each of the steps of the grasp-execution task. One of the earliest works on integrated systems for the execution of the task of grasping object (*autonomous object manipulation*) was reported by Ikeuchi et al. [91]. This system could grasp 3D real-world objects with previously known geometry in scenes with a highly structured lighting system. Another well-known system is Handey, presented by Lozano-Pérez et al. [110], which was a vision-guided robotic grasping system capable of recognizing known objects of polyhedral geometry in cluttered scenes, grasping, re-grasping if necessary, and manipulating the objects.

Stansfield [186] reported a vision-based system for grasping unknown 3D objects that used an expert system to generate grasps for multi-fingered hands. A structured lighting system was used to perform a 3D reconstruction of the environment. Bendiksen and Hager [15] reported another vision-based system that performed a 3D reconstruction of the extracted contour of an object. Two parameters were used to generate a set of grasp candidates. Grasps were evaluated according to the force required from the fingers on the object to achieve an equilibrium –rather than a force-closure– grasp. A blind movement of the robot towards a 3D reconstructed location of the grasp points was executed, requiring one and a half minutes from selection to execution of the grasp on a Sun SPARC 10 computer.

Bard et al. [12] describe a system that includes a three-fingered hand, a stereo head mounted on a second robot arm. This system performs a 3D reconstruction of the observed scene and plans the execution of the grasp based on this reconstruction. Sanz et al. [172] introduce a simpler integrated system, in which they use a single camera mounted on a robot arm with a two-finger, parallel-jaw gripper to compute and execute 2D grasps on planar objects. In [177], Seitz mounts a three-fingered gripper with tactile sensors on a robot arm; a tiltable camera is also mounted on the robot arm; 3D reconstruction is achieved through movements of the camera over the object to manipulate; the grasp search is performed on-line based on a rough 3D reconstruction of the object. Other examples of complete grasping systems include [50, 172, 99, 1].

In many of these systems only a control task is defined and there is no specification on how to add other tasks that could be simultaneously executed or how to define a sequence of tasks. Nevertheless, some complex tasks such as manipulation involve the execution of a sequence of steps. The use of a unified visual servo strategy for such tasks is still a matter under consideration [104]. In order to take them into account, some authors have proposed a modular design for the control structure. For instance, Kosecka et al. [102] modeled a visual servo system as a finite state automata (FSA). The states in this automata corresponded to the execution of actions, and the events to observations and actions; events can cause transitions between states. A similar approach was suggested by Dodds et al. [50].

Some architectures for dealing with more complex behaviors have been developed, mainly in the field of mobile robotics, such as the *subsumption architecture* or the *motor schema architecture*. Grupen and Henderson [70] applied their architecture, based on *autochtonous behaviors*, to grasping and manipulation tasks.

This thesis visual determination, tracking and execution of an unknown or unmodeled object. In particular, it is considered the case in which the object is ideally planar –in practice, relatively flat– and the grasp has to be executed with a two-fingered, parallel-jaw gripper using vision as the source of sensor data. Each step in this task is analyzed separately, considering several options in some cases. In addition, the use of an architecture for integrating them and building a complete grasp-execution system is studied.

The rest of this chapter is organized as follows. Section 1.2 provides an overall description of the problem considered in this thesis. Next, section 1.3 summarizes the main contributions of this work. Finally, section 1.4 outlines the contents of the following chapters of the thesis.

1.2 Problem description

The task considered in this thesis is to use the visual information provided by a vision system to guide a robot arm towards an object, so that the fingers are eventually positioned around or close to a pair of grasp points, each of them belonging to one of the contours of the object. The object is assumed to be relatively planar and to lie on a flat surface that is initially parallel to the image plane of both cameras. The cameras are rigidly attached to the robot arm, close to the gripper, so that they move together with it. The main goal can thus be stated as the development of a control structure for executing the task of performing an approximation movement in order to grasp a relatively flat object with a two-finger, parallel-jaw gripper in a non industrial environment.

1.2.1 Physical setup

The main physical elements involved in the problem considered are introduced here:

- **Object to grasp.** The object is not previously known, so no model of it is available. It is assumed to be rigid and planar. Ideally, it can be considered as a shape that lies on a given plane. In practice, it will be considered as an extrusion of its projected silhouette. It is also assumed that the object remains static within the workspace during the execution of the control task. Finally, the object is also assumed to lie on a flat surface.

- **Gripper and robot arm.** The gripper is attached to the end of a robot-arm. Grippers with two flat parallel fingers are considered. For grippers with more than two fingers,

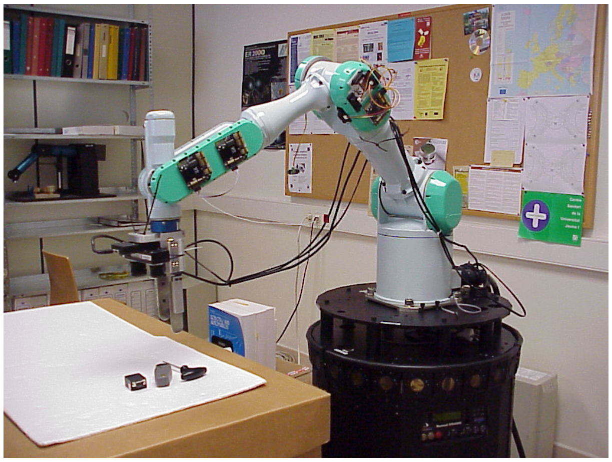


Figure 1.1: Main elements involved in the considered problem.

only a configuration with two of them parallel or a configuration with two parallel fingers will be considered. Therefore, each of the grasps selected for an object will consists of two points on the border of its silhouette. These points will be named *grasp points*. The gripper features a minimum and a maximum opening of the fingers.

- **Vision system.** Due to its versatility and inexpensiveness, the combination of a simple two-fingered gripper and a single camera, either in an eye-in-hand or in a fixed-camera configuration, is the fundamental configuration for vision-based grasping [77, 185, 175]. A stereo head with a pair of cameras has been considered in this thesis. The stereo head is mounted at the end of the robot arm, close to or on the gripper. There is a rigid relationship between the camera and the base of the gripper. This relationship can be learned –using methods such as [51, 206]– or be previously known. As the cameras are mounted on the robot gripper, the position, orientation and size of the projection of the object in the images provided by the cameras may change as the gripper moves closer to or away of the object. It is assumed that these cameras are rigidly attached to the stereo head, so that they cannot move independently from each other.

Figure 1.1 illustrates the relationship between the main elements involved in this problem.

1.2.2 Goals

The objective considered in this thesis comprises the following subgoals:

- **Definition of a mechanism for the selection of an object in an image.** A procedure to segment an object of interest from the background and other objects in an image will have to be defined, as well as the conditions under which such segmentation will be possible.

- **Definition of a mechanism for the selection and tracking of grasp points.** The criteria according to which a grasp will be evaluated as stable will be specified, and a grasp search procedure will be defined. In addition, as the grasp points are going to be used as control features to guide a robot arm towards an object, it is necessary to have a procedure to track them along the images captured during the positioning task.

- **Definition of a control law to guide a robot arm towards the object to be grasped.** This control law will have to consider the available information and the associated restrictions regarding the location of the object and the grasp points.

- **Definition of an architecture for the integration of all the components of the control task.** This architecture should be defined in such a way that helps to simplify the development of the individual components and the connection between them. In addition, it should be possible to extend it in the future with new structures in order to define more complex tasks.

1.3 Contributions

Some of the main contributions of this thesis are:

- **Development of a method for the evaluation and search of a feasible grasp on an object for a two-fingered gripper.** This method is based on the analysis of the contour of the object [130, 175, 132].

- **Development of several methods for tracking a selected grasp.** These methods have been applied to perform the tracking between pairs of stereo images and along a sequence of views of the object [160]. In particular, the following tracking methods have been proposed:

  - *Based on the structure of B-splines* [160]. A B-spline associated to the object along a sequence of images has been used as a reference frame to define the location of the grasp points.
  - *Based on second-order moments of the object.* In this case, the reference frame is provided by geometrical features –the centroid, area, and axis of minimum inertia– that are computed through second-order moments of the object.
  - *Based on the computation of an homography between pairs of images.* This homography has been used to translate grasp points from one image to another one. Adjustments to this translation have been made to make sure that the translated points belong to the object.

- **Definition of a vision-based control scheme to guide the gripper of a robot arm (*Gripper-to-object positioning*) towards an unknown object.** The selected grasp points are used as control features. The number of degrees of freedom available for controlling the robot depends on the selected method for tracking the grasp points. In addition, a procedure is defined to automatically compute the target position of the grasp points based on their current position and other system parameters.

- **Definition of a behavior-inspired architecture for the specification of a task.** This architecture has not only been used to define the grasp-execution task considered in this thesis, but also for the specification of other tasks for a different physical setup [148].

- **Analysis of several strategies for obtaining a more compact representation of a contour (*curve segmentation*).** The following strategies have been considered:
  - Computation of polygonal approximations of the contour using genetic algorithms [159, 193].
  - Automatic initialization of a B-spline that approximates the shape of the contour [67].

1.4 Outline of the thesis

The rest of this thesis is organized as follows:

- **Chapter 2** provides a brief description of the approach followed in this thesis.
- The architecture considered for developing the grasp-execution control task is outlined in **chapter 3**.
- The proposed procedures for the selection of the object from an image are explained in **chapter 4**. This chapter also analyzes several options for a more efficient representation of the contour of an object and their suitability for the grasp-execution task.
- The procedures corresponding to the grasp search and the grasp tracking are introduced in **chapter 5**.
- **Chapter 6** provides the definition of the control law used to guide the robot based on the location of the grasp points.
- Finally, **chapter 7** provides some general conclusions and briefly describes some lines of future work.

Chapter 2

Methodology

The problem considered in this thesis and the restrictions imposed are explained here.

2.1 Development of a control task based on visual servoing

The execution of a selected grasp is seen here as a positioning problem that involves the object to grasp and the fingers of the gripper. This execution is defined as a visual servoing control loop in which the visual control features considered are the grasp points computed for a selected object. The goal of the control loop is thus to guide the robot arm towards these points.

2.1.1 Main concepts in visual servoing

Visual servoing is a robot control approach in which visual information is used as feedback input to control the *pose* –usually interpreted as the position and orientation– of a robot with respect to a given object or a set of target features [41, 90]. The target pose of the robot in this controlled movement is referred to in some works [16, 121] as the pose of the robot *at equilibrium*. The work of Shirai and Inoue [182] from 1973 was one of the earliest to use vision inside the control loop. Nevertheless, the term *visual servoing* was first used by Hill and Park [80] in 1979, in order to distinguish their approach from those in which vision was used in an open-loop way, with observation/interpretation and movement steps being alternatively executed to perform a task. A more generic term, *visual feedback*, had been used until then. In open-loop systems, vision data had been used to produce a world representation, which was then considered to plan a task; the task was then executed with blind movements, not allowing changes in the environment.

With visual servoing, vision is included in a closed-loop control system, making it possible to work with non-static objects. It is thus possible to build more flexible robotic systems, since a strict control and previous knowledge of the environment is not required. The increase of computational power since the introduction of visual servoing has lead to a increasing number of built systems and has allowed the use of more sophisticated and robust techniques for the extraction of visual features inside the control loop. Several introductory papers and reviews on the evolution of visual servoing since its early years can be found in the literature [40, 41, 74, 90, 104].

In visual servoing, relevant features are extracted from the camera –or cameras– available in the system in order to feed the control loop. Hutchinson et al. [90] define an *image feature* as any structural feature that can be extracted from an image –such as an edge or a corner– and an *image feature parameter* as a quantitative value that can be computed from one or more image features. According to Espiau et al. [55], an *image feature* corresponds to the projection in the image plane of an *scene feature*, which can be defined as a set of 3D elements –such as points, lines or vertices– rigidly attached to a single body. The set of n image features s_i considered for controlling the robot can be grouped in a vector $\mathbf{s} = [s_1, s_2, ..., s_n]^t$. It can be considered that $\mathbf{s} \in \mathcal{F} \subseteq \Re^n$, where $\mathcal{F}$ represents the *image feature parameter space*.

The selection of image features depends on the task to perform. However, features that can be easily extracted and tracked should be chosen [28, 162], in order to provide the control law with new input data at an adequate frequency. The use of prediction methods to estimate the location of the image features can help to improve the robustness and the speed of the tracking [2, 198]. These methods can also be useful to handle the problem of the occlusion of features. A review of feature tracking methods used in visual servoing works can be found in [40].

The visual servoing control law uses the values of the image features to determine the movements the robot should perform in its *task space*. The *task space* of the robot, represented here by $\mathcal{T}$, is the set of poses –that is, positions and orientations– that the robot can achieve [90]. Being the configuration space of the robot, it can be seen as a smooth m-manifold [106], where m is the dimension of the task space. In many applications, the robot is abstracted as a device that can move through a 3D space and $\mathcal{T}$ is considered as $\mathcal{T} = SE^3 = \Re^3 \times SO^3$, where $\Re^3$ represents the space of translations and SO^3 the space of rotations with respect to the X, Y, and Z axes. A pose $\mathbf{r} \in \mathcal{T}$ can be expressed as: $\mathbf{r} = [T_x, T_y, T_z, R_x, R_y, R_z]^t$, where values T_i indicate translations and values R_i rotations, for $i \in \{x, y, z\}$. In general, each image feature s_i can be considered as a function of the pose: $s_i = s_i(\mathbf{r})$. In some applications, the task space can be reduced to a subset of the above, that is, $\mathcal{T} \subseteq SE^3$. The dimension of the task space will determine the number of *degrees of freedom* (d.o.f.) that are available for controlling the robot.

Many works have considered the robot as a velocity-controlled device. Therefore, it is important to compute the velocity $\dot{\mathbf{r}}$ in task space to apply to the robot in order to correct the error between its current and desired pose. In the case that the task space is $\mathcal{T} = SE^3$, this velocity can be generally expressed as $\dot{\mathbf{r}} = [\mathbf{T}, \Omega]^t = [V_x, V_y, V_z, \omega_x, \omega_y, \omega_z]^t$, where values V_i correspond to translational velocities and values ω_i to rotational velocities, for $i \in \{x, y, z\}$. This vector $\dot{\mathbf{r}}$ is known as the *velocity screw* of the robot.

In many works [16, 90, 104], the design of the control law has followed the so called *task function* formalism [31, 167]. According to this approach, it is possible to express any servoing scheme according to the regulation to zero of a function called the *task function*, or *control error function*. When the current pose of the robot matches the target or desired one, the value returned by the task function should be zero.

With the assumption that the object of interest is motionless, the most common approach for the control law is a simple proportional control law [90, 104]. Some works on an optimal design of the control law can be found in [75, 144].

2.1.2 Classification of visual servo systems

Sanderson and Weiss [168] introduced a taxonomy of visual servo systems based on two criteria:

- **Organization of the control structure.** This criterion is related to the level at which the control law computes commands for the robot. Two types of systems have been distinguished:

    - *Dynamic look-and-move systems.* The control architecture is hierarchical, so that control is performed at two levels. In a first, higher level a control law produces coordinate velocities or points that are set as input to a second, lower level controller that is in charge of stabilizing the robot at joint level. Most of the reported systems in the literature follow this approach [90, 104].
    - *Direct visual servo systems.* In this case, there is a single controller that directly computes joint-level commands for the robot.

- **Computation of the error signal.** This criterion considers the space in which the difference or error between the current and the desired pose of the robot –and, therefore, the task function– is computed. Two types of systems were distinguished in the work of Sanderson and Weiss [168]:

    - *Position-based visual servo systems (PBVS).* The error is computed in 3D Cartesian space, and both the current and the desired pose of the robot have to be expressed in this space. These systems use the image features to perform an estimation of the current pose of the object of interest, usually with respect to a camera-attached coordinate system. The computation of this estimation often requires knowledge of the internal parameters of the camera and in some cases a model of the object. In this type of systems, the task function is also referred to as *kinematic error function* [90] or *virtual kinematic constraint* [55, 90, 167].
    - *Image-based visual servo systems (IBVS).* Actually, Sanderson and Weiss had proposed the term *visual servo* for this type of systems. Hutchinson et al. [90] suggested the term *Image-based visual servo systems (IBVS)*, since *visual servo* have been widely used to make reference to any closed-loop vision-based control system. In IBVS systems, the error is computed directly in image space. The target pose of the robot, as well as the current pose, are expressed in image space. In these systems, the task function is also called *image error function* [90].

    In addition to the two types above, a third one could be considered [41]:

    - *Hybrid methods.* These methods use IBVS to control certain degrees of freedom (d.o.f.) of the robot and use other techniques to control the remaining ones. Some of the main approaches in this category are:

        - ∗ *Visual compliance.* IBVS is used to control translation parallel to the image plane and Cartesian PBVS to control depth [28, 58].

- ∗ *$2\frac{1}{2}$D visual servo.* In this case, IBVS is used to control translational degrees of freedom and rotational motion is controlled through the recovery of the epipolar geometry between the current and target images [114, 115]. Similar approaches can be found in [47, 133].

One of the problems of PBVS systems is that they depend on a precise system calibration –including the calibration of the camera and the relationship between the camera and the robot. In addition, the estimation of the pose of the object of interest often requires the use of a model [104]. Some works that follow this approach are [2, 52, 162, 199, 200]. Several model-based methods for the recovery of the 3D pose of an object are described in [48, 53, 65]. A survey on works that make use of object models can be found in [188]. Although there are many works for the 3D reconstruction of unknown objects [23, 37, 127, 191], they are in general too computationally demanding for the real-time requirements of a visual servoing system. Nevertheless, these methods perform a complete 3D reconstruction of the object, which is rarely necessary for visual servoing. Therefore, methods could be developed –or adapted– that reconstruct only those object features that are really needed by the visual servoing control law [41].

IBVS systems are based on the assumption that if the current view of the object of interest matches the target or desired view, then the object must be in the desired relative pose with respect to the robot [41]. This target is often obtained following a “teach by showing” approach. In an off-line step, the robot is moved to the desired position, where the target image features are extracted and recorded. After that, the robot is moved to an initial position, from which the visual servo approximation movement starts [104]. In some cases, the target view is defined as a fixed relative position with respect to the image, with no learning step [16]. Horaud et al. [84] learned the target pose in projective space and projected it during the approximation step onto the image space. In general, IBVS is considered very robust with respect to camera and robot calibration errors [90, 196]. In fact, a coarse calibration of either the cameras or the robot affects only the time needed by the control law to converge [104]. Some other works that follow the IBVS approach are [55, 73, 76, 121].

Visual servoed systems can also be categorized based on their robot-camera configuration. In this case, the following criteria can be considered [90, 104]:

- **Number of cameras.** Typically one or two cameras have been considered. In some works, a redundant vision system is built with more than two cameras.
- **Camera configuration.** The following options are available:
  - *Eye-in-hand.* The camera –or cameras– is mounted on the end-effector of the robot. With this configuration, it is possible to have a more detailed view of the object of interest.
  - *Stand-alone.* The camera –or cameras– is fixed on the workspace of the robot. This configuration provides a wider field of view of the scene.

The use of a single camera in an eye-in-hand configuration has been a very common setup in many reported works [104]. In this case, the *hand-eye calibration* –that is, the transformation between the end-effector and the camera coordinate frames– is assumed to be known. Works with this configuration using image feature [84, 142] or model-based [53, 57, 103] tracking techniques can be found in the literature. A single camera in a stand-alone configuration was more common in early systems [182]. Other recent works with this approach are described in [57, 105, 192, 203].

In systems with stereo pairs of cameras, the usual approach is to estimate the disparity and the depth of the scene [101, 120]. However, one of the problems with regard to this computation is the detection of matching features between two or more images. The use of a stereo head mounted on the end-effector is less common than in a stand-alone configuration, since in the latter is easier to make the *baseline* –the line joining both cameras– long enough to obtain an accurate depth estimation [104]. Some systems using a stereo head in a eye-in-hand configuration are described in [100, 116, 155]. Some examples of two cameras in a stand-alone configuration can be found in [73, 82, 83].

Hutchinson et al. [90] consider an additional criterion for the classification of visual servo systems:

- **Observability of the end-effector and the target.** This criterion that, in general, the accuracy of the positioning of the end-effector with respect to the target cannot be guaranteed unless both can be observed [73, 83, 90, 197]. Two categories can be considered:
  - *Endpoint open-loop (EOL) systems.* These are systems in which only the target can be observed. Systems following this approach can be found in [172].
  - *Endpoint closed-loop (ECL) systems.* In these systems, both the target and end-effector of the robot can be observed. Although control is more precise with this configuration, the need not only the target, but also the end-effector makes the computational cost of the extraction of image features higher [90]. Some ECL systems have been reported in [203].

2.2 Architectures for robot control

2.2.1 Concept of robotic architecture

Dean and Wellman [46] state that "an *architecture* describes a set of architectural components and how they interact." Matarić [124, 123] defines the concept of *architecture* in the field of robotics as a methodology that "provides a principled way of organizing a control system. However, in addition to providing structure, it imposes constraints on the way the control problem can be solved."

2.2.2 Classical architectures

Many different robotic architectures have been defined in the literature. In a classical approach, a model of the observed scene was build based on sensor data, some task was planned based on this model and then executed without the use of more sensor data. Often, some predefined knowledge was required to build the scene model. In this architecture the control system was organized as a sequence of *functional units* or *modules*, as shown in figure 2.1. This approach was also known as the *sense-plan-act* paradigm.

A classical architecture was behind the design of some of the first autonomous robots, such as Shakey [138]. Another complete and well-known example of this approach can be

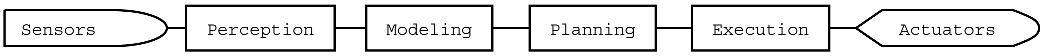


Figure 2.1: Classical decomposition of a robot control task into functional units (adapted from [24]).

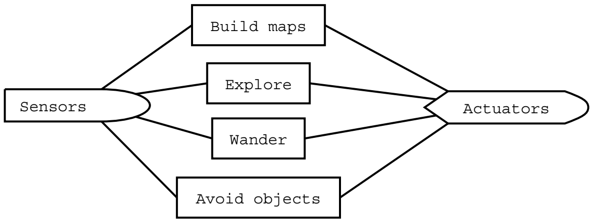


Figure 2.2: Example of decomposition of a robot control task based on behaviors (adapted from [24]).

found in the Handey project [109, 110]. In this project, a complete geometric model of the environment was used with an object recognition module in order to obtain an interpretation of the scene. A plan for a grasp-execution task was then produced based on this model-based interpretation and then blindly executed. This approach has received criticism for not being able to deal with unknown elements in the scene and being computationally expensive, which does not allow the control of tasks in real time.

2.2.3 Behavior-based architectures

Brooks [24] has suggested an alternative decomposition of control tasks in modules that where called *task-achieving behaviors*, or simply *behaviors*, as shown in the example in figure 2.2. In behavior-based systems, the robot controller consists of a structured net of behaviors, each of which is in charge of achieving or maintaining a specific goal [124]. From a psychological point of view, a behavior is essentially a reaction to a sensorial stimulus [10]. From an engineering point of view each behavior is a processing element or procedure and can be considered as a control law in Control Theory [124]. As behaviors within the same system are assumed to be running simultaneously and asynchronously, an arbitration mechanism will have to be defined when more than one is in charge of determining the action to be taken by a given actuator. Behavior-based robotic systems have proved to be more appropriate than the classical ones when the real world cannot be accurately characterized or modeled, since they do not rely on a model of their environment [10]. Otherwise, the effort in developing them may not be worth, due to the accurateness and efficiency that classical systems have shown in highly-structured scenarios. However, although many applications have been described in the literature –a good review can be found in [10]–, most of the described works focus only in the field of mobile robotics.

Brooks [24] proposed a purely *reactive* behavior-based system, in which individual behaviors are designed as simple as possible and no internal representation nor time history are taken into consideration. Behaviors are grouped in several *levels of competence*, each level corresponding to a control task. These levels of competence are developed from the lowest to the highest level, following an bottom-up approach, so that each level includes a subset of a lower level. This architecture was named the *subsumption architecture*, since each level subsumes the levels over which it is defined.

Each layer is build from a set of processors or modules that send messages to each other. In an earlier version of this architecture [24], each module was defined as a finite state machine (FSM) and had the ability to hold some data structures. Modules interchanged data through input and output lines. Later [26], augmented finite state machines (AFSMs), with the ability to share registers were considered.

Coordination between levels is achieved in this architecture through two mechanisms: the inhibition by one module of the output of another module and the suppression of the output of one module by another one, which provides a replacement output.

As the subsumption architecture had received some criticism due to the difficulty of specifying a task, a *behavior language* [26, 25] was defined that groups the AFSMs into more manageable units. Nevertheless, some authors [10] consider that a significant learning curve is still associated to the development of tasks with this architecture.

Another well-known behavior-based architecture, more strongly inspired by biological sciences, was introduced by Arkin [10, 7, 6] and is based on results from the *schema theory* [4, 3]. The schema theory appeared in neurophysiology in the beginning of the twentieth century. One of the first reported applications was related to the explanation of postural control mechanisms in humans [78]. Several definitions have been given of the term *schema*. According to Arbib [3], a *schema* is "an adaptive controller that uses an identification procedure to update its representation of the object being controlled". Arkin [10] defined it as "the basic unit of behavior from which complex actions can be constructed", and added that "it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted."

In the field of autonomous robotics, schemas provide a larger grain modularity than other models such as neural networks. In a schema-based architecture, schemas define therefore a collection of behavioral primitives that can act distributedly and concurrently to build a more complex behavior in response to stimuli from the environment [10, 9]. In fact, they can be considered as software objects that can be easily reused [112].

Motor schema behaviors, or simply *motor schemas*, are large-grain behavioral modules that are connected to environmental sensors and provide an output for the actuators of the robot [10, 8]. Internally, each behavior includes some modules known as *perceptual schemas*, which provide information about the environment that is specific to that behavior, following an *action-oriented perception* approach. Each perceptual schema can, in turn, include other more specific perceptual schemas. Motor schemas act concurrently and asynchronously. Coordination between them is based on determining the overall response of the robot considering the relative strength of each schema. Figure 2.3 illustrates the definition of a task based on motor schemas.

Other behavior-based architectures have been proposed. Their peculiarity arises mainly from the behavioral granularity, the design methodology, the encoding of the output of the behaviors and the coordination mechanism. Some of them are:

- *Circuit architecture* [98]. Reactive behaviors are expressed here using a logical formalism, which allows an evaluation of the performance and properties of the system. Behaviors are grouped into levels of abstraction, and coordination is carried out

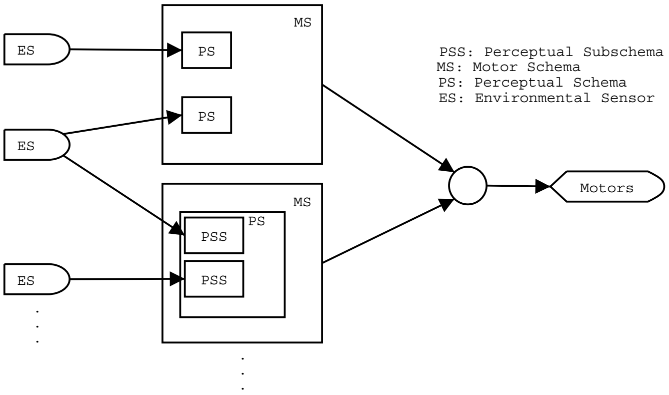


Figure 2.3: Decomposition of a robot control task based on motor schemas (adapted from [10]).

through arbitration within each level first and then between levels.

- *Colony architecture* [38, 39]. Direct descendent of the subsumption architecture, it allows a more flexible specification of the relationships between behaviors. A hierarchical ordering of priorities between behaviors is considered for coordination. A well-known development based on this architecture is Herbert [27], a mobile robot designed to wander about the corridors of a building and pick up the soda cans it finds.

- *Autochtonous behaviors* [70]. Unlike many behavior-based architectures, its development has been oriented towards grasping and manipulation tasks, instead of robot navigation. It relies more on models built from sensor data than other behavior-based systems.

A more exhaustive review can be found in [10].

In addition to the *behavior language* [26, 25], other specific languages for some behavior-based architectures have been described, such as [176, 195]. Horswill [86] proposed a neutral language that can be used to implement several architectures.

2.2.4 Hybrid and other architectures

Behavior-based systems have proved their effectiveness in dynamic and complex domains. Nevertheless, in some situations, such as in industrial workcells, where it is possible to have a strict control over the environment, classical, deliberative systems have been often preferred. In fact, the reactive approach has been appropriate to deal with the immediacy of sensory data but also, unlike deliberative systems, less effective in integrating world knowledge [10]. These architectures have also been criticized for limiting their work domain to the imitation of abilities of low-level life forms such as insects.

Some research works in psychology and neuroscience [140, 180] have indicated the existence of two models of behavior: willed and automatic. Automatic behavior is associated to reactive systems, while the willed behavior is in charge of conscious control, and provides an interface with the automatic behavior. For this reason, some researchers have advocated for the integration of deliberative reasoning in behavior-based systems in order to fully exploit their potential. According to this approach, the use of representational knowledge is necessary to extend the work domain of autonomous robots. Architectures designed based on this principle are known as *hybrid architectures*.

Hybrid architectures often include at least one layer to represent deliberation and another one to represent reactivity. Sometimes there is an additional interface layer between them [79]. The deliberative layer can be in charge of configuring the reactive layer, giving it suggestions, modifying some specifications, or taking some decisions when it is necessary. Some developed hybrid architectures include AuRA [5], Atlantis [63] and the Generic Robot Architecture [139].

In the industrial domain, many reported applications have a modular and distributed architecture, they still follow a hierarchical, classical approach [72, 107]. Nevertheless, hybrid architectures have been successfully applied in these environments too, where the term *behavior-based assembly architecture* has also been used [150, 113]. In [150], the deliberative layer is used as a planner, while the reactive layer deals with uncertainties and hardware and environment details. Different implementation of the same behavior could exist –*behavioral modules*– that take into account different work conditions.

In other approaches, a network of agents have been proposed to control the robot [141, 62].

2.3 Proposed approach

As it has been mentioned in section 2.1, the execution of a selected grasp is considered as a positioning problem involving the object and the fingers of the gripper. The goal is thus to move the robot arm so that a relative desired position between the object and the fingers is achieved [84]. One of the goals of the work developed in this thesis has been to provide a framework that could be used not only for the grasp execution, but also for a wide variety of similar tasks. An effort has been made to allow the independent design and development of each of the functional components the task is composed of, as well as to ease the connection and integration of these components.

Several architectural developments have been studied for this purpose. Nevertheless, only those features that have been considered necessary to define a simple task like the grasp execution have been taken, in order to keep the architectural design as simple as possible. However, attention has been put to allow the extendability of the architecture. Some indications are given on how to include support to complex tasks, in which several actions have to be executed simultaneously, and also to the possibility of executing a sequence of tasks, allowing the switching from one task to the next one depending on the observation of the environment.

With regard to the grasp-execution task, it has been defined as a visual servoing control loop. The visual control features considered in this loop are the grasp points for a selected object. The goal of the control loop is to guide the robot arm towards these points.

Information on the location of the object is extracted from the images provided by the stereo pair of cameras. The basic representation of this object is its shape, described by its contours. Several options have been considered for the extraction and selection of the object, as well as for an internal representation of the contours. The contours of the object are used for the selection of a pair of points belonging to the contours that define the position at which the fingers of the gripper should be placed to stably grasp the object. An analysis of grasp stability is performed and a procedure for the search and evaluation of grasp point is provided. In addition, as the perceived location in image space changes as the robot moves towards the object, several strategies for tracking the grasp points along the sequence of images, as well as to find them in a pair of images, is defined.

As the extremes of the gripper fingers, which provide a hint of the target location of the grasp points, may not be observable in the image captured by the cameras. Therefore, they have not been considered as a reference. Instead, a mechanism is proposed to determine in image space a target location. Nevertheless, this location may not be the position at which the fingers of the gripper can closed, but a closer one.

The definition of a visual servoing control law based on the location of the grasp points in image space has been made taking into account the restrictions imposed by the grasp tracking strategy. A basic control law controlling four degrees of freedom of the robot has been defined and tested. Indications are given for the use of a more flexible law with 6 degrees of freedom.

Finally, an effort has been made to use simple and fast algorithms for the development of the main components of this task, in order to allow an on-line extraction –that is, inside the control loop– of most of the information required for executing the task.

The main concepts regarding this approach are developed in the following chapters of this thesis.

Chapter 3

Architecture for the control task

This chapter describes the methodology followed to solve the proposed problem.

3.1 Proposed architecture

3.1.1 Overall description

A component architecture is defined here following a behavior-based approach, due to its ability and efficiency in dealing with complex and dynamic scenarios. The goal, nevertheless, has not been to define a complete architecture –such as the ones described in section 2.2– in a first step, but one simple enough to solve the problem that has been considered in this thesis. The architecture will evolve to a more complex form as more complex problems or situations are considered in future works. The complexity could therefore be increased taking into account the experience with previous developments.

The proposed architecture is built around three basic types of components:

- **Virtual sensors**. They provide data acquired from real sensors installed in the system (cameras, infrared cells, etc.).
- **Virtual actuators**. They are components that receive commands or data to be sent to physical/real actuators installed in the system (robot arm, gripper, etc.).
- **Virtual filters**. These components process the data they receive, from virtual sensors and/or from other filters, and produce some results, which are provided to either other filters or to virtual actuators. They are not only in charge of tasks such as feature extraction, but can also implement elements such as a control law. Filters are connected between them as a chain. A chain of filters, with either a single branch or multiple branches, constitutes a way of grouping these filters into a higher level filter.

Virtual sensors and actuators are connected through a chain of filters. This chain as a whole constitutes the processing of a loop in the control system that it is in charge of performing a given action with the robot. Based on the above types of components, the concept of *behavior* is introduced. A *behavior* is defined in this architecture as a chain of filters –which can also be seen as a higher level filter– that receives and produces the necessary inputs and outputs, respectively, to perform a given action. These inputs and outputs will

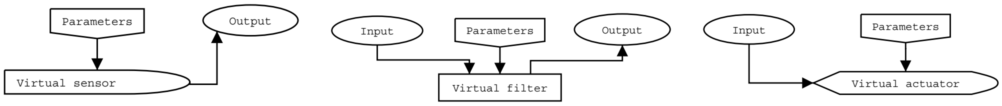


Figure 3.1: External interfaces of a virtual sensor, a virtual filter and a virtual actuator.

normally be directly provided and received by virtual sensors and actuators, respectively. Therefore, a behavior is here simply a higher level filter that will normally be directly connected to the virtual sensors and actuators. A *task* is defined as the set of all connected virtual sensors, behaviors and virtual actuators that are active at the simultaneously within the system.

For simplicity, only one concurrent active behavior has been considered at this initial stage in the development of the architecture. Therefore, no more than one action can be executed simultaneously. In those cases in which two or more behaviors should be simultaneously active a coordination mechanism would have to be defined. This definition has been left for future work. Nevertheless, some indications will be given in section 7.

Virtual sensors, filters –including higher level filters such as behaviors– and actuators will have interfaces, through which they are connected with each other. For the sake of simplicity of design, only three types of interfaces have been considered:

- **Input interface**. It indicates the set of data that a given component requires as input.
- **Output interface**. Specification of the set of data that a given component provides as output.
- **Parameter interface**. Specification of the set of parameters that can be used to configure a component. For each parameter not specified during the setting up of the component, a default value will be assumed.

A filter will have an input, an output and a parameter interface, while a virtual sensor will have only an output and a parameter interface, and a virtual actuator only an input and a parameter interface. Each module will have only one of each of their corresponding types of interface. Figure 3.1 shows the external interfaces of each the types of module considered in this architecture.

In addition to its functionality, it is the set of interfaces of a component what characterizes it as a sensor, a filter or an actuator. This allows to group a set of components into a single unit that can be considered as a higher level sensor, filter or actuator, since this unit keeps the functionality and external structure of that type of module. A chain of filters grouped as a single unit, for instance, can be seen both functionally and structurally as a higher-level filter, as it has been mentioned above. Although they have not been used in this thesis, it is also possible to build higher-level modules with sensors and actuators. For instance, a chain starting with a sensor and involving one or more filters could also be considered as a sensor. Analogously, a chain of filters ending with an actuator would constitute another actuator. Anyway, the components do not even need to be connected in a chain to build a higher-level component, as long as the resulting unit can be classified as such.

The connection between the appropriate components of a task is set when the task is defined and does not change at run time. Each component can be replaced by another one

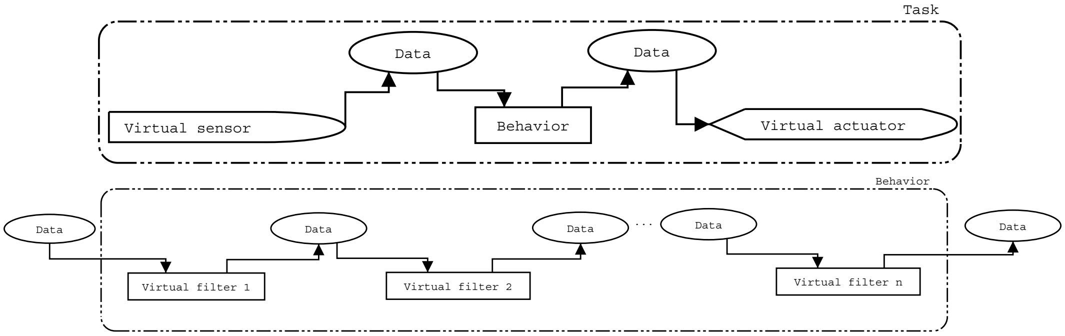


Figure 3.2: Generic definition of a task and a behavior.

with the same definition of each of their interfaces –input, output and parameter, when applicable. As the motor and perceptual schemas in [10, 112, 8], the virtual sensors, actuators and filters are individual modules, although with a different internal structure, that can be easily reused.

Figure 3.2 illustrates the generic definition of a task and a behavior with this architecture.

3.1.2 Data transmission

The transmission of data along the chain of sensors, filters, and actuators is not explicitly defined in this architecture, so that it can be set in the most convenient way depending on the implementation of each task. The only requirement is obviously that the data produced through the output interface of one module can be made available to the input of the other module, or modules, it is connected to. In addition, any module must be completely independent of any possible module or modules that may produce the set of data it uses as input, and also of any possible module or modules that may use the set of data it produces as output. This means that a module cannot be aware of how their input data are produced and of how their output data will be used.

Two data transmission procedures are proposed. The first procedure has been applied to the case in which the connected components share the same memory space. This procedure has been named *shared-memory data transmission*. This is the case, for instance, when these components run on the same computer or are part of the same executable program or module. In this situation, the data to be produced through the output interface of one component is stored, externally to this component, in the shared memory space. Components that require these data at their input interface read them from the shared memory space. Figure 3.3 illustrates how data are transmitted according to this model.

The second proposed procedure is applied when the connected components do not have access to any common memory space. This is the case, for instance, when the implementation of each component is part of a different executable program or module. The components of the control task are therefore distributed among a set of separate blocks that have to communicate with each other. In this situation, they are extended with a communications wrapper and become *extended components*. This procedure is referred to as *remote data transmission*. *Extended components* follow a layered design and are composed of:

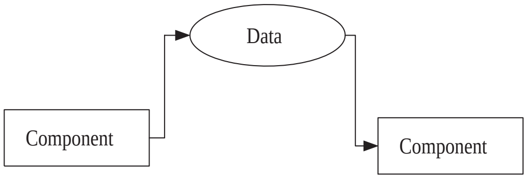


Figure 3.3: Data transmission in the proposed architecture through shared memory.

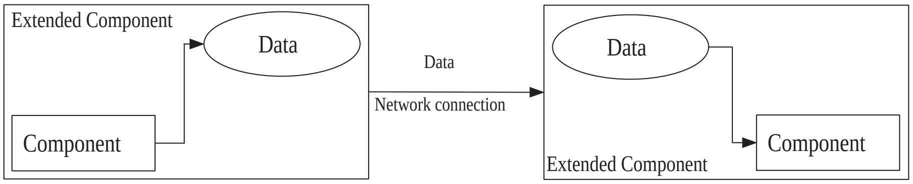


Figure 3.4: Remote data transmission in the proposed architecture between extended components.

1. A *kernel*, which is the original component –sensor, actuator, or filter–, including all its functionality.

2. A *communications wrapper*, which handles the communication with other extended components.

The communications wrapper reproduces the interfaces of the original component –only the input and output interfaces, nevertheless, in the current version of the architecture– and adds the capability of transmitting and receiving the data. Data provided through an output interface is stored in the memory space of the wrapper. They will be at same time requested by the communications wrapper of another extended component, which will provide them to the input interface of its kernel. One extended component can include one or several kernels, with a common communications wrapper for all the kernels. Figure 3.4 shows how the communication is carried out between extended components. Some implementation examples of a communications wrapper are given in [148, 62].

The use of extended components provides and additional level of modularity in the definition of a given task, since it allows to group the task components into separate blocks and distribute in this way the computational complexity of the task. It is therefore possible to use this architecture to define tasks with a scalable degree of complexity. Extended components have been used in [148] to define a distributed control task in an industrial workcell.

3.2 The visually-guided grasp-execution task

The following basic task components have been considered in this thesis:

- **Behavior.** The central component of the task, it takes as input the images and the calibration data sent by the virtual sensors and produces a movement command to

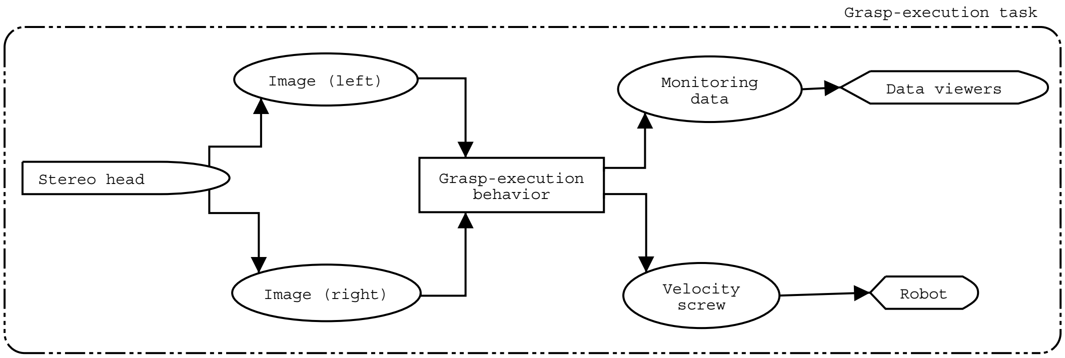


Figure 3.5: Basic structure of the grasp-execution control task.

be sent to the robot, as well as other intermediate data that can be monitored for debugging purposes.

- **Virtual sensors**, which will produce the data to be used as input by the behavior:
  - A sensor that provides the images acquired by the stereo pair of cameras. It also provides calibration data regarding this pair. This calibration data is assumed to be already known or to have been previously computed.
- **Virtual actuators**, which will take their input data from the output of the behavior:
  - An actuator receiving the movement commands to sent to the robot.
  - Data viewers to display the movement commands and the debug data produced by the behavior have also been considered useful.

Figure 3.5 shows how these components are connected.

The behavior is a multi-branched chain of filters in which the following filters are involved:

- **Primary visual perception.** This filter takes one image as input, tries to segment the objects from the background, extracts the contours of the detected objects and selects one of the objects.
- **Grasp search and tracking.** This filter takes as input a description of the object selected by the primary visual perception filter. If this is the first input received, it searches for a stable grasp on this description. On the following objects, it will try to apply a grasp computed for a previous object; if no previously computed grasp is available, a new grasp search will be performed. If available, a description and an evaluation of the grasp selected for or applied to the current object will be produced as output, together with indications of the result of the tracking process, including the availability of a grasp. This filter is used to track a grasp on the objects extracted from a sequence of images.
- **External grasp tracking.** This filter takes as input a description of an object and a description of a grasp. It will try to apply this grasp to the input object. If possible, it

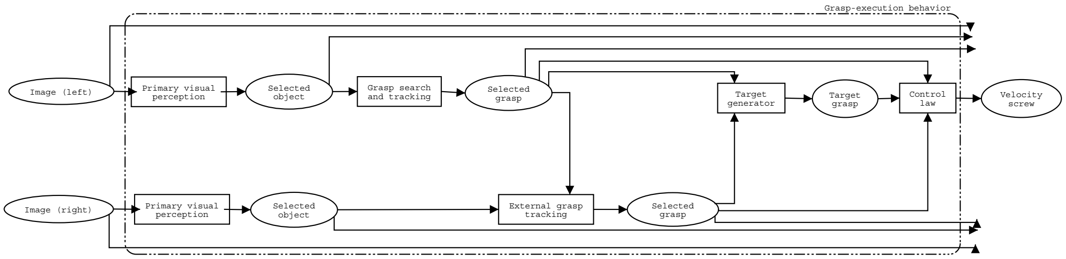


Figure 3.6: Basic structure of the behavior.

will produce a new description of this grasp applied to the new object and an evaluation of this grasp, together with indications of the results of this operations, including the possibility of applying the grasp. This filter is used to apply the description of a grasp computed in one image to an object extracted in another image.

- **Target generator.** It takes as input two descriptions of the same grasp, obtained after processing the images provided by a stereo pair of cameras, together with calibration data regarding this stereo pair. Based on this information, it tries to determine the location in each image where the grasp points should appear in each image when the gripper fingers are placed on the object around them.
- **Control law.** It takes an input two descriptions of the same grasp, each one obtained after processing a different image, as well as target location of the grasp point in each image. It produces a command movement to be sent to the robot.

Figure 3.6 shows how the above filters are interconnected within the behavior.

The internal structure and operation of each of the component filters of the behavior will be explained in the following chapters.

Chapter 4

Primary visual perception: Selection of the object of interest

This chapter provides some basic information regarding the components of the task associated to the vision devices. In addition, the object selection and contour extraction are also explained here. Finally, an analysis is performed on different options to represent the contour.

4.1 Image acquisition

The image acquisition corresponds to the *stereo head* virtual sensor shown in figure 3.5, which is associated to a stereo pair of cameras in the grasp-execution task. Each camera of this pair provides a gray-level image. The image acquisition is assumed to be performed simultaneously by both cameras.

As mentioned in section 3.2, the image acquisition module can provide not only the images acquired by the cameras, but also calibration information referred to the stereo head. This information is composed of two main sets of data: the model of the stereo head itself, which includes the parameters describing the cameras and their relationship within the stereo head, and the model of camera configuration, which describes the relationship between the stereo head and an external reference. Since, in the configuration considered in this thesis, the stereo head is mounted on the robot arm, the robot arm is used as this external reference. This calibration information is required along the grasp execution behavior to interpret the images provided by the cameras and estimate the relationship between the 2D features observed in the images and their 3D correspondences in the scene.

The calibration information is fixed for a given configuration of the vision system, so, in general, it can be known a priori or estimated through an off-line calibration procedure. In this thesis, the explicit use of automatic calibration procedures has been avoided. Instead, rough estimations of the calibration data have been obtained from the documentation of the involved devices –the stereo head and the robot arm– and simple manual measurements. This simplifies the design of the whole task and helps to make other modules more robust by reducing their dependence from an accurate calibration.

Table 4.1: Intrinsic parameters of the model of each camera c in the stereo head ($c \in \{l, r\}$). Coordinates are expressed in pixels and lengths in meters. Scale factors are adimensional.

Parameter	Description
(u_0^c, v_0^c)	Coordinates in image space of the camera center, which is the point at which the optical axis intersects with the image plane.
f^c	Focal length.
k_u^c, k_v^c	Scale factors of the pixels along the directions of the u and v coordinates, respectively.
d_u^c, d_v^c	Lengths of each pixel along the directions of the u and v coordinates, respectively.

4.1.1 Model of the stereo head

The model of the stereo head used in this thesis involves the specification of three sets of values: the *internal* or *intrinsic parameters*, which are associated to each camera, the *external parameters*, which describe the coordinate transformation between the two cameras, and the *image parameters*.

The set of intrinsic parameters used in this thesis is based on the camera model of Tsai [194], which considers the parameters described in table 4.1. In order to make a use of the units that is coherent with that followed in other modules, it will be considered that coordinates in image space in this model are expressed in pixels and lengths in meters.

In addition to the parameters mentioned in table 4.1, Tsai also considered other two, κ_1 and κ_2, which are related to the radial distortion due to the lenses. Nevertheless, the effect of these parameters can generally be ignored in the case of lenses featuring a narrow angle of view, being only more prominent with wider-angle ones. For this reason, and in order to simplify the geometric interpretation of the image, it is assumed that narrow-angle lenses are used and these parameters can therefore be neglected.

The external parameters express the relative location of the coordinate frames associated to the two cameras. Let $\mathfrak{F}_l$ and $\mathfrak{F}_r$ the frames associated to the left and right cameras, respectively, and ${}^{c'}T_c$ the matrix corresponding to the transformation from frame c to frame c' ($c, c' \in \{l, r\}$). The general form of this matrix is:

$$ {}^{c'}T_c = \left[\begin{array}{cccc} & {}^{c'}R_c & & {}^{c'}t_c \\ 0 & 0 & 0 & 1 \end{array} \right] \tag{4.1} $$

where ${}^{c'}t_c = \{{}^{c'}t_{x_c}, {}^{c'}t_{y_c}, {}^{c'}t_{z_c}\}$ is the translation component of the transformation and ${}^{c'}R_c = {}^{c'}R_c({}^{c'}\alpha_c, {}^{c'}\beta_c, {}^{c'}\gamma_c)$ the corresponding rotation matrix, with several conventions existing for the definition of angles ${}^{c'}\alpha_c$, ${}^{c'}\beta_c$ and ${}^{c'}\gamma_c$. In this thesis, the convention is followed to express rotations through the roll-pitch-yaw (RPY) angles (α, β, γ), which describe three consecutive rotations: a first one around the X axis with angle γ, followed by a rotation around the Y axis with angle β, and a last rotation around the Z axis with angle α:

$$ R_{RPY}(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_x(\gamma) \tag{4.2} $$

Table 4.2: External parameters of the cameras in the stereo head. Lengths are expressed in meters.

Parameter	Description
$b_{l,r}$	Baseline. Distance between the origins of the frames associated to the cameras in the stereo head.

Table 4.3: Image parameters related to each camera c in the stereo head ($c \in \{l, r\}$). Dimensions are expressed in pixels.

Parameter	Description
dim_u^c	Dimension of the image along the direction of the u coordinate.
dim_v^c	Dimension of the image along the direction of the v coordinate.

Therefore, the transformation ${}^{c'}T_c$ between the camera frames can be expressed in general through a six-parameter manifold, ${}^{c'}x_c = \{{}^{c'}t_{x_c}, {}^{c'}t_{y_c}, {}^{c'}t_{z_c}, {}^{c'}\alpha_c, {}^{c'}\beta_c, {}^{c'}\gamma_c\}$. The distance $b_{l,r}$ between the origins of frames $\mathfrak{F}_l$ and $\mathfrak{F}_r$ is called the *baseline*.

For simplicity, it has assumed the case in which the optical axes of the cameras are parallel and coincide with the Z axes of their respective frames, and the frame transformation can be reduced to a simple translation:

$$
\begin{aligned}
{}^{c'}x_c &= \{{}^{c'}t_{x_c}, 0, 0, 0, 0, 0\} && (4.3) \\
{}^{c'}T_c &= \begin{bmatrix} 1 & 0 & 0 & {}^{c'}t_{x_c} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && (4.4)
\end{aligned}
$$

with $|{}^{c'}t_{x_c}| = b_{l,r}$. In particular, it has been considered that

$$
{}^{r}x_l = \{-b_{l,r}, 0, 0, 0, 0, 0\} \tag{4.5}
$$

In this ideal case, and assuming the same focal length in both cameras, the image planes are coplanar. The X and Y axes of $\mathfrak{F}_l$ and $\mathfrak{F}_r$ are also assumed to be aligned with the directions of the u and v coordinates in the image space. Under these conditions, the only external parameter required to describe the relationship between both cameras of the stereo head would be the baseline, as indicated in table 4.2. As it has been mentioned in the case of the internal parameters, in order to keep a coherent use of units among different modules, the baseline, being a length, will be expressed in meters.

Finally, the set of image parameters, described in table 4.3, completes the information provided by the internal and external parameters of the cameras and gives some basic information about the image produced by the cameras.

The information provided by the above sets of parameters can be used to compute the 3D coordinates of a point from the known pixel coordinates of its projection on the images provided by both cameras, as well as to compute the pixel coordinates of the projection of a known 3D point. The definition of these computations depends on the camera projection model selected. Some common camera projection models are the *perspective projection*, the *scaled orthographic projection*, and the *affine projection* [83, 85, 90]. Among them, it is the perspective projection model the most widely used [90]. This projection model will be assumed through the rest of this thesis.

Following the perspective projection model, given the pixel coordinates $p^l = (u^l, v^l)$ and $p^r = (u^r, v^r)$ in the image space of the left and right camera, respectively, corresponding to the same physical point P, the 3D coordinates $P^c = (X^c, Y^c, Z^c)$ of this point, expressed with respect to the coordinate frame of camera c ($c \in \{l, r\}$) can be recovered as:

$$Z^c = b_{l,r} \frac{F_u^c}{d_u^c |(u^l - u_0^l) - (u^r - u_0^r)|} \tag{4.6}$$

$$X^c = Z^c \frac{d_u^c (u^c - u_0^c)}{F_u^c} \tag{4.7}$$

$$Y^c = Z^c \frac{d_v^c (v^c - v_0^c)}{F_v^c} \tag{4.8}$$

where:

$$F_u^c = f^c d_u^c \tag{4.9}$$

$$F_v^c = f^c d_v^c \tag{4.10}$$

In the ideal case in which ${}^r x_l = \{-b_{l,r}, 0, 0, 0, 0, 0\}$, the following can be applied:

$$Z^r = Z^l - b_{l,r} \tag{4.11}$$

$$X^r = X^l \tag{4.12}$$

$$Y^r = Y^l \tag{4.13}$$

Analogously, the 2D projection $p^c = (u^c, v^c)$ of a 3D point $P^c = (X^c, Y^c, Z^c)$ onto the image space of camera c ($c \in \{l, r\}$) can be computed according to the projection model as:

$$u^c = u_0^c + \frac{F_u^c X^c}{d_u^c Z^c} \tag{4.14}$$

$$v^c = v_0^c + \frac{F_v^c Y^c}{d_v^c Z^c} \tag{4.15}$$

Considering equations 4.11 to 4.13, and assuming identical cameras –i.e., cameras with identical intrinsic parameters–, it should follow:

$$u^r = u^l - \frac{F_u^r b_{l,r}}{d_u^r Z^r} \tag{4.16}$$

$$v^r = v^l \tag{4.17}$$

In practice, equations 4.11 to 4.13, and equations 4.16 to 4.17 do rarely hold, due to errors in the real relative location between the cameras and differences in their actual intrinsic parameters. Better results, however, have been obtained –in terms of reconstruction followed by projection– ignoring the relationship ${}^{r}x_{l}$ given by equation 4.5 and performing the reconstruction and projection using equations 4.6 to 4.8, and 4.14 to 4.15 with the assumption of identical intrinsic parameters.

4.1.2 Model of camera configuration

This model comprises the set of parameters that describe the transformation between each of the cameras in the stereo head and the end-effector of the robot arm. In the configuration considered in this thesis, the stereo head is mounted on the end-effector, so there exists a fixed, rigid relationship between them. Let $\mathfrak{F}_c$ and $\mathfrak{F}_e$ the frames associated to camera c ($c \in \{l, r\}$) and the end-effector, respectively, and ${}^{c}T_{e}$ the matrix corresponding to the transformation from frame e to frame c. Following equation 4.1, the general form of this matrix is:

$$ {}^{c}T_{e} = \left[\begin{array}{cccc} \multicolumn{3}{c}{{}^{c}R_{e}} & {}^{c}t_{e} \\ 0 & 0 & 0 & 1 \end{array} \right] \tag{4.18} $$

with ${}^{c}t_{e} = \{{}^{c}t_{x_e}, {}^{c}t_{y_e}, {}^{c}t_{z_e}\}$ being the translation component of the transformation and ${}^{c}R_{e} = {}^{c}R_{c}({}^{c}\alpha_{e}, {}^{c}\beta_{e}, {}^{c}\gamma_{e})$ the corresponding rotation matrix, with $({}^{c}\alpha_{e}, {}^{c}\beta_{e}, {}^{c}\gamma_{e})$ being RPY angles. The general form of the manifold corresponding to this transformation is thus ${}^{c}x_{e} = \{{}^{c}t_{x_e}, {}^{c}t_{y_e}, {}^{c}t_{z_e}, {}^{c}\alpha_{e}, {}^{c}\beta_{e}, {}^{c}\gamma_{e}\}$.

In order to simplify the overall transformation between the coordinate frames associated to the end-effector and the cameras, it is assumed that the Z axis and the planes defined by the X and Y axes of all these frames are parallel. In this case, the RPY angles can be simplified as $({}^{c}\alpha_{e}, {}^{c}\beta_{e}, {}^{c}\gamma_{e}) = ({}^{c}\alpha_{e}, 0, 0)$, and the rotation matrix can be simplified to a rotation around the Z axis: ${}^{c}R_{e} = {}^{c}R_{z_e}({}^{c}\alpha_{e})$. The frame transformation can thus be expressed as:

$$ {}^{c}x_{e} = \{{}^{c}t_{x_e}, {}^{c}t_{y_e}, {}^{c}t_{z_e}, {}^{c}\alpha_{e}, 0, 0\} \tag{4.19} $$

$$ {}^{c}T_{e} = \left[\begin{array}{cccc} \multicolumn{3}{c}{{}^{c}R_{z_e}({}^{c}\alpha_{e})} & {}^{c}t_{e} \\ 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cccc} \cos {}^{c}\alpha_{e} & -\sin {}^{c}\alpha_{e} & 0 & {}^{c}t_{x_e} \\ \sin {}^{c}\alpha_{e} & \cos {}^{c}\alpha_{e} & 0 & {}^{c}t_{y_e} \\ 0 & 0 & 1 & {}^{c}t_{z_e} \\ 0 & 0 & 0 & 1 \end{array} \right] \tag{4.20} $$

Therefore, the model of the configuration is described by four parameters, ${}^{c}t_{x_e}$, ${}^{c}t_{y_e}$, ${}^{c}t_{z_e}$, and ${}^{c}\alpha_{e}$, the meaning of which is summarized in table 4.4. Following the convention considered for other parameters, the coordinate values corresponding to lengths are expressed in meters and those corresponding to angles, in radians.

4.2 Object selection

This preliminary processing corresponds to the *primary visual perception* filter described in section 3.2. This filter takes as input a gray-level image –in particular one of the images provided by the stereo head– and provides the description of a selected object. This description consists of a list of the external and internal contours of that object.

Table 4.4: Parameters of the model of camera configuration for each camera c ($c \in \{l, r\}$). Lengths are expressed in meters and angles in radians.

Parameter	Description
${}^{c}t_{x_e}$, ${}^{c}t_{y_e}$, ${}^{c}t_{z_e}$	Transformation components corresponding to the translation between the frame associated to camera c and the frame associated to the end-effector (e).
${}^{c}\alpha_e$	Transformation component corresponding to the rotation around axis Z between the frame associated to camera c and the frame associated to the end-effector (e).

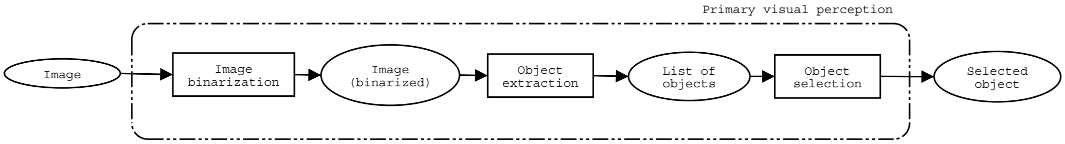


Figure 4.1: Structure of the primary image processing filter.

The processing performed by this filter is quite simple and comprises the following tasks: (1) *image binarization*, which performs a binarization based on the analysis and manipulation of the histogram that segments the objects from the background [93]; (2) *extraction of the contours of the objects*, with a *sampling* of the contour points, for data reduction [174], and (3) the selection of an object out of the set of detected objects based on a given criteria. As shown in figure 3.6, this processing is performed for each of the two images provided by the stereo pair of cameras. The internal structure of this filter is shown in figure 4.1.

Other more sophisticated approaches, considering the integration of different cues for the detection of the object, can be found in the literature [183].

4.2.1 Image binarization

The segmentation of objects from the background in the gray-level image corresponds to the *image binarization* filter shown in figure 4.1.

This filter takes as input a gray-level image and produces a binary image in which the pixels corresponding to objects and those corresponding to the background are assigned different values. This filter expects that objects can be easily distinguished in the images from the background based on their gray levels. In fact, it expects the images to be bimodal, with the image histogram showing a clear distinction between the gray levels corresponding to the objects and those associated to the background [93]. The procedure followed in this work for segmenting this type of images tries to find a *threshold* or gray level corresponding to the border between them.

Once the threshold has been found, the images are binarized. All image pixels with a gray level below this threshold as assigned the same level and another level is assigned to the other pixels. After this transformation has been performed, the image is said to have been binarized, since it is only defined by two gray-level values [170, 156].

4.2.2 Object extraction

This module corresponds to the *object extraction* filter shown in figure 4.1. This filter takes as input a binarized image and extracts a list of object descriptions, each description consisting of a set of contours corresponding to the external and internal borders of the shape of the object.

The extraction of these descriptions is performed for each image without using any a priori knowledge or data computed from previously processed images.The contours are first individually extracted from the binarized image. For this extraction, the image is scanned until a pixel belonging to an object is found. The border of the shape of this object is then followed in order to collect all the pixels that define it. Therefore, the contour is defined as a list of points.

However, the shape of an object can have not only an external contour but also internal contours corresponding to holes. Therefore, once all the contours have been obtained, an analysis is performed in order to group the internal and external contours extracted from the same object. Similar approaches for object extraction can be found in [171, 68]. Nevertheless, this capability of managing internal holes will not be the used within the grasp-execution task, since some of the filters it is composed of do not include support for handling internal contours.

In order to reduce the effect of noise introduced during the processing performed by this and previous filters, each extracted contour is smoothed using a Gaussian filter. Next, in order to have a more compact description, the contour is sampled using a technique known as *m-sampling* [157], so that only one out of every *m* points is kept and the others are removed. Iñesta [92] showed that with $m = 3$ or $m = 4$ the estimation of the length of the contour after the m-sampling is similar in many objects to the length computed considering all the points of the contour. Experiments in this thesis were performed with $m = 3$. Sections 4.3 and 4.4 analyze other options for a more efficient representation of the contour.

4.2.3 Object selection

As several objects can be extracted from an image, there is a need of an automatic selection procedure. This selection is performed by the *object selection* filter shown in figure 4.1. This filter takes as input a list of descriptions of selected objects and produces as output the selection of one of the objects in this list.

With no previous knowledge on the location of the object of interest, generic criteria have to be considered. The following ones have been used:

- *Size of the object.* The largest or the smallest object can be selected.

- *Distance to the reference points.* In this case, the selected object is the one the centroid of which is the closest to or the farest to a given reference point. During the development of the *object selection* filter for this thesis, the center of the image was used as reference point. Anyway, this criterion could be used in the future to introduce previous knowledge regarding the position of the object; this could be achieved by using the centroid of the object in another image –a previous image or the other one from a stereo pair– as the reference point, or the expected centroid of the object to select in the current image, for instance.

If required, a previous control action would position the robot arm and the vision system so that an appropriate view of the object to perform this selection is obtained. This selection could also be improved with the use of object tracking algorithms such as the one described in [183].

4.2.4 Object description

Each extracted object description consist of a list of one or more contours, corresponding to the borders of projected silhouette of the object in image space. One of these contours must be the external contour, and the others, if any, are associated to the internal contours of the object.

Each contour consists of an indexed list of points. In order to ease the calculus of geometric properties of the object or any further computation based on the list of contours, the convention that the list of contour points is ordered clockwise is followed.

4.3 Alternative representations of the contour: Polygonal approximations

4.3.1 Motivation

The problem of the efficient representation of a digital contour has received a considerable attention in the literature over the last years. In this section, polygonal approximations (PA) are used to provide a compact contour description of the contour that simplifies the representation of an object.

Curve segmentation is a technique that consists of grouping elements of a curve under a given criterion in order to obtain a compact representation of it. Two main analytical approaches have been proposed for this task: (1) an approximation using a polygon, subject to given constraints [146]; and (2) the location of a subset of high significance points in the contour, called *dominant points*, which suffice to characterize it [94, 125, 190, 164]. Nevertheless, whatever the approach, there is not a curve segmentation method that can be considered the best; the suitability of a given method strongly depends on its intended application.

The first approach is based on the study of the local properties of the curve in the vicinity of each point, while the second one is based on a global analysis of the curve and the polygon. In both cases, the goal is to maximize or minimize one or more factors, such as the length of each segment, the number of dominant points, or the error of the approximation. Therefore, polygonal approximation (PA) making can be regarded as an optimization problem.

The first algorithm for computing a PA of a given contour was proposed by Ramer in 1972 [158] and later by Duda and Hart [54]. This algorithm iteratively splits the contour into finer segments until a given error criterion was satisfied. Pavlidis and Horowitz [146] described a procedure for finding PAs based on splitting and merging segments until the error criterion was satisfied by each segment. More recently, Hu and Yan [87] have used the theory of perceptual organization for obtaining PAs attempting to match the human performance. Other approaches can be found in [165].

In this section, the computation of a PA is seen as an optimization problem, and the genetic algorithm (GA) is used as a tool for finding solutions according to several criteria [193, 159]. GAs are a technique that has become very widely used in many areas of research to solve search and optimization problems [45, 81]. In this approach, we consider the subset of contour points that are vertices of the PA as individuals in a population. Then the GA is used to find the individual that provides the best values for the target variables. Other works have also begun to explore this application of the GAs [89, 201], but they consider only one evaluation function, and tend to use problem-specific operators, which are avoided here for the sake of generality [108].

4.3.2 Polygonal approximation procedure

In most GA tasks there are only two elements that are problem-dependent: the encoding of the possible solutions as individuals in a population, and the definition of an evaluation function to assess each of these individuals. In addition, it is necessary to set the configuration parameters of the GA to the values that are most appropriate for the problem.

Problem representation

As it has been mentioned above, a contour is obtained as an ordered sequence of points. For the purposes of selecting the vertices of the PA for this contour using a GA, a codification has been defined. This codification consists of a binary string of the same length of the contour in which a `1` indicates that the corresponding point in the contour is selected as a vertex of the PA and `0` otherwise. The GA will perform a search in a population of strings representing sets of vertices. This representation of the solutions is independent of the particular codification used to represent the points in the contour.

In this work, the evaluation function used by the GA to compute the goodness of each set of vertices in a population is based on the computation of the error between the PAs and the original contours. The functions that have been considered for computing this error are:

1. Integral square error: $E_2 = \sum_{i=1}^{N} e_i^2$
2. Maximum error: $E_\infty = \max_{1 \leq i \leq N} e_i$
3. Optimization error: $E_\text{o} \; = \begin{cases} n_d/N & \text{if } E_2 = 0 \\ E_2 \cdot n_d/N^2 & \text{otherwise} \end{cases}$

where N is the number of points in the contour, n_d the number of vertices in the PA, and e_i the distance from each contour point to the closest segment of the approximation.

The maximum error evaluates how far the most distant point of the curve is to the polygon, and the integral square error evaluates how similar the curve and the approximation are. The optimization error [94] is a way to evaluate simultaneously the similarity to the original shape and the compression rate achieved, being a good measure of the quality of the approximation.

The evaluation function has been designed in order to guide the selection of the set of points according to one of the following goals:

1. Finding the PA with the minimum error

2. Finding the PA with the minimum error that contains no more than a given number of vertices

3. Finding the PA with the minimum number of vertices and the associated error of which does not exceed a given threshold

The evaluation functions used to achieve these goals are, respectively:

1. *error().* The error function itself.

2. *error_nd().* The error function plus a penalty if the number of vertices exceeds a given threshold, and a value inversely proportional to the number of vertices otherwise.

3. *nd_error().* The number of vertices plus a penalty if the error function exceeds a given threshold, and a value inversely proportional to the error otherwise.

The GA will be used to find a set of points that minimizes the selected evaluation function.

Configuration of the genetic algorithm

Four problem-specific parameters have been considered:

1. *The error function*

2. *The evaluation function*

3. *The threshold for the number of vertices.* It is the threshold used by the evaluation function (2) described above, and also the target number of points when the stopping error criterion is the achievement of a threshold value. This threshold is given as a fraction of the number of points in the original contour.

4. *The error threshold.* It is the threshold used by the evaluation function (3) described above, and also the target error when the stopping error criterion consists of the achievement of a threshold value.

The parameters related to the configuration of the GA cover aspects like the number of individuals in the population, the replacement policy, and the mechanism for the selection of the individuals that will have the chance to reproduce. The range of values assigned to each parameter includes those typically used in many reported applications. The selected parameters are:

1. *The number of individuals in the population*

2. *The number of generations*

3. *The stopping criterion.* Four criteria to decide when to stop the GA have been considered: (1) the number of generations; (2) evaluation stability (i.e., when the evaluation of the best solution has not changed after a number of generations); (3) population too similar (i.e., when a high percentage of the individuals in the population have the same evaluation value); (4) achievement of threshold values (i.e., when the variable to minimize through the evaluation function reaches a specified threshold and the goodness condition considered by that function —a maximum number of vertices or a maximum error— is met).

   In options (2) and (3), the values of the number of generations with no change in the best solution and the percentage of individuals with similar evaluation function are the defaults used by the *PGAPack* library [108].

The configuration of the GA is affected by many other parameters, such as the number of new individuals created each generation, the selection of individuals for crossover, the type of crossover, etc. An extensive review on these and other parameters can be found in [81].

One of the goals of testing the GA under different parameters configuration is to study which configurations lead to better point sets, and its ability to find a good solution under different configurations.

4.3.3 Experimental results

Test setup

The performance of the GA for making PAs was tested on a set of object shapes. It was also tested under different parameter configurations, based on the selection of parameters explained in section 4.3.2. The ranges of values used for the selected parameters and their default values are summarized in table 4.5. For other parameters only a default option has been considered [159].

Several error measurements (E_o, E_2 and E_∞) were computed in order to evaluate the results yielded by the GA, and the computation time for each test was also measured. Some tests were performed in which the algorithm is executed a number of times over each shape with the same parameter configuration in order to study the output stability in terms of standard deviation of the output values.

The whole set of tests was executed on a 300 MHz Pentium II running Windows NT 4.0. The program written to test the use of the GA is based on the *PGAPack* library [108].

Test results

As a first result, it is observed that the increase in the number of individuals leads to smaller errors, but the computation time increases linearly with both the number of individuals and the number of points in the original contour.

The more generations the algorithm spends refining the solutions, the smaller the obtained errors are. However, there seems to be an inflexion point (around the 400th generation), from which these error values are not significantly improved.

When the GA is used to minimize the error given a maximum allowed number of vertices, the algorithm shows the expected result that the error increases if the number of ver-

Table 4.5: Tested values for selected configuration parameters of the GA.

PARAMETER	RANGE OF VALUES	STEP	DEFAULT VALUE
Error function	E_2, E_∞, E_o	—	E_o
Evaluation function	*error(), error_nd(), nd_error()*	—	*error()*
Number of individuals in the population	[100, 1000]	100	500
Generation of the initial population	Random, Seed individual	—	Random
Fast algorithm for generating the seed individual	Non-zero curvature	—	Non-zero curvature
Mutation probability for the seed individual	[0.05, 0.9]	0.05	0.1
Number of generations	100 and [200, 2000]	200	100
Stopping criterion	No. of generations, Stability, Popul. too similar, Threshold values	—	No. of generations
Threshold for the number of vertices	N/2, N/3,..., N/10	—	N/5
Error threshold	(E_2) [0.1, 1] (E_∞) [0.5, 5] (E_o) [0.01, 0.1]	(E_2) 0.1 (E_∞) 0.5 (E_o) 0.01	0.05 0.05 0.05

tices is reduced. In addition, computation time does not suffer significant changes when the maximum allowed number of vertices is reduced. In general, better results are obtained when the error to minimize is the optimization error. Analogously, when the goal is to minimize the number of selected vertices given a maximum allowed error, this number decreases when the maximum allowed error increases.

Figure 4.2 shows the sets of vertices produced when the goal was to minimize the optimization error, and all other parameters were set to their default values.

The experiments to check the stability of these results revealed high deviations for the number of vertices (n_d) in the approximations produced by the GA when trying to minimize parameters other than n_d. Deviations were always low in the case in which the goal was to minimize the error given a maximum allowed number of vertices or minimize the number of vertices given a maximum error. Nevertheless, high deviations were observed for the maximum error when the goal was to minimize the number of vertices.

The results of the GA are comparable to those of some well-known curve segmentation methods [159]. The proposed method is more costly than these ones are for the selected shapes. However, the cost of this algorithm has shown to increase linearly with the number of points in the original contour, as opposite to the quadratic increment of dominant point extraction methods, so better results can be expected for larger contours.

4.3.4 Contour description

Polygonal approximations can be used to simplify the description of each contour associated to the list of contours that the description of an object consists of. The original list of contour points is reduced to a shorter list, which contains contour points that constitute the vertices of a polygon that approximates the original contour with a small error.

Therefore, a list of points can still be used to represent each contour after the computation of polygonal approximations. If needed, a more dense representation of the contour could be recovered by interpolating points along the segments that join the vertices of the approximating polygon.

4.3.5 Suitability for the grasp-execution task

Although results are quite good, the time required to obtain a polygonal approximation from a given contour is relatively high. In addition, for the purposes of modules that appear further in the filter chain, the polygonal approximation did not provide significant advantages over the m-sampling that could justify the increase of the computational cost. Therefore, this approach for a compact representation of the contour has not been considered in further steps.

4.4 Alternative representations of the contour: B-splines

4.4.1 Motivation

In addition to polygonal approximations, many geometrical, compact representations of contours have been proposed in the literature [67, 166, 184]. Another representation that has been considered for describing the contour are *B-splines* [20, 66, 67, 147]. Object tracking

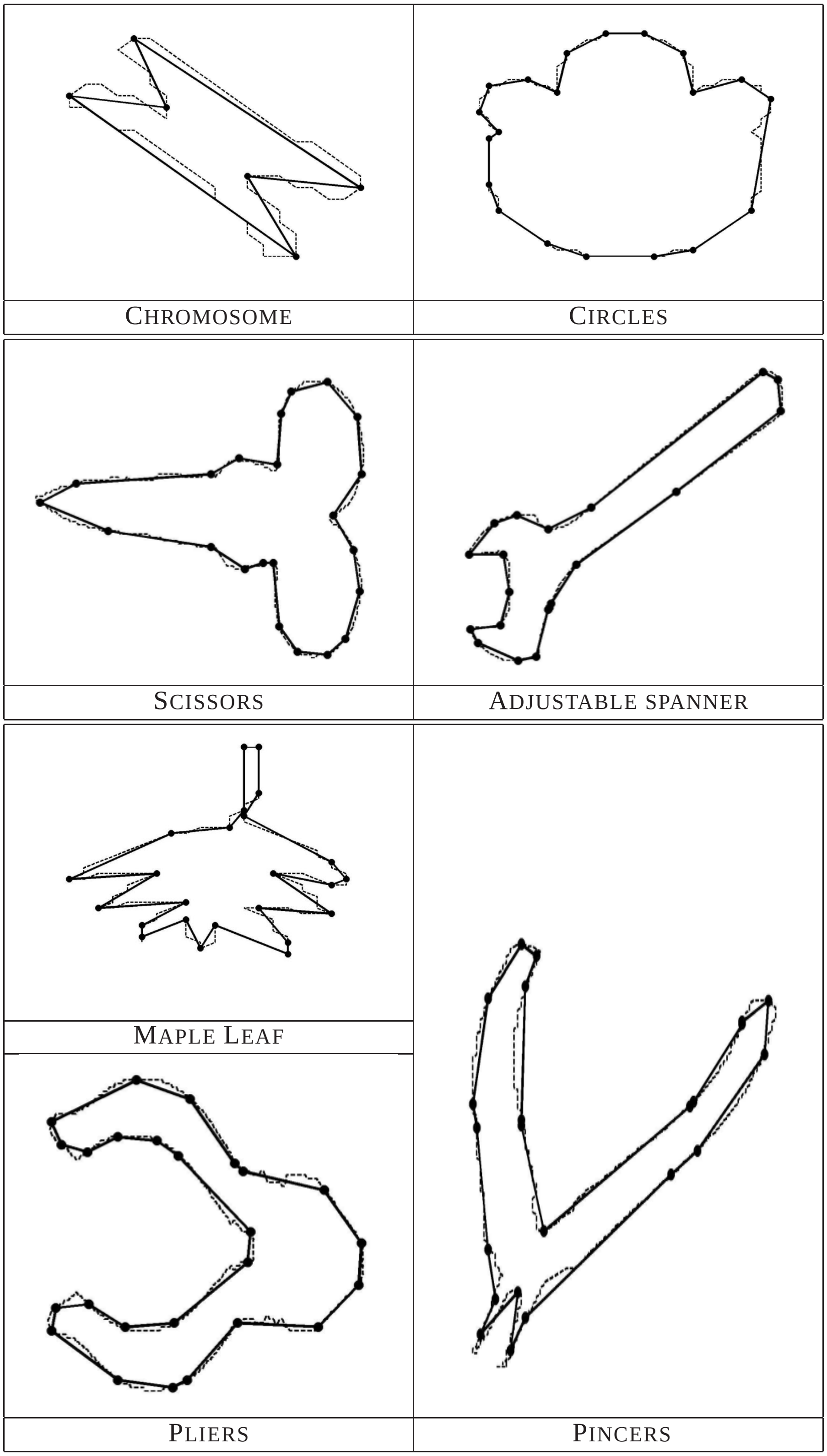


Figure 4.2: PAs produced by the GA when minimizing the number of vertices with a maximum $E_o = 0.05$

algorithms based on B-splines or similar curves can be found in the literature [20, 160]. *Active contours* (or *contour trackers*) have also been successfully applied to tasks such as grasp search [19, 160], visually-guided path planning for robot manipulation [21] or 3D reconstruction [126, 127]. This section describes a procedure for the automatic definition of a B-spline that approximates a reference contour, which was introduced in [67, 66].

B-splines –or simply *splines*– are a compact representation of the contour that is the basis of several methods for tracking and segmenting objects in a sequence of images, which may correspond to a cluttered scene. A B-spline is a parametric curve defined by a polygon of control points [13, 22]. The B-spline consist of a sequence of piecewise polynomial functions, where the curve pieces, *spans*, are smoothly connected curve points known as *knots*. The polynomial functions are weighted by the control points in order to produce the curve pieces.

B-splines are often initialized manually or using computer drawing tools. Nevertheless, this method does not produce accurate results when the spline has to approximate a contour –an object contour, for instance– given as reference. The automatic setup of a B-spline so that it approximates a given shape has been considered by many works [29, 20, 145]. Normally the described procedures perform first an initialization step, in which an initial B-spline is built, and then a refinement step, so that the number and position of the control points of the B-spline are modified in order to make the B-spline more accurately approximate the reference shape.

In [145] corners and points in flat sections are selected from the contour of the reference shape in order to determine the location of the control points, but no further refinement step of the B-spline is described. [20] performs an initialization that uses a geometrically-fixed first curve, taking into account only the size of the object. A refinement step is used then to improve the spline. In [71] the initial location of the control points is based on Duda and Hart’s [54] polygonal representation of the reference contour. Nevertheless, [29] warns that contour regions that are poorly modeled by a spline of a given order require a higher concentration of control points. Therefore, only the clustering of control points around high curvature regions is not necessarily the optimal procedure to fit a spline to a reference contour. In [29] the initial selection of the control points is based on an analysis of the reference contour. The refinement step tries an *error energy function* that measures the deviation between the spline and the reference contour. A more limited approach is introduced in [61], where the spline setup is based on the identification of shape patterns on the reference contour.

The procedure proposed in this section selects the control points in the initialization step based on the curvature of the reference contour. Taking into account the comments by [29] on high curvature regions and control points, the proposed selection of control points considers not only regions with high curvature, but also the intermediate contour regions that lie between them. The knots of the approximating spline are selected differently depending on the type of curvature in each region, and, from them, the control points are computed. The purpose of this step is to obtain a good first approximation that can be improved using simple methods in the refinement step. For simplicity, this procedure is used here to obtain an approximation of only the external contour of the reference shape. In order to consider also the internal contours, the same procedure would have to be individually repeated for each of them.

In [184] the curvature of the outline of the reference shape is also used to build a parametric approximating curve.

4.4.2 Procedure for the setup of an approximating B-spline

Initialization step

The goal of this step is to select a set of control points that builds a good first approximating B-spline. The selection of the control points is based on an analysis of the curvature of the reference contour [67].

The *k-cosinus* [163] is used here to compute the discrete curvature at each point of the reference contour. The *k-cosinus* at a point p_i from the contour is defined as the angle between vectors $\overrightarrow{p_ip_{i-k}}$ and $\overrightarrow{p_ip_{i+k}}$:

$$KCos(p_i) = \widehat{\overrightarrow{p_ip_{i-k}}\overrightarrow{p_ip_{i+k}}} \tag{4.21}$$

The parameter k is selected to be a percentage of the number of points in the contour. This parameter was experimentally set from contour lengths ranging from 300 to 800 points. The curvatures for all the contour points are stored in a vector known as the *curvature vector*.

The curvature vector is explored in order to extract a set of characteristic points that will be used to compute the control points of the first approximating spline. In particular, the following points have been considered:

- Convexities, detected as local minima in the curvature vector.
- Concavities, detected as local maxima in the curvature vector.
- Inflexion points, which are intermediate points with zero curvature between concavities and convexities.

The detection of these points is based on a simple analysis of the first and second derivatives of the curvature vector.

The above set of characteristic points is filtered in order to keep only the points that will be useful for computing the set of control points. This selection is based on the following criteria:

- Convex regions. All consecutive characteristic points corresponding to the convexity and two inflexion points that appear immediately at both sides of this convexity.
- Concave regions. Two situations are considered:
  - In the case of a single concavity point detected between two inflexion points, the concavity point is skipped, since the inflexion points provide enough information to compute the control points corresponding to this region.
  - In the case that several consecutive concavity points are detected, all of them are kept, together with the two inflexion points that appear at both sides of this concavity.

Next, a tangent to the contour at each of the above selected characteristic points is considered. The intersections between these tangents provide the set of control points of the first approximating B-spline. The selected characteristic points from which the control points have been computed will be thus the knots where the spans of this first B-spline meet.

Refinement step

The refinement of the B-spline implies computing the error between the B-spline and the reference contour and then modifying the set of control points in order to obtain a new spline that reduces the error [66, 67]. These two steps are iteratively repeated until a given *degree of accuracy E* is achieved. The *degree of accuracy* is the maximum allowed error between the points of the reference contour and the points belonging to the approximating B-spline.

The computation of the *approximation error* between the B-spline and the reference contour requires finding the correspondence between each point p_{s_i} sampled from the B-spline and a point p_{c_i} in the reference contour. The *approximation error* of the spline at point p_{s_i} will be computed as the Euclidean distance between points p_{s_i} and p_{c_i}. Point p_{c_i} is found as the intersection between the reference contour and the normal to the spline at point p_{s_i}.

Each point of the spline should have a corresponding point in the contour. Hence, if such a correspondence cannot be found for any point p_{s_i} of the spline, that is an indication of an error in the set of control points considered to build the spline. This error is corrected through an analysis of the set of control points, taking into account the fact that each subset of three consecutive control points influences a span of the spline. The following cases have been considered:

1. It is not possible to find a correspondence for the initial point of the span, or the normals at both extremes of the span intersect with each other before intersecting the contour. In this situation, the second control point in the sequence of three that influences this span is deleted.

2. It is not possible to find a correspondence for the last point of the span. In this situation, it is the third control point in the sequence of three that influences this span that is deleted.

The refinement of the B-spline is performed through the splitting of spans in which the maximum measured error between the span points and their corresponding contour points exceeds the selected *degree of accuracy*. The span is splitted into two new spans at the point where the error with respect to its correspondence in the contour is maximal. This correspondence in the contour is called the *break location*. The two new spans that replace the old one will join at the break location. This point can be therefore considered as a new knot that have been inserted in the span. This refinement of the spline through a knot-insertion strategy has also be considered in other works such as [29, 71, 49].

4.4.3 Experimental results

Figure 4.3 shows the approximating B-splines for a set of reference contours. The reference contours appear in the first row. The second row shows the first approximating B-splines, together with their corresponding reference contours.

4.4.4 Benchmark

In this work, a benchmark has been proposed to show the efficacy of the methods for generating approximating B-splines from reference contours. In this benchmark, the reference

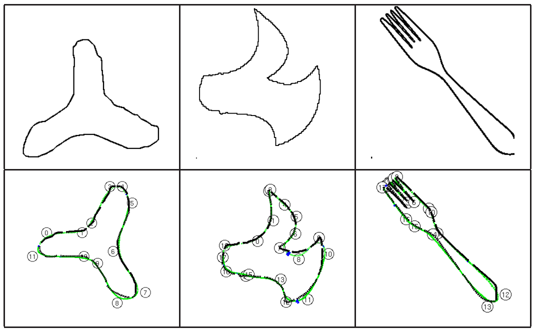


Figure 4.3: Reference contours and final approximating B-splines after the refinement step.

Table 4.6: Results of the proposed benchmark for the spline-setup methods.

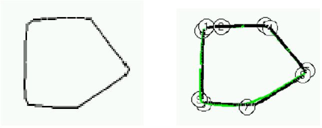
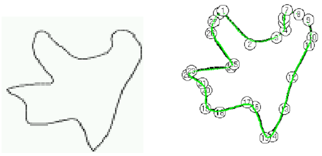
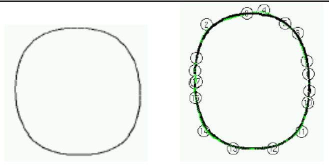
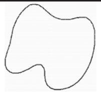
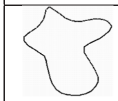
	No. of control points	Degree of accuracy	Number of iterations
	10/11	2	2
	25/29	3	3
	4/18	2	3

Table 4.7: Number of iterations, achieved accuracy and control points for different desired degrees of accuracy.

Contour	Desired accuracy	Number of iterations	Achieved accuracy	No. of control points
	2 3 5 7	4 4 4 4	2 2 2 3	27 22 21 19
	2 3 5 7	3 2 2 1	2 2 2 3	26 22 21 19

contour is a B-spline. An ideal B-spline initialization method would produce an exact copy of the B-spline given as reference. However, the original and the generated B-splines can be compared according to several parameters. Here, two parameters have been considered: (1) the number of control points, and (2) the approximation error.

Table 4.6 shows the results of applying this benchmark with some reference B-spline to the method described in this section. The number of iterations required in the refinement step to achieve an approximation with the desired degree of accuracy is indicated. For the experiments corresponding to this table, a degree of accuracy of 3 pixels was considered.

Table 4.7 shows the number of control points of the B-splines produced by the refinement step for different desired degrees of accuracy [66]. The degree of accuracy is here measured in pixels, too. A decrease in the desired accuracy implies not only using more control points, but also performing more iterations in the refinement step in order to cope with the complexity of the input contour. As mentioned in [29], with excessive increase in the number of control points it would be recommendable to increase the order of the spline. Nevertheless, the goal of the proposed refinement procedure was to reduce the error between the spline and the reference contour, not taking into account the number of control points. However, in some situations, control points that are close to each other could be substituted by a single one having an increased multiplicity with no significant increase in the error. This could be used in a post-refinement step to reduce the number of control points.

Note that the desired accuracy can be achieved with only a few iterations in the refinement step, due to the quality of the approximation obtained in the initialization step.

4.4.5 Contour description

In this case, each contour is described as a B-spline. As it has been explained in previous sections, this type of curve is defined in the 2D case by a set of basis functions and a vector of 2D weights or control points. With a fixed or known configuration of the set of basis functions, the spline can be uniquely defined by the control points. Nevertheless, in order to have a contour description similar to the one considered in previous sections, based on a list of contour points, the spline is sampled using a normalized parameter. Therefore, after the sampling of the spline, the contour can be represented as a list of points.

Due to the definition of the spline, the sampled points are regularly distributed along the contour in the parameter space of the spline, although not in the 2D image space. In addition, since the parameter used for sampling is normalized, the distribution of the sampled points in parameter space is invariant with respect to the transformations applied to the B-spline, as long as the set of basis functions is not modified and these transformations produce a rigid movement of the set of control points. The transformations complying with this requirement are: translations and rotations on the image plane, and changes of scale. Therefore, the sampling of the B-spline points is invariant, in the parameter space of the B-spline, with respect to the above transformations. This can be used in further computations when an invariant representation with respect to the contour is required.

Depending of the sampling resolution, the list of sampled points can be considered as a polygonal approximation of the contour. Like in the case of contour description after polygonal approximations (see section 4.3.4), a more dense representation of the contour could be recovered from this list of sampled points by interpolating points along the segments that join them.

4.4.6 Suitability for the grasp-execution task

B-splines provide a good framework for tracking the shape of an object along a sequence of images [20]. They have been used in [160] for tracking and object along a sequence of images and tracking grasp points at the same time over the object. They will be therefore used as a base representation of the object in further modules in the filter chain considered in this work (see section 5.4).

4.5 Generic object description

As mentioned in previous section, a generic object description has been considered in this thesis consisting of a list of one or more contours. For each contour, the following information is required:

- An indication of whether the contour is an internal or external one.
- A description of the contour.

Although several options have been studied for the internal representation of the contour, it has been generally considered as a list of points. Each point is described through its (u, v) coordinates in image space. Depending on whether the list of points is considered as the complete set of contour points (see section 4.2.4), a polygonal approximation (see section 4.3.4), or a list of sampled points (see section 4.4.5), several possibilities for indexing the contour points within the contour could be considered. In this thesis, the following ones are internally maintained, together with the list of (u, v) coordinates, as part of the description of the contour:

- **Location coordinates.** The location coordinates correspond to indexes that are associated to the complete list of contour points. The indexes included in these coordinates for each point are:
  - *C*. Index of the contour, within the list of contours of the object, that the point belongs to.
  - *P*. Index of the point, within the complete list of the points belonging to contour *C*, corresponding to the specified point.
- **RRP coordinates.** They are used when the contour is described as a polygon, the vertices of which can be either those of a polygonal approximation or the points in a list of sampled points. These vertices have been named *reference points*. The contour points that lie on the segment joining the reference points are named *interpolated points*. The indexes included in these coordinates are:
  - *C*. Taken from the description based on location coordinates.
  - *Ref*. Index of the reference point, within the list of reference points corresponding to contour *C*, that is used as a base position to locate the specified point.
  - *Interp*. Index of the point, interpolated between reference point *Ref* and the following reference point, that corresponds to the specified point.

  The reference points can be considered as a reference frame within the contour with respect to which the location of the interpolated points can be expressed.

Either when the list of contour points is considered as a complete list or a polygon, it is possible to compute any of the above indexing coordinates from the other, so both can be maintained simultaneously. Depending on the intended use of the object description, the most suitable ones could be selected.

Chapter 5

Advanced visual perception: Search and track of grasps

This section describes first the procedure for selecting a new grasp on an object description. The strategy for grasp tracking is explained next.

5.1 General overview

The *primary visual perception* filter described in chapter 4 has isolated an object in each image and extracted its contour. The next step in the filter chain defined within the behavior is to extract, or to translate from another observed object, a pair of *grasp points* on the silhouette of the object. The *grasp points* will be the points of contact between the gripper fingers and the object when the object is grasped [32, 149, 185]. They have also been referred to as *grasping points* [99, 172] or, more often, *contact points* [15, 44, 154] in other works. These points will be used by the control law to guide the robot arm towards this object. These two grasp points define a *grasp segment* and a *grasp line*, which will also be used as a description of the grasp.

As shown in figure 3.6, this task is performed by the *grasp search and tracking* and the *external grasp tracking* filters. The former tries to track on the current observation of an object a grasp computed on a previous observation of that object. This filter is used in the proposed behavior to track a grasp along a sequence of images. When there is not a previous grasp available, a new one is computed on the current object. The *external grasp tracking* filter performs the translation of the grasp from an external object –i.e. extracted from another image– to the current one. Within the proposed behavior, it is used to translate a grasp computed in one of the images of the stereo pair to the other image.

Like the object extraction, the algorithms for the grasp search, translation and tracking have been designed so that they can be applied to objects with holes –that is, with internal contours, in addition to the external one. The description of the object is based on a list of points. Therefore, the filters for the search and tracking of the grasp points use this type of list as one of its inputs.

The *primary visual perception* filter, as described in section 4.2, provides object descriptions that include the internal contours corresponding to the holes of the object. Nevertheless, both the object extraction and the object selection are executed without taking into account any information extracted from previous images. Consequently, no object tracking is performed by this filter. However, some experiments have been performed using the representation of the contour based on B-splines described in section 4.4. In these experiments, a B-spline has been used as an active contour to track the position of the object of interest along a sequence of input images. Nevertheless, this B-spline has been associated to the external contour of the object and no tracking has been considered of the contours corresponding to internal holes. For this reason, and in order to take into account not only the *primary visual perception* filter, but also these experiments, the global functionality of searching and tracking grasps has only been tested on external contours. Off-line grasp search experiments, however, have been performed on objects with internal contours.

Anyway, as the grasp points are object-related features, the grasp translation and tracking is based on their location with respect to the object, instead of with respect to the image. This tracking does not rely on any specific procedure for the search of a new grasp or the evaluation of a previous one on the current object. Therefore, grasp search algorithms other than the one proposed in this chapter can be used here, as long as they provide a pair of grasp points.

5.2 Grasp search

The grasp search is performed by the *grasp search and tracking* filter when no previous grasp is available or all previous grasps have been discarded. The grasp search module includes the definition of a procedure to evaluate the grasp stability of a pair of points that are proposed as grasp points. This procedure is used by the *grasp search and tracking* filter to check that the tracked grasp points still correspond to a stable grasp.

This search is associated to the concept of *grasp synthesis*, which, according to Shimoga [181], is the “determination of the required finger properties in order for the grasp to acquire some desired properties”. The grasp evaluation is related to the concept of *grasp analysis*, which this author defines as “the study of grasp properties for a given set of finger properties”.

The input to the search and evaluation procedures is an object description consisting of one external and zero or several internal contours. This description is given in 2D space –in particular, in image space–, since the above procedures have been designed to deal with 2D grasps. This description is made available through one of the inputs of the *grasp search and tracking* filter. In addition, the evaluation procedure also requires as input a pair of points that are proposed as grasp points. Each of these points must belong to one of the contours of the object. The output of the search procedure is the description of the selected grasp, while the evaluation procedure produces a qualitative evaluation of the quality as a grasp of the pair of points given as input.

The presented strategy is based on previous works [132, 130, 172, 175]. It takes into account the whole morphology of the object, including not only the external contour but also the internal ones. In addition, it is able to find not only *squeezing grasps*, which are executed by closing the fingers on the object, but also *expansion grasps*, performed with an opening of the fingers [32, 130]. The grasps produced are *fingertip grasps*, in which each finger of the robot hand ideally contacts with the object at a point. *Enveloping grasps*, which are formed by wrapping the fingers around the object and are found in some works on 3D grasping [18], are not taken into account.

Initially, as in [132, 130], all the grasps that comply with a given set of stability criteria are found. Then, they are sorted according to a single evaluation value that depends on the stability criteria, so that eventually one grasp –the one with the best evaluation– can be selected. The outcome of the grasp search will be thus a pair of points, each one belonging to one of the contours of the object, at which the object can be stably grasped.

5.2.1 Characterization of the grasp stability

The stability of a grasp depends on the forces exerted by the gripper fingers on the object, and also on other external forces, including the gravity. The number and types of the forces involved depends on how the contacts between the object and the robot hand are modeled [18]. A number of conditions have been defined with respect to the stability of a grasp.

An important concept regarding the stability of a grasp is the *force closure* [137]. A grasp is said to have force closure if any set of forces and torques exerted over the object can be balanced by the contact forces exerted by the fingers. Nguyen [137] and Faverjon et al. [56] characterized two-finger, force-closure grasps by the fact that the line joining the contact points must lie within the friction cones at the contact points. Grasps that comply with this condition are also referred to as *antipodal point grasps* –or simply *antipodal grasps*– and the contact points as *antipodal grasp points* [32, 97].

The *non-sliding condition* [122] is aimed at ensuring that the object will not slide between the fingers of the gripper when being grasped. This condition takes into account the coefficient of friction between the object and the fingers and is based on the friction model of Coulomb [137]. According to this model, the friction force f^r due to the contact between two bodies is tangent to the surface of contact and is proportionally related by a constant μ, the *coefficient of friction*, to the normal f^n of the applied forces f^a. The *non-sliding condition* was defined as

$$f^t \leq \mu f^n \tag{5.1}$$

and states that if the resultant f^t of tangential forces do not exceed that of the friction force –or normal force times μ– exerted by one object over the other, there will be no sliding between them. The set of force vectors f^a that meet this condition defines a cone of forces in space that is known as the *friction cone*.

The non-sliding condition and the force closure can be considered included in the concept of *spacial grasp stability* [128] (*object stability* in [136, 95]), which is related to the tendency of an object to return to its original equilibrium position after an external force has been applied. This stability is related to the forces that appear on the whole object when the fingers of the robot gripper are placed on it. The concept of *contact grasp stability* [128, 136, 95] makes reference to the ability to maintain contact when the object is subjected to perturbation forces and is related to the tensions that appear on the surfaces of contact between the object and the fingers. The main factors that have an effect on the contact stability are the shape of the surfaces at the contact points and the distance between the contact points. The lower the curvature at the contact points and the closer these points are from each other, the greater the stability.

Nevertheless, the search for stable grasps on the object based on force analysis often requires solving complex operations. For this reason, many works base their search on the analysis of the geometric properties of the object. Vision-based works usually perform a geometric analysis based on the shape of the object. Typically, a number of measures of grasp stability are defined and a quality value is computed from them [130, 132, 149, 77, 172]. These methods are heuristic in nature, since they estimate generic values of some parameters related to the force closure (friction, force to apply to the object,...) based on either vision or previous experiments. According to [119], a minimum of three fingers are required to guarantee force closure in the plane, for a piecewise smooth curve, while Chen and Burdick [32] state that two fingers are sufficient for a smooth curve.

The analysis of the stability of a grasp performed in this thesis is based on geometric criteria defined in previous works [130, 132, 172]. No previous knowledge of the object or any of its physical parameters is assumed and only visually-extracted information is used. The effect of required physical parameters, such as the coefficient of friction, is experimentally estimated from visual observations. This allows to efficiently estimate the grasp stability and handle unknown objects.

5.2.2 Object descriptions for grasp search

For simplicity, many works on grasp stability, specially in the case of fingertip grasps have restricted their scope to those situations in which a 2D representation of the object, instead of 3D, can be considered as accurate enough [18]. A contour-based representation has been therefore considered by most of the works on the search of 2D grasps, though often it is not extracted from an image [56]. The contour may be represented as a list of points [130, 172], as a polygon [137], or as a parametric curve, either associated to an object in an image [149, 19] or being a pre-defined model [154, 56]. Nevertheless, internal contours –which correspond to visible holes in the silhouette of the object– are considered for the grasp search only by a few works [130, 172].

Early works such as [128, 56, 119, 122] on grasp search were based on the use of predefined object models. Faverjon and Ponce [56], for instance, modeled the boundary of a planar object through a piecewise collection of parametric curves. The main drawback of using this approach in a complete system is the impossibility to handle unknown objects. In addition, even if an unmodeled object could be recognized, the natural variations in shape can render useless any precomputed grasp. Partially known or unmodeled objects need grasp determination based on vision.

In other works [130, 149, 172], the contours have been extracted from vision data. Stanley [185] performed a quad-tree resolution expansion of the image in order to extract the contour. Among vision-based works that have not used the object contour, Jarvis [96] and Hauck [77] based their methods on the analysis of the skeleton. In [96] the distance to the outline of the object was encoded in the skeleton. Some works [149, 19] have considered a description based on deformable contours [20] to carry on the grasp search. Taylor [189] also used them to perform a 3D object exploration as part of the grasp search. In general, an advantage of these methods is that they do not require a previous identification of the object [186], since they are not based on models.

In this thesis, as explained in section 4.2.4, the object is described through one external and zero or more internal contours, which correspond to holes in the silhouette of the object. Each contour consists of an indexed list of points.

5.2.3 Strategies for grasp search

The force-closure condition has been used in many important papers on planar grasp determination. Nevertheless, the approach followed by many of them is based on the use of analytical optimization methods that require long computation time.

Faverjon and Ponce [56] search for force-closure grasps assuming hard point contacts between the object and the fingers. This work provides a complete solution, since it is able to find all grasps. A set of grasps is computed in the configuration space of a two-fingered gripper for a given coefficient of friction. Nevertheless, this solution is computationally expensive and does not consider expanding grasps. Ponce et al. [154] perform the grasp search for two-fingered grippers. Object boundaries are represented by collections of polynomial parametric curves, known in advance, and force-closure grasps are characterized by systems of polynomial constraints in the parameters of these curves.

Blake [19] maps also force closure grasps in the configuration space of a two-finger gripper. The object description is an active contour that has been used to track an object along a sequence of images. Optimal finger positions are obtained from subsets in the configuration space of the gripper and are those that require the least friction force for force-closure. No prior knowledge of the coefficient of friction is required. This work makes use of the local symmetry of the object and models fingers as thin, frictional cylinders.

Another procedure for the search of force-closure grasps is proposed by Chen and Burdick [32], who consider not only squeezing but also expansion grasps. This procedure is based on the definition of a *grasping energy function*, which is proportional to the distance between the grasp points. The selected grasp, or *maximal grasp* is the grasp that features the maximal distance between the grasp points. Both 2D and 3D object representations –splines and spherical product surfaces– have been considered in this work.

Following a different approach, the procedures presented in works such as those of Jarvis [96], Hauck [77] or Stanley [185] are not based on the application of optimization techniques to an analytic criterion. Instead, they perform a search over the description of the object in order to find the best grasp according to a set of conditions. Analogously, Kamon [99] uses visually-computed thresholds to select the grasp points on the contour of the object. Sanz [172, 174] associates thresholds to a set of stability criteria, and relies on a heuristic to find the best grasp. Morales [130, 132] finds a set of grasps that comply with these criteria and then computes a global quality value for each grasp that is used to select the best one.

Another type of grasp, referred to as *cage grasp*, was considered by Rimon et al. [161]. The *caging problem* is defined as the problem of surrounding an object with a multi-fingered hand such that the object has some freedom to move but still cannot scape the “cage” formed by the fingers [17, 179]. Another approach for performing the selection of a grasp, proposed in the works of Bekey [14] and Cutkosky [43], makes use of *grasping rules*. In these works, the grasping rules are based on studies on how humans grasp and manipulate objects.

Among the works on learning in 2D grasping, Moussa [134] defines *generic grasping functions*, which are gripper-object relationships that represent a gripper successfully grasping an object. A procedure is used to learn and select between available grasping functions. Other works on learning include [99].

The grasp search considered in this thesis follows previous works [132, 130, 172, 174]. A list of grasps is produced based on a set of stability criteria. A global quality function is used to evaluate all these grasps and select the best one.

Finally, general reviews on grasp search and characterization can be found in the works of Bicchi [18] and Shimoga [181].

5.2.4 Universe of graspable objects

In general, it can be said that a set of contours obtained from a single image is an incomplete description of an object, which is intrinsically a three-dimensional entity, since it is just a description of its projected silhouette onto the bidimensional image plane. In addition, since, according to the approach followed in this thesis, the observed objects are not known beforehand, no additional knowledge can be assumed other than this projection. Therefore some conditions have to be imposed to the set of valid objects on which it will be possible to perform a search for grasp points. These conditions are aimed at ensuring that the physical object properties that are considered during the grasp search can be correctly estimated through the 2D visual information available, that is, the set of contours corresponding to the projection of the object.

In particular, the following conditions have been considered:

- As the proposed grasp-search procedure requires an estimation of the center of mass of the object, valid objects are those in which this center of mass can be estimated through the *centroid* or geometric center of the projected silhouette.

- Valid objects are assumed to be relatively flat, or that at least they can be treated as an extrusion of the silhouette delimited by the extracted contours.

- They are lying on a flat surface that is parallel to the image plane of each camera.

- They are assumed to have a uniform distribution of mass and no hidden holes or concavities.

- They are rigid, so that they are not deformed by the gripper when being grasped.

- They have the appropriate dimensions to be graspable by the robot gripper.

Although the above conditions are not strict, the effectiveness of the grasp-search procedure –and, in general, of the whole grasp execution– will be related to the degree in which they are met. Despite these restrictions, the set of valid objects is quite ample. Figure 5.1 shows some examples of objects that have been used in the experiments, together with the contours extracted from them.

5.2.5 Generic grasp description

A generic description of a grasp has been considered in this thesis that internally maintains several specific descriptions:

- Location coordinates of the grasp points

- RRP (Relative Reference Points) coordinates of the grasp points

- Relative location of the grasp line with respect to second-order moments

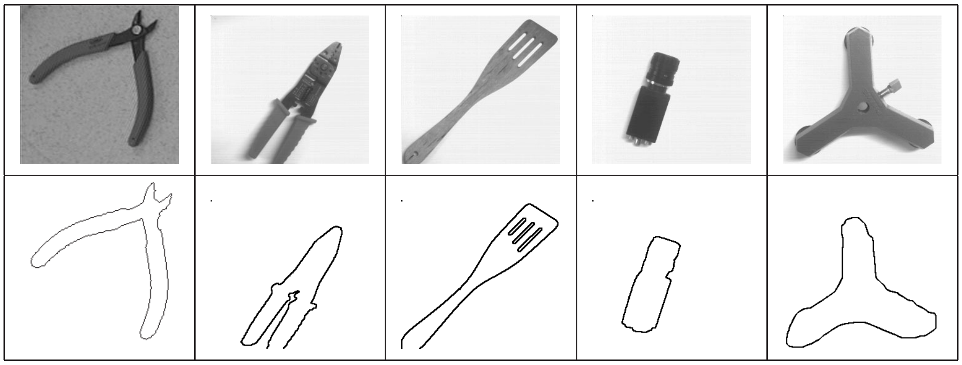


Figure 5.1: Examples of objects, with their corresponding contours that meet the restrictions imposed by grasp-search algorithm.

When any of them changes, the others are automatically updated, so that all internal descriptions are coherent and represent the same grasp. Anyway, the use of a generic grasp representation encapsulating a number of specific representations, allows to use the most convenient one depending of the processing to perform with the grasp.

In addition to the components of each specific grasp description, the following common data are also included in the description:

- The type of grasp. It is one of the following:
  - *Squeezing grasp.* Grasp that is executed by closing the gripper fingers around the object.
  - *Expansion grasp.* Grasp that is performed with an opening of the gripper fingers.

Moreover, any grasp will be associated to the following data:

- A description of the object the grasp has been computed for.
- A description of the gripper that has been considered when computing the grasp.

Location coordinates of the grasp points

It is composed of the following data:

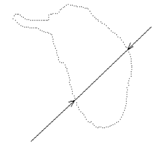
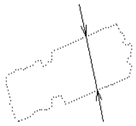
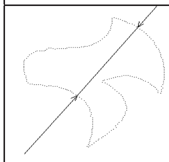
- *C1, P1.* Location coordinates *C* and *P*, respectively, of grasp point 1.
- *C2, P2.* Location coordinates *C* and *P*, respectively, of grasp point 2.

RRP coordinates

This description includes the following data:

- *C1*, *Ref1*, *Interp1*. RRP coordinates *C*, *Ref* and *Interp*, respectively, of grasp point 1.
- *C2*, *Ref2*, *Interp2*. RRP coordinates *C*, *Ref* and *Interp*, respectively, of grasp point 2.

Table 5.1: Grasp descriptions expressed in location, RRP and relative location coordinates.

Grasp	Location	RRP	Relative
	C1: 0 P1: 53 C2: 0 P2: 113	C1: 0, Ref1: 156 Interp1: 1 C2: 0, Ref2: 332 Interp2: 1	distC: 0.00164534 angleWImin: -87.1873° distGP1: -0.00345828 distGP2: 0.00384211
	C1: 0 P1: 48 C2: 0 P2: 95	C1: 0, Ref1: 141 Interp1: 1 C2: 0, Ref2: 276 Interp2: 1	distC: -0.00396697 angleWImin: -267.57° distGP1: -0.00375348 distGP2: 0.00385768
	C1: 0 P1: 38 C2: 0 P2: 181	C1: 0, Ref1: 111 Interp1: 1 C2: 0, Ref2: 535 Interp2: 1	distC: 0.000206447 angleWImin: -210.485° distGP1: -0.00254343 distGP2: 0.00507176

Relative location of the grasp line with respect to second-order moments

The grasp description is in this case based on the specification of the relative location of the grasp points and the grasp line with respect to an object-centered coordinate system. In particular, the minimum inertia and maximum inertia axes [187] of the silhouette of the object have been selected as the axes coordinate system, and the centroid as its center. These features coincide with the major, minor, and center of the best-fit ellipse, respectively.

These features can be obtained from the normalized geometric and central moments up to order two of the silhouette. In [187] a method is provided to compute these values from a list of points corresponding to the external contour of an object with no internal contours. A method known as the *delta rule* to compute the geometric moments that considers all the image points belonging to the object silhouette is introduced in [204] and extended as the *extended delta rule* in [173] to take into account silhouettes with internal holes. The central moments are computed from the geometric moments [68]. The central moments, which provide the values for the orientation of the axes of the best-fit ellipse, are only invariant to translations. The normalized central moments, normalized with respect to the area of the object, are also invariant to changes of scale.

Therefore, the set consisting of the centroid and the axes of this ellipse defines a coordinate system that moves rigidly with the object under translations and rotations on the image plane and changes of scale. It is thus an object feature that has a geometric meaning and is invariant with respect to four degrees of freedom with respect to which the grasp points can be located.

This description uses the following data:

- *distC.* Normalized distance, from the centroid of the object along the direction vector of the minimum inertia axis of the object (Imin), of the intersection between the grasp line and Imin. This distance has been normalized with respect to the area of the object.

- *angleWImin.* Angle between the grasp line and the minimum inertia axis, measured from the minimum inertia axis towards the grasp line. This angle has been considered in degrees in this thesis.

- *distGP1.* Normalized distance, along the direction vector of the grasp line, between grasp point 1 and the intersection between the grasp line and the Imin axis. This distance has been computed in pixel coordinate space and normalized with respect to the area of the object.

- *distGP2.* The equivalent to *distGP1* for grasp point 2.

Table 5.1 shows several grasp descriptions expressed in location, RRP and relative location coordinates.

5.2.6 Grasp evaluation

Each pair of points that are selected as candidates to grasp points will have to meet several criteria to be considered a valid grasp [130, 172, 175]. Concretely, two criteria have been considered to ensure contact stability [171]:

- *Gripper-adaptation criterion.* It evaluates the adaptation between the object and the gripper fingers in terms of the estimated area of contact when the fingers are placed on the object in order to execute the grasp. This estimation is based on the curvature of the observed contour of the object at the points where the fingers should be placed. Two curvature values α_1 and α_2, each one associated to one of the two grasp points, are computed. These values should not exceed a threshold α (*curvature threshold*).

- *Force-closure criterion.* Its purpose is to ensure that the gripper does not cause the object to slide due to a torque when it closes its fingers to grasp it. Its evaluation is based on the concept of *friction cone* [137] and the *non-sliding condition* [122]. It makes use of the angles β_1 and β_2 (see figure 5.2) between the normals N_1 and N_2 to the contour of the object at the grasp points and the *grasp line* ($\overline{P_1P_2}$). Force closure [137] is achieved when the grasp line lies inside both friction cones. These angles should not exceed a threshold β (*angular threshold*).

Additionally, the distance between the contact points should also be taken into account, since it affects the contact stability. Moreover, it can determine the feasibility of the grasp, since two-fingered grippers have a minimum and a maximum opening distance. This distance has been considered in [172, 171], although only as a feasibility factor, not as an evaluation factor.

In addition to the two above evaluation criterion, Sanz [172, 175] considers a third one that takes into account the effect of an external force such as the gravity:

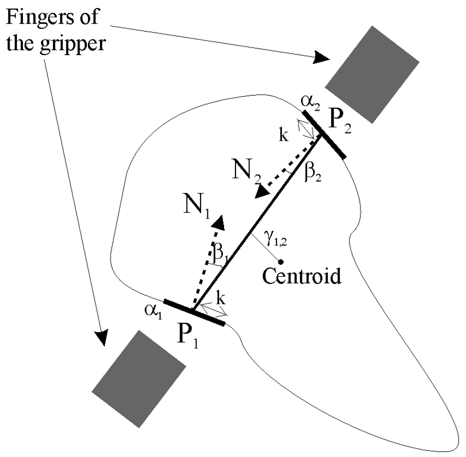


Figure 5.2: Geometric interpretation of the quality criteria used for the selection of grasps.

- *Distance-to-centroid criterion.* Its purpose is to ensure that the object will not be significantly affected by its own weight once it is grasped. This will happen when the *grasp line* –the line joining the contact points– is close enough to the centroid, which is assumed to be an estimation of the center of mass of the object. This distance will be referred to as $\gamma_{1,2}$ and should not exceed a threshold γ (*distance threshold*).

This criterion is used in [132, 130, 160] to compute a global quality value for each grasp that complies with the contact stability condition. Figure 5.2 illustrates how the above criteria are used in the evaluation of the quality of a grasp.

The values of α, β and γ have been determined empirically [171]. The experiments have shown that the relaxation in the value of γ has greater effects on the grasp stability than that in α and β. In fact, the effect of gravity on the object may have little relevance in several situations, such as in the case of light objects, when the fingers can exert forces of enough intensity over the object, or simply in the absence of gravity.

5.2.7 Grasp search algorithm

The proposed algorithm assumes contacts with friction between the object and the fingers, and considers the Coulomb friction model [137]. The static coefficient of friction between the object and the fingers is not known beforehand. The only considered *a priori* information is the geometry of the gripper –in particular, the width of its fingers.

Algorithm 5.1 outlines the procedure for the grasp search [130, 160]. First, a selection of grasp points over the contours of the object that have contact stability is performed. This initial search is based on the gripper-adaptation and the force-closure criteria mentioned in section 5.2.6. The selected grasps are then sorted according to the evaluation value that will take also into account the distance criterion, and the grasp with the best evaluation will be selected.

Algorithm 5.1 Grasp search

```
Extract a list of grasp regions
Generate a list of pairs of compatible regions
for each pair of compatible regions do
    Select best pair of grasp points
end for
Evaluate each selected grasp using a set of quality parameters
Sort grasps using this evaluation and select the best one
```

Step 1: Extraction of grasp regions

The purpose of this step is to find regions on the object contour that comply with the curvature threshold, which will be referred to hereinafter as *grasp regions*. In this step, a *curvature function* is computed at each point on the contour [174]. For the calculus of this function, a neighborhood centered at each point is considered that is of the same size as the projection of the width of the fingers of the robot on the image ($2k$ in figure 5.2). In this work, the *angular k-torsion* [152, 153] has been the function considered to evaluate the discrete curvature of the contour. The *k-torsion* at a point p_i of the contour is defined as the angle between two vectors $\overrightarrow{p_ip_a}$ and $\overrightarrow{p_ip_b}$:

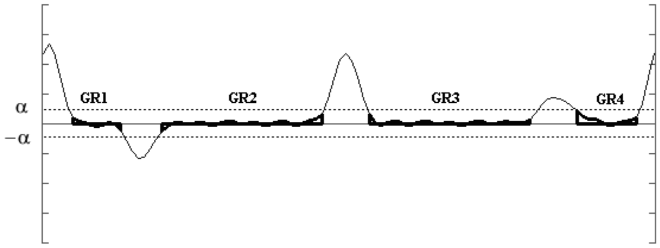
$$KTors(p_i) = \widehat{\overrightarrow{p_ip_a}\overrightarrow{p_ip_b}} \tag{5.2}$$

where p_a and p_b are the average of the k points that precede and follow p_i, respectively, in the list of points the contour consists of. Other curvature functions, however, such as the *k-cosinus* [163] (see section 4.4.2) or the *k-curvature* [69] are also valid.

The task of finding the grasp regions lies in the analysis of the curvature function and grouping in a single region consecutive points with curvature below the curvature threshold α defined in section 5.2.6, as shown in figure 5.3. Each point belonging to a grasp region is the center of a neighborhood with relatively low curvature at which one of the robot fingers could be placed for executing a grasp. Therefore, grasp regions are a grouping of consecutive points at which any of the robot fingers could be positioned.

Ideally, it should be possible to approximate these regions as straight lines. Nevertheless, in some objects, such as the one shown in figure 5.4, this procedure could produce large regions, covering portions of the contour that could not be reduced to straight lines. For this reason, the accumulated curvature along a grasp region is used to limit its size. The difference between the curvature at each point and the threshold α is accumulated as each point is added to a grasp region GR_j. When the accumulated difference reaches a given threshold, the grasp region GR_j is considered finished and a new one GR_{j+1} is started. The size of the grasp regions is also lower-bounded, since, being too small, they would not allow for tolerances in the positioning of the fingers of the robot.

Unlike in other works [96, 99, 172], the contours corresponding to the internal holes of the object have also been considered for the grasp determination. Therefore, when the internal contours of an object are available, the grasp regions are extracted from both the external and the internal contours.



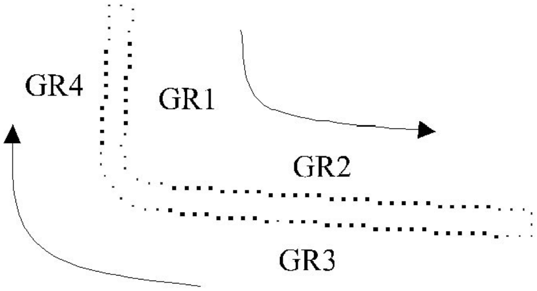


Figure 5.3: Grasp regions on the contour of an Allen wrench (below), selected based on the analysis of the curvature function (above).

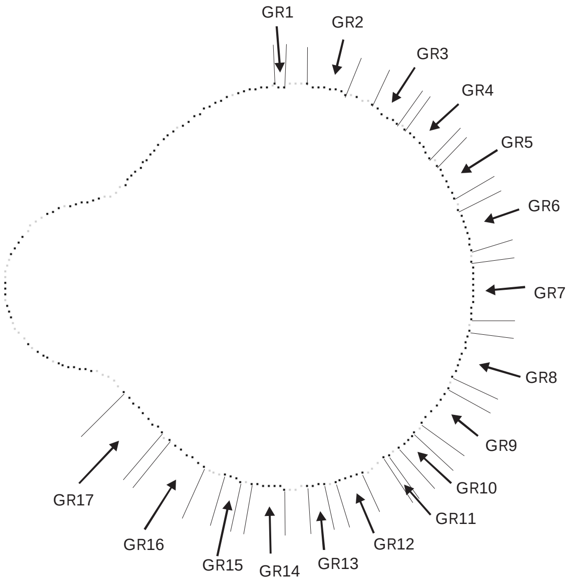


Figure 5.4: Grasp regions on a circular contour. The use of the accumulated curvature produces a set of small regions, instead of a large one covering most of the contour.

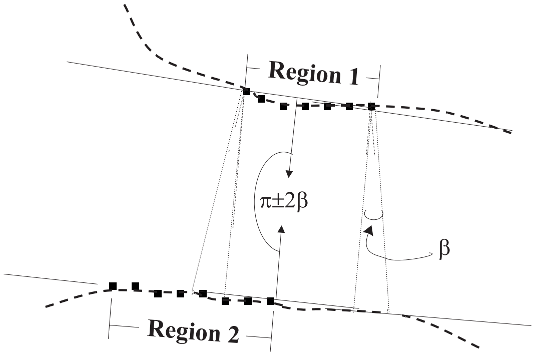


Figure 5.5: Compatibility test between grasp regions.

Step 2: Generation of a list of pairs of compatible regions

Once the grasp regions have been found, the next step is to build a list of pairs of grasp regions where both fingers of the robot could be placed to grasp the object. These regions are characterized by containing two points –one per region– such that, when the robot fingers are placed at them, the grasp complies with the force-closure criterion. The regions in each of these pairs will be termed hereinafter *compatible regions*.

The compatibility between two grasp regions is verified through the following conditions:

1. The angle between the normal vectors to each region is π. For this verification, the angular threshold (β) is considered for tolerance in the angle between both vectors; therefore, the allowed range between these vectors is $\pi \pm 2\beta$.

2. The projection of each region, in the direction of its normal, intersects with the other region, that is, the regions are confronted.

These conditions are verified not only between regions within the same contour, but also between regions of different contours, so all combinations of regions are checked. Figure 5.5 shows how these conditions are checked.

The pairs of compatible regions that meet the above conditions can be classified as corresponding to *squeezing* or *expansion grasps*:

- *Squeezing grasps* are those in which the space between the regions belongs to the object, and the robot fingers should be placed around these regions for grasping. They can be detected by the fact that the normals to each region points towards the space between both regions.

- *Expansion grasps* appear when the space the regions is empty, and the fingers should be placed in this space for executing the grasp. In this case, the normals to each region point towards outside the space between the regions.

The geometric rules given above for distinguishing between expansion and squeezing grasps assume that the list of contour points is ordered clockwise. Figure 5.6 shows some examples of squeezing and expansion grasps. Squeezing grasps are drawn as arrows pointing towards each other, while arrows pointing in opposite directions are used for expanding grasps.

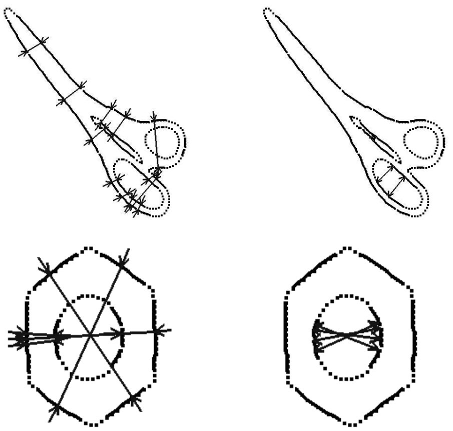


Figure 5.6: Examples of squeezing (left) and expansion (right) grasps.

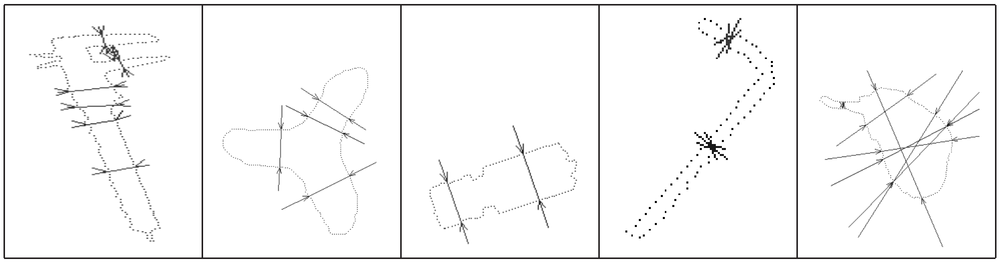


Figure 5.7: Sets of grasps extracted from compatible regions.

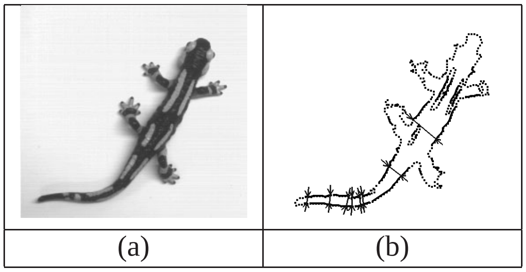
Step 3: Grasp refinement

This step involves the selection of a pair of grasp points within each pair of compatible regions, and can be seen as a refinement of the compatible regions. This selection can be performed by exhaustively checking each point from each region and select the pair that best complies with the quality criteria described in section 5.2.6. However, a faster procedure has been used and only a neighborhood around the center of the regions is checked.

Figure 5.7 shows the set of grasps selected in this step for several objects. Figure 5.8 shows the output of this step on an object that can not be properly segmented with the method proposed in section 4.2.1, since the image is not bimodal. As it can be observed, as long as the shape of the object is preserved, valid grasps can still be obtained. Figure 5.9 shows the results of this step on an object proposed in [56]. In that work, an analytical method was described for finding all the grasps that satisfy the force-closure condition. The grasp refinement proposed here constitutes a simpler procedure that can find the same grasps.

Step 4: Global grasp evaluation

The selected grasps are evaluated according to how loosely they comply with the gripper-adaptation and the force-closure criteria. In addition, a measure associated to the distance-



(a) (b)

Figure 5.8: Toy salamander: (a) Original image; (b) Obtained grasps and grasp regions.

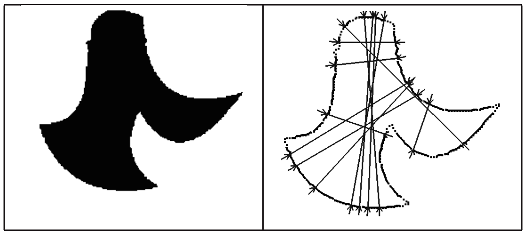


Figure 5.9: Synthetic image from [56].

to-centroid criterion is also considered. This produces a set of three quality values. Since these values correspond to different magnitudes, a normalization is performed [130, 132].

Using this set of normalized values, a global evaluation is computed. This global evaluation of a grasp is task-dependent. In this thesis, a simple linear combination of the quality values, each value being given the same weight, has been considered. This computation produces always a positive evaluation value. The optimal evaluation corresponds to value 0. Table 5.2 shows the measurements associated to the three above stability criteria and the global quality value for each of the grasps selected on a given object.

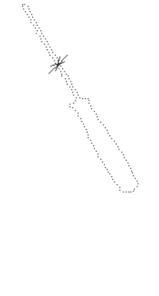
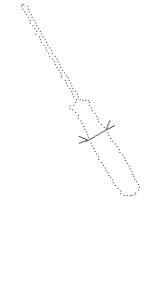
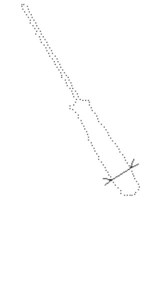
A study on which could be the optimal distribution of weights, and on other possibilities for combining the quality values –considering, however, the case of a three-fingered gripper– can be found in [36, 131]. Other procedures for computing a single evaluation value for a grasp could also applicable at these step. An example, although computationally more expensive than the procedure considered here, is described in [59].

Step 5: Grasp sorting and selection

The grasps are sorted according to their global evaluation, and the grasp with the highest evaluation is selected. The selected grasp is described by a pair of points on the contour, each one being the center of the region where the robot fingers should be placed. It is also specified if it is an expansion or a squeezing grasp.

In table 5.2, the selected grasp would be grasp number 2, since it features the lowest global quality value and complies with all the thresholds associated to the stability criteria. Figure 5.10 shows the grasps selected out of the sets of stable grasps indicated in figure 5.7.

Table 5.2: Stability measurements for a set of selected grasps. Angles α_i and β_i are expressed in degrees and distances γ_i in image space pixels. Measurements exceeding the considered thresholds appear in bold type.

Grasp no.	Grasp	Stability measurements
1		α_1: 178.315° α_2: 178.922° β_1: -14.0362° β_2: -15.4819° $\gamma_{1,2}$: **79.9031** Global quality value: 2.14897
2		α_1: 174.116° α_2: 178.756° β_1: -6.65943° β_2: 3.62148° $\gamma_{1,2}$: 3.27951 Global quality value: 0.753267
3		α_1: 175.487° α_2: 176.128° β_1: 3.82875° β_2: 1.41442° $\gamma_{1,2}$: **53.0433** Global quality value: 0.991562
Stability thresholds: α: 160°, β: 10°, γ: 25		

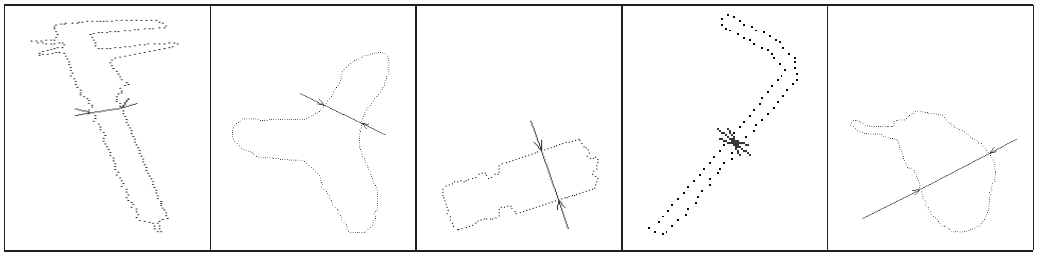


Figure 5.10: Grasps selected by the grasp search module.

5.3 Translation of a grasp between object descriptions

The procedure for grasp translation involves applying a grasp description for an already available computed object description to a new object description, independently of the origin of both descriptions. This procedure is the central component of the *external grasp tracking* filter, in which the object descriptions come from images provided by different cameras, and the grasp is translated from the description in one reference image to the other one. The grasp translation is thus used by this filter as a procedure to find the correspondences in the image computed from one camera to the grasp points selected in the image acquired with the other camera.

The input to the grasp translation procedure is composed of a grasp description, the object description that the grasp has been computed for, and another object description to which the grasp will be applied. The output is a new grasp description that is the same grasp given as input but with its internal representation updated for the new object. These input and output coincide with those required by the *external grasp tracking* filter.

This procedure is also a fundamental operation within the *grasp search and tracking* filter. In this case, the object descriptions come from images in a sequence acquired from the same camera, and the grasp is translated from one description to the next one in the sequence.

The movement of an object between pairs of images can be seen as the result of applying some transformations, either in the case of a pair of stereo images or in the case of a sequence of images acquired by the same camera. In the first case, the transformations are associated to the rigid relationship between the stereo cameras, while, in the second case, they depend on the movement model of the object. Depending of the particular applied transformations there exist some object features that are invariant with respect to them, so these object features are observed identically in all images. If the location of the grasp points can be expressed with respect to these invariant features, then this location is invariant with respect to the above transformations. Therefore, if one object description is a transformation of another one, the location of a grasp in the latter can also be applied to the former. These invariant features constitute a reference frame with respect to which grasps can be invariantly specified.

This strategy allows to translate –and, therefore, to track– the grasp points along a sequence of images or to find correspondences between pairs of images [160]. Similar strategies for finding correspondences have also been used in other works [64].

The translation procedure does not rely on any specific set of invariant features –that is, any specific reference frame. In this thesis, the following possibilities for this set and its associated grasp description have been explored:

- **Translation based on an invariant grasp description with respect to a B-spline.** Sampled points on the B-spline used as the invariant set of features. The grasp is expressed in RRP coordinates (see sections 4.5 and 5.2.5), with the reference components of the coordinates (*Ref1* and *Ref2*) being located at the sampled points on the B-spline.

- **Translation based on an invariant grasp description with respect to second-order moments of the object.** In this case, relative location coordinates with respect to second-order moments (see section 5.2.5) are used to describe the grasp. The invariant features considered are the centroid and the minimum inertia axis of the object.

- **Translation based on the computation of the homography between the contours of two objects.** In this case, the point correspondences used to build the homography constitute the set of invariant features. The grasp points are translated based on this homography.

Nevertheless, with the appropriate transformations, it is possible to switch between the above descriptions or to have the grasp expressed with respect to other reference frames. The most convenient representation of the grasp can thus be selected depending on the task to perform.

5.3.1 Translation based on an invariant grasp description with respect to a B-spline

As mentioned in section 4.4.1, B-splines are often used as *active contours* to track the shape of an object within a set of images –which could be either a sequence or images from different images. Essentially, the tracking procedures described in some works [20, 160, 183] associate an active contour to the tracked object in every image, which is the result of applying some transformations to a contour used in a reference image –for instance, the previous image in the sequence. These transformations are defined in image space and comprise: translations on the image plane (2 dimensions), rotations around the normal to this plane and changes of scale. If no refinement step is added to these transformations that modifies the shape of the active contour to better fit it to the apparent silhouette of the object, it can be considered that it has been rigidly translated from one image to the next one. As mentioned in section 4.4.5, under these conditions, the sampling of points on a B-spline used as active contour is invariant with respect to the above transformations. Therefore, they can be used as an invariant reference frame with respect to which all the points in the contour –including the grasp points– can be expressed.

As mentioned in section 4.5, the RRP coordinates can provide a straightforward indexing of the points of the contour in cases such as when the contour is obtained from the sampling of a B-spline. In this case, the points sampled from the spline can be used as the *reference points* of this coordinate system. The rest of the contour points can be computed from them as *interpolated points*. The advantage of this indexing is that, being defined from the sampled points of the spline, is invariant with respect to the transformations mentioned above. Therefore, the RRP coordinates of a pair of grasp points can be used to translate the grasp points from a contour to another one, provided that each contour is extracted from an instance of the same spline to which some of the above transformations have been applied, since their RRP coordinates coincide. The difference lies in the location coordinates in each contour, which are not the same, but can be computed from the RRP coordinates and the list of points in each case. This grasp translation based on the RRP coordinates is invariant with respect to the same four transformations mentioned above, with which the spline is rigidly translated. This means that this transformation is invariant with respect to four degrees of freedom (d.o.f.).

Table 5.3 shows some examples of this translation based on the RRP coordinates, with the sampled points of a B-spline being used to set the reference points. As it can be observed, the RRP coordinates of the grasp selected for the original contour are applied to the new contour to define the grasp.

This strategy has been used in [160] to track a moving object and a grasp on this object simultaneously. Nevertheless, the experiments described in this work consider only the external contour of the object, since it was the only one used for tracking the object. However, this does not imply a loss of generality of this strategy, which could be easily extended to use all contours, external and internal, for the tracking.

5.3.2 Translation based on an invariant grasp description with respect to second-order moments of the object

This strategy is an alternative to the previous one when no object tracking is performed, and therefore no transformed B-spline is available. In this case, the description of each contour as a list of points, extracted individually in each image, does not provide directly an invariant framework for expressing the location of the grasp points, since the order and the number of contour points may be different. Therefore other object features have to be considered.

As explained in section 5.2.5, the second-order moments of the object can be used to define a reference system that is invariant with respect to the same degrees of freedom as the points sampled from a transformed B-spline: translations and rotations on the image plane, and changes of scale. In particular, the centroid of the silhouette of the object and the major and minor axis of its best-fit ellipse –which coincide with the minimum and maximum inertia axes, respectively– are a set of features that can define such a system. They are obtained from moments up to order two and, in addition, have a clear geometric meaning.

Section 5.2.5 describes a grasp description based on the relative location of the grasp line with respect to these features. This description is invariant with respect to the four d.o.f. mentioned above. This means that, given two contours, the second being the result of applying some of the above transformations to the first one, the relative location of the grasp line in each contour corresponding to the same grasp coincide. Therefore, this relative description of the grasp line can be used to translate a grasp from one contour to the other one. The location and RRP indexing coordinates can be computed from the relative location of the grasp line and the list of contour points, and will be different for each contour. Table 5.4 shows an example of this type of translation.

Another set of invariant features is given by the *Hu invariants* [88]. These moments are also invariant with respect to translations, rotations and changes of scale on the image plane. They have been extensively used in the field of shape recognition. Nevertheless, they have not been considered in this thesis due to problems with expressing the location of the grasp points with respect to them.

Both the second-order moments and the Hu invariants have shown in experiments with real images to be quite sensitive to noise in the extraction of the silhouette of the object [117, 118].

Table 5.3: Grasp translation using an invariant description with respect to a B-spline.

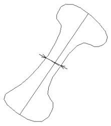
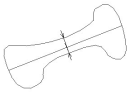
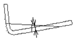
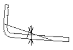
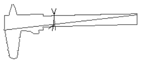
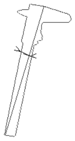
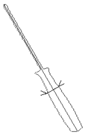
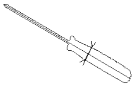
Original/translated grasp	Location coords.	RRP coords.	Relative coords.
	C1: 0 P1: 587 C2: 0 P2: 242	C1: 0, Ref1: 40 Interp1: 6 C2: 0, Ref2: 15 Interp2: 25	distC: -6.27154·10^{-5} angleWImin: -95.9817° distGP1: -0.00126012 distGP2: 0.000721252
	C1: 0 P1: 607 C2: 0 P2: 243	C1: 0, Ref1: 40 Interp1: 6 C2: 0, Ref2: 15 Interp2: 25	distC: -1.49535·10^{-5} angleWImin: -87.8033° distGP1: -0.00126508 distGP2: 0.000715779
	C1: 0 P1: 64 C2: 0 P2: 102	C1: 0, Ref1: 14 Interp1: 4 C2: 0, Ref2: 24 Interp2: 3	distC: -0.0124232 angleWImin: 95.6864° distGP1: -0.00722776 distGP2: 0.0077414
	C1: 0 P1: 80 C2: 0 P2: 102	C1: 0, Ref1: 14 Interp1: 4 C2: 0, Ref2: 24 Interp2: 3	distC: -0.0100289 angleWImin: 98.4322° distGP1: -0.00490846 distGP2: 0.00570836
	C1: 0 P1: 18 C2: 0 P2: 115	C1: 0, Ref1: 4 Interp1: 2 C2: 0, Ref2: 34 Interp2: 5	distC: 0.010855 angleWImin: 90.7285° distGP1: -0.0140811 distGP2: 0.0134111
	C1: 0 P1: 18 C2: 0 P2: 112	C1: 0, Ref1: 4 Interp1: 2 C2: 0, Ref2: 34 Interp2: 5	distC: 0.011364 angleWImin: 86.2359° distGP1: -0.0147243 distGP2: 0.0126604

Table 5.4: Grasp translation based on second-order moments.

Original/translated grasp	Location coords.	RRP coords.	Relative coords.
	C1: 0 P1: 34 C2: 0 P2: 79	C1: 0, Ref1: 100 Interp1: 1 C2: 0, Ref2: 235 Interp2: 1	distC: 0.00611918 angleWImin: -102.796° distGP1: -0.00892122 distGP2: -0.00253176
	C1: 0 P1: 32 C2: 0 P2: 75	C1: 0, Ref1: 95 Interp1: 1 C2: 0, Ref2: 224 Interp2: 1	distC: 0.00611918 angleWImin: -102.796° distGP1: -0.00892122 distGP2: -0.00253176
	C1: 0 P1: 74 C2: 0 P2: 102	C1: 0, Ref1: 221 Interp1: 1 C2: 0, Ref2: 515 Interp2: 1	distC: -6.4419·10^{-5} angleWImin: -260.217° distGP1: -0.000506822 distGP2: 0.00273194
	C1: 0 P1: 50 C2: 0 P2: 146	C1: 0, Ref1: 148 Interp1: 1 C2: 0, Ref2: 433 Interp2: 1	distC: -6.4419·10^{-5} angleWImin: -260.217° distGP1: -0.000506822 distGP2: 0.00273194
	C1: 0 P1: 51 C2: 0 P2: 100	C1: 0, Ref1: 152 Interp1: 1 C2: 0, Ref2: 296 b Interp2: 1	distC: 0.000864146 angleWImin: -89.922° distGP1: -0.00411561 distGP2: 0.00347848
	C1: 0 P1: 58 C2: 0 P2: 117	C1: 0, Ref1: 172 Interp1: 1 C2: 0, Ref2: 347 Interp2: 1	distC: 0.000864146 angleWImin: -89.922° distGP1: -0.00411561 distGP2: 0.00347848

5.3.3 Translation based on the computation of the homography between the contours of two objects

An *homography* $\mathbb{H}$ is a linear projective transformation between two planes [54, 135]. This transformation produces, given a point in one plane, its corresponding point on the other plane. It is normally used in computer vision to find correspondences in two different views between points that belong to the same physical plane. The computation of a homography matrix requires the a priori knowledge of at least four point correspondences between the two views, provided that no three of them are collinear. With more than four correspondences, it is possible to compute the homography matrix using numerical, iterative methods, which are less sensitive to measurement errors in the initial correspondences [135]. Anyway, the correspondences between two views of an object –provided that all the points of the object belong to the same physical plane– constitute an object-centered system, and the homography can be used to translate points from one view to the other.

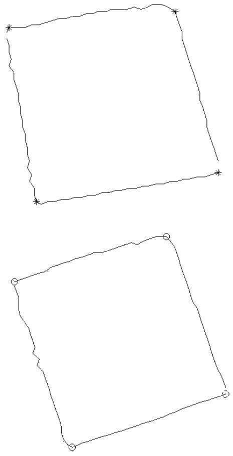
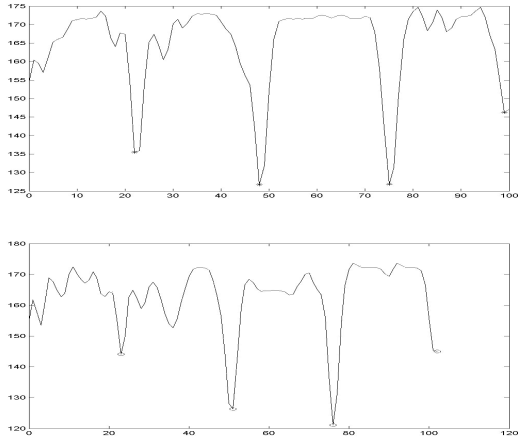
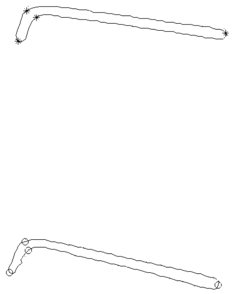
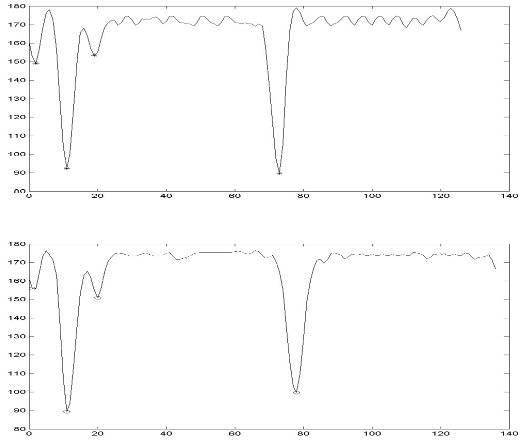
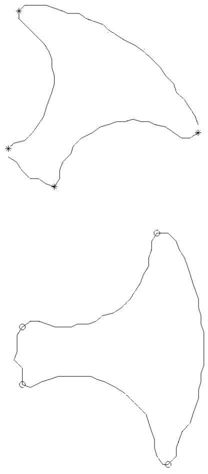
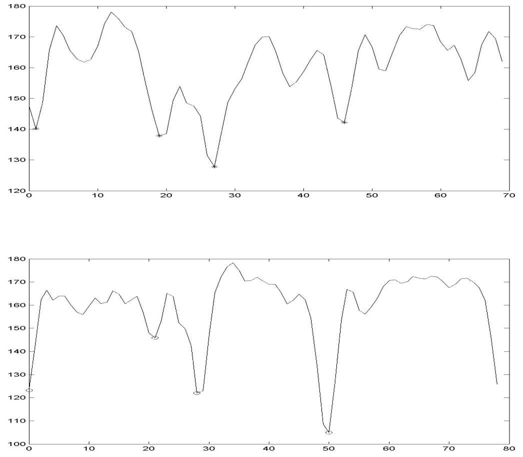
The initial correspondences required for the computation of the homography are often assumed or selected manually [111]. Zhang et al. [205] select corners in one image and try to match image subwindows centered at them in the other image. In other works, the selection of correspondences is based on the search of epipolar tangencies between two images [37, 127].

In this thesis, a homography has been considered for translating the grasp points between two views of an object, assuming that, as mentioned in section 1.2.1, the object is planar. In addition, a matching procedure has been developed for the automatic selection of correspondences between the projected silhouettes of the object that is based on the analysis of the curvature of their contours. For simplicity, only the external contours have been considered. Anyway, an extension of this procedure to include both the external and the internal contours would imply the definition of an initial step to determine the correspondences between the internal contours. The matching procedure would be applied to corresponding contours.

The proposed matching procedure uses the curvature to try to select points that are representative of the shape of the contour. In particular, it selects corners in one of the corresponding contours, which are identified as peaks in the vector of curvatures computed for that contour. This use of the curvature for the selection of relevant points has also been considered in other works [67, 184, 193]. As four point matches are required to compute the homography, the four points within the contour with the highest curvature are selected. The procedure tries next to match them on the other corresponding contour. For each point p_i, instead of considering an image subwindow and try to find a match in the other image as in [205], the procedure considers a neighborhood in the curvature function centered at that point and tries to find the match in the curvature function computed for the other contour. The center p'_i of the matched neighborhood is taken as the point corresponding to p_i. Table 5.5 shows four selected points on the contour of an object and the correspondences found on the contour corresponding to a different observation of that object, using the curvature of both contours.

Let I_i and I'_i be the images that contain the corresponding contours. Once the homography matrix has been computed, the grasp points $p_{g_{1_i}}$ and $p_{g_{2_i}}$ available in image I_i are translated into image I'_i. Ideally, the translated points should lie on the corresponding contour extracted from I'_i. Nevertheless, due to errors in the contour extraction in I_i, in the

Table 5.5: Selected points (∗) on a reference contour and its curvature and their correspondences (○), detected by curvature matching.

Reference/current contour	Curvature of reference/current contour
	
	
	

selection of the corresponding points, and I'_i and/or in the computation of the homography matrix, the translated points p'_1 and p'_2 may happen not to be in that contour, but close to it. For this reason, the line joining them is considered to be the translated grasp line and the intersections between this line and the contour are computed. At least, there should be two intersections. The two intersection points p'_{g1_i} and p'_{g2_i} that are closest to p'_1 and p'_2, respectively, are considered to be the translated grasp points in image I'_i. This strategy ensures that the translated grasp points always belong to a contour. The results of translating grasps within the pairs of figures shown in table 5.5 can be observed in tables 5.6, 5.7 and 5.8.

The homography is invariant with respect to a movement with six d.o.f. between two views of the plane. These d.o.f. comprise translations on the image plane, changes of a scale –equivalent to a translation along the normal to the image plane-, and rotations around both the axes used to define the image plane and the axis perpendicular to this plane. Therefore, the proposed homography-based grasp translation has the advantage over the two other methods explained above that it can be used to translate grasp points from one image to the next one with six degrees of freedom. This translation provides two grasp points that can be used to compute the relative location of the grasp line with respect to the second-order moments of the object, from which the location and RRP indexing coordinates of the grasp points can be found.

5.4 Grasp tracking

The tracking of a grasp along a sequence of images is performed by the *grasp search and tracking* filter shown in figure 3.6. The input of this filter is an object description. The filter searches for a grasp on the first object received and tries to track this grasp on each new object that is received later. As output, it produces a description of the grasp selected or tracked on the object given as input.

The search of the grasp on the first object received is performed using the grasp-search module described in section 5.2. In the case that no stable grasp can be found on this object, the grasp search is repeated for any new object received until a grasp can be selected. The tracking of an already computed grasp from a previous object to the current one involves a translation of the grasp to the current object and an evaluation of the stability of the translated grasp. The translation can be executed according to any of the procedures described in section 5.3. The evaluation of the translated grasp is based on the *gripper-adaptation*, *force-closure* and *distance-to-centroid* criteria introduced in section 5.2.6.

If any of the above stability criteria fails, the grasp should not be considered stable. Nevertheless, as it has been observed in the experiments, it may fail at isolated occurrences along the sequence of object descriptions, due to the noise that appears in the extraction of each description. For this reason, a *grasp evaluation tolerance* has been defined as the number of stability criteria that are allowed to fail in the evaluation of a grasp. In most of the experiments, a value of 1 of this tolerance has shown to be enough. In particular, the β_1 and β_2 angles associated to the force-closure criterion (see figure 5.2) have shown to be the most sensitive to local noise in the shape of the object around the grasp points. The failure of two tests have been rare and has involved the gripper-adaptation and the force-closure criteria.

When a stable grasp is available for the current object, either as the outcome of a grasp

Table 5.6: Grasp translation on the contour of a floppy disk based on a homography.

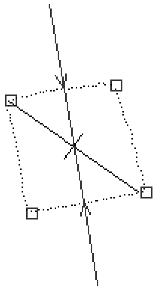
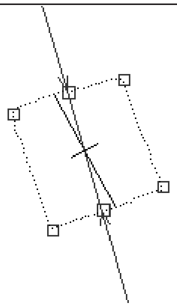
Reference contour	Current contour
	
Selected point correspondences: (158, 97) (173, 173) (251, 157) (229, 86) Selected grasp points: p_{g_1} = (189, 85) p_{g_2} = (202, 160)	Selected point correspondences: (140, 96) (167, 174) (242, 143) (214, 72) Translation of the grasp points: p'_1 = (172, 77) p'_2 = (195, 155) Applied grasp points: p'_{g_1} = (171, 76) p'_{g_2} = (194, 156)
Location coordinates: C1: 0 P1: 11 C2: 0 P2: 64	Location coordinates: C1: 0 P1: 12 C2: 0 P2: 64
RRP coordinates: C1: 0 Ref1: 32 Interp1: 1 C2: 0 Ref2: 189 Interp2: 1	RRP coordinates: C1: 0 Ref1: 33 Interp1: 1 C2: 0 Ref2: 188 Interp2: 1
Relative coordinates: distC: $5.3693 \cdot 10^{-5}$ angleWImin: 45.5085° distGP1: -0.00682327 distGP2: 0.00636881	Relative coordinates: distC: -0.000274003 angleWImin: 12.6099° distGP1: -0.00628479 distGP2: 0.00671952

Table 5.7: Grasp translation on the contour of Allen wrench based on a homography.

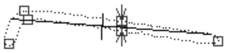
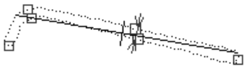
Reference contour	Current contour
	
Selected point correspondences: (98, 108) (117, 85) (267, 105) (114, 78) Selected grasp points: p_{g_1} = (183, 95) p_{g_2} = (183, 88)	Selected point correspondences: (135, 140) (155, 116) (317, 154) (152, 109) Translation of the grasp points: p'_1 = (222, 133) p'_2 = (220, 125) Applied grasp points: p'_{g_1} = (220, 133) p'_{g_2} = (222, 125)
Location coordinates: C1: 0 P1: 46 C2: 0 P2: 103	Location coordinates: C1: 0 P1: 52 C2: 0 P2: 111
RRP coordinates: C1: 0 Ref1: 136 Interp1: 1 C2: 0 Ref2: 305 Interp2: 1	RRP coordinates: C1: 0 Ref1: 151 Interp1: 1 C2: 0 Ref2: 326 Interp2: 1
Relative coordinates: distC: 0.0109106 angleWImin: -94.2873° distGP1: -0.00128966 distGP2: 0.00380866	Relative coordinates: distC: 0.00630399 angleWImin: -85.5316° distGP1: -0.00137337 distGP2: 0.00359572

Table 5.8: Grasp translation on the contour of the handle of a tool based on a homography.

Reference contour	Current contour

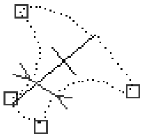
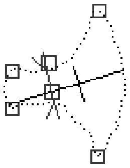

	

Image coordinates Selected point correspondences: (200, 183) (253, 163) (187, 118) (183, 169) Selected grasp points: p_{g_1} = (187, 152) p_{g_2} = (201, 161)	Image coordinates Selected point correspondences: (114, 97) (167, 126) (163, 42) (114, 76) Translation of the grasp points: p'_1 = (131, 68) p'_2 = (133, 84) Applied grasp points: p'_{g_1} = (130, 72) p'_{g_2} = (134, 90)
Location coordinates C1: 0 P1: 14 C2: 0 P2: 32	Location coordinates C1: 0 P1: 14 C2: 0 P2: 37
RRP coordinates C1: 0 Ref1: 41 Interp1: 1 C2: 0 Ref2: 95 Interp2: 1	RRP coordinates C1: 0 Ref1: 41 Interp1: 1 C2: 0 Ref2: 110 Interp2: 1
Relative coordinates distC: 0.00987252 angleWImin: -107.826° distGP1: -0.00220172 distGP2: 0.00612202	Relative coordinates distC: 0.00683894 angleWImin: -86.0528° distGP1: -0.00524987 distGP2: 0.00239644

search or after a translation, it is appended at the end of a list known as the *tracking window of grasps*. The procedure for grasp tracking uses this list as a record of already-computed grasps. When a grasp translation produces a grasp that exceeds the *grasp evaluation tolerance*, it is considered that the shape of the object has changed significantly and hence the translated grasp is no longer valid. Nevertheless, some of the grasps in the *tracking window* may still be applicable, so they are checked. These grasps are applied to the current object and evaluated until one of them complies with the evaluation tolerance. The grasps are checked starting from the end of the tracking window, so that more recently added grasps are checked first. When a valid grasp is found, it is copied at the end of the window, in order to always have the last applied grasp in this position.

If it has not been possible to successfully apply any of the grasps stored in the tracking window, a new grasp search is performed. If this search produces a stable grasp, it is appended at the end of the tracking window and hence considered as the new grasp to track. Otherwise, in order not to interrupt the grasp tracking, it has been decided to use the last tracked grasp –that is, the one that is at the end of the tracking window– in spite of its failure to meet the evaluation tolerance.

The overall procedure for grasp tracking is summarized in algorithm 5.2.

Algorithm 5.2 Grasp tracking

```
if no grasp has been previously computed then
  Search for a grasp on the contour (see alg. 5.1)
  if a stable grasp has been found then
    Add the grasp to the tracking window
  end if
else
  {Search for a stable grasp in the tracking window}
  Start at the end of the tracking window of grasps
  repeat
    Take previous grasp in the window
    Translate loc. of taken grasp to current contour
    Evaluate the taken grasp
  until a stable grasp is found or all the grasps in the window have been evaluated

  {Search for a new grasp if no stable one is found}
  if a stable grasp has been found then
    Add the grasp to the tracking window
  else
    Search for a grasp on the contour (see alg. 5.1)
    if a stable grasp has been found then
      Add the grasp to the tracking window
    else
      Use last selected grasp (although unstable)
    end if
  end if
end if
```

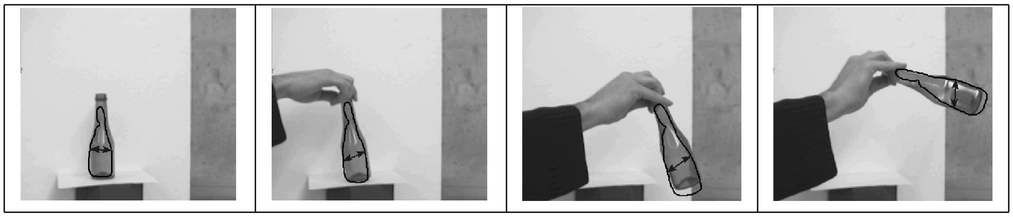


Figure 5.11: Tracking of an object and a selected grasp on a sequence of images.

The precision of the grasp tracking depends on the invariance, with respect to the transformations to be applied to the objects, of the relative location of the grasp points with respect to the object. Therefore, it is possible to perform the grasp tracking as long as the grasp description is invariant along the sequence of object descriptions. This conditions the posterior use of the grasp points –for instance, as input to a control law. Consequently, if the grasp points are intended to be used as one of the inputs to a control law, this law has to be restricted to produce, directly or indirectly, those transformations in the projection of the object under which the tracking of the grasp points is possible.

Table 5.9 shows an example of the tracking of a grasp along a sequence of contours performed by the *grasp search and tracking* filter. In this case, the grasp translation is based on the description of the grasp with respect to a spline that has been associated to the object to grasp. As mentioned in section 4.4.1, the use of splines and other parametric curves have been used in many methods for tracking an object along a sequence of images [20]. These methods have proved to be very useful in scenes in which the tracking of the object may be disturbed by the background other elements in the scene, as it is the case of the sequence of images shown in figure 5.11 [160]. In this sequence, an object is moving with respect to a fixed camera. The spline was manually associated as active contour to the object in the first image. The same spline used to track the object provides the reference frame with respect to which a stable grasp computed for the first object can also be tracked.

Table 5.10 provides an example of tracking in the case in which the grasp translation is based on the description of the grasp with respect to second-order moments. In tables 5.11 and 5.12, the tracking of the grasp is based on the computation of a homography between pairs of contours. Two cases are considered: the use of pairs that are consecutive along the sequence, and the use of the first and the current contours for building such pairs. As it can be observed, the use of the first contour for computing the homography provides a more accurate tracking of the grasp.

In the case of a stereo system, two sequences of contours are available, which are obtained from the images provided by each of the two cameras. In this case, the grasp tracking is performed through the joint use of the *grasp search and tracking* and the *external grasp tracking* filters. The former is used to perform the tracking, using algorithm 5.2, along one of the two sequences of contours. The latter translates the grasp tracked in each contour of this sequence to the corresponding contour in the other sequence. The decision on which sequence to use for tracking and which one to use for translation is arbitrary. In the particular implementation developed for this thesis, the grasp is tracked along the sequence associated to the left camera, and translated to the contours of the sequence corresponding to the right camera. An example of grasp tracking on a stereo sequence is shown in table 5.13.

Table 5.9: Grasp tracking with the grasp translation based on a B-spline.

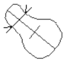
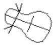
Sequence of contours				
Grasp descriptions				
Location coords.				
C1:	0	0	0	0
P1:	93	64	74	64
C2:	0	0	0	0
P2:	138	94	110	97
RRP coords.				
C1:	0	0	0	0
Ref1:	15	15	15	15
Interp1:	2	2	2	2
C2:	0	0	0	0
Ref2:	24	24	24	24
Interp2:	1	1	1	1
Relative coords.				
distC:	-0.00999382	-0.0151325	-0.0137287	0.0148569
angleWImin:	-39.3171°	95.4385°	96.5924°	-86.8101°
distGP1:	-0.00431769	-0.00695379	-0.00650696	-0.00671937
distGP2:	0.004994	0.00727206	0.00653133	0.00749864
Grasp type:	Squeezing	Squeezing	Squeezing	Squeezing
Grasp evaluation				
α_1:	160.651°	171.133°	175.35°	168.647°
α_2:	170.339°	173.205°	177.421°	179.343°
β_1:	2.48487°	-6.37438°	0.182468°	-5.65497°
β_2:	0°	-3.20428°	4.76364°	0°
$\gamma_{1,2}$:	23.3345	15.3298	17.8885	15
Global quality value:	1.57359	1.46212	1.48261	1.43592
No. of failing tests:	0	0	0	0
Stability thresholds: α: 160°, β: 10°, γ: 25				

Table 5.10: Grasp tracking with the grasp translation based on second-order moments.

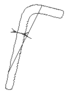
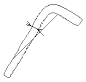
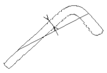
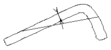
Sequence of contours				
Grasp descriptions				
Location coords.				
C1:	0	0	0	0
P1:	22	16	16	17
C2:	0	0	0	0
P2:	103	89	86	109
RRP coords.				
C1:	0	0	0	0
Ref1:	65	47	47	50
Interp1:	1	1	1	1
C2:	0	0	0	0
Ref2:	306	266	254	325
Interp2:	1	1	1	1
Relative coords.				
distC:	-0.000806452	-0.000806452	-0.000806452	-0.000806452
angleWImin:	-105.095°	-105.095°	-105.095°	-105.095°
distGP1:	-0.0062984	-0.0062984	-0.0062984	-0.0062984
distGP2:	-0.00185601	-0.00185601	-0.00185601	-0.00185601
Grasp type:	Squeezing	Squeezing	Squeezing	Squeezing
Grasp evaluation				
α_1:	175.769°	178.869°	176.672°	178.507°
α_2:	173.714°	173.601°	179.794°	174.365°
β_1:	0.784825°	0.537973°	2.99508°	5.52754°
β_2:	-0.251022°	1.47416°	0.528266°	4.16842°
$\gamma_{1,2}$:	2.09529	2.47407	3.71307	1.22053
Global quality value:	0.677195	0.67042	0.702498	0.775569
No. of failing tests:	0	0	0	0
Stability thresholds: α: 160°, β: 10°, γ: 25				

Table 5.11: Grasp tracking with the grasp translation based on the computation of a homography between consecutive contours. Points are expressed in image coordinates.

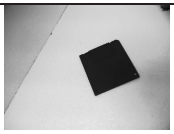
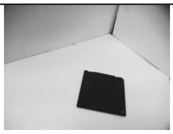
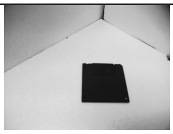
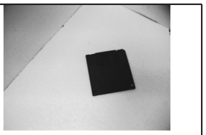
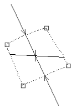
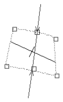
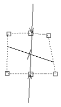
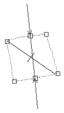
Image no.	1	2	3	4
Sequence of images				
Sequence of contours				
Grasp translation				
Selected points/ correspondences:	(145, 99) (175, 174) (258, 138) (217, 68)	(155, 127) (142, 195) (234, 210) (230, 144)	(152, 113) (150, 185) (239, 186) (227, 117)	(159, 95) (171, 172) (251, 159) (232, 88)
Contour pair for the homography:	–	1-2	2-3	3-4
Selected/ translated grasp:	p_{g_1} = (174, 77) p_{g_2} = (208, 150)	p'_1 = (191, 128) p'_2 = (185, 194)	p'_1 = (187, 107) p'_2 = (187, 181)	p'_1 = (194, 84) p'_2 = (203, 161)
Applied grasp:	–	p'_{g_1} = (191, 128) p'_{g_2} = (182, 197)	p'_{g_1} = (189, 108) p'_{g_2} = (186, 180)	p'_{g_1} = (194, 84) p'_{g_2} = (202, 160)
Grasp evaluation				
α_1:	177.461°	177.684°	179.381°	179.532°
α_2:	179.639°	178.735°	180°	178.941°
β_1:	-2.06759°	-4.66335°	-1.85622°	0.764964°
β_2:	0.696185°	-3.87852°	2.38594°	4.41145°
$\gamma_{1,2}$:	0.695397	5.38913	3.45534	3.0882
Global quality value:	2.114	2.09657	2.06694	2.31499
No. of failing tests:	0	0	0	0
Stability thresholds: α: 160°, β: 10°, γ: 25				

Table 5.12: Grasp tracking with the grasp translation based on the computation of a homography with respect to the first contour. Points are expressed in image coordinates.

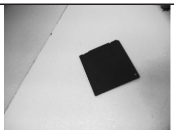
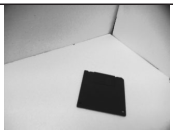
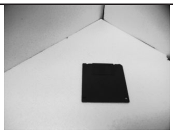
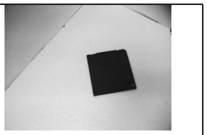
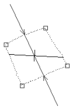
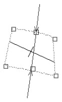
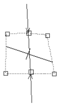
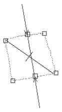
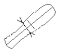
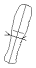
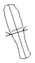
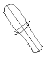
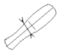
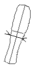
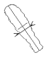
Image no.	1	2	3	4
Sequence of images				
Sequence of contours				
Grasp translation				
Selected points/ correspondences:	(145, 99) (175, 174) (258, 138) (217, 68)	(155, 127) (142, 195) (234, 210) (230, 144)	(152, 113) (150, 185) (239, 186) (227, 117)	(159, 95) (171, 172) (251, 159) (232, 88)
Contour pair for the homography:	–	1-2	1-3	1-4
Selected/ translated grasp:	p_{g_1} = (174, 77) p_{g_2} = (208, 150)	p'_1 = (191, 128) p'_2 = (185, 194)	p'_1 = (186, 107) p'_2 = (190, 177)	p'_1 = (192, 83) p'_2 = (206, 158)
Applied grasp:	–	p'_{g_1} = (191, 128) p'_{g_2} = (182, 197)	p'_{g_1} = (185, 108)i p'_{g_2} = (189, 180)	p'_{g_1} = (192, 85) p'_{g_2} = (205, 160)
Grasp evaluation				
α_1:	177.461°	177.684°	179.518°	179.005°
α_2:	179.639°	178.735°	180°	179.433°
β_1:	-2.06759°	-4.66335°	-5.37918°	-3.07866°
β_2:	0.696185°	-3.87852°	-3.17983°	0.552563°
$\gamma_{1,2}$:	0.695397	5.38913	3.05085	3.53397
Global quality value:	2.114	2.09657	2.20242	2.25469
No. of failing tests:	0	0	0	0
Stability thresholds: α: 160°, β: 10°, γ: 25				

Table 5.13: Tracking along a sequence of stereo images. The relative location coordinates with respect to the second order moments of the object are used to provide an invariant description of the grasp.

Left image				
Grasp evaluation				
α_1:	178.353°	177.965°	177.457°	173.86°
α_2:	179.712°	173.077°	176.531°	174.313°
β_1:	2.96068°	**10.0555°**	8.91493°	4.92388°
β_2:	-2.69428°	-8.91493°	-6.38709°	**-10.7817°**
$\gamma_{1,2}$:	9.49118	10.9153	8.05655	9.60441
Global quality value:	1.06012	1.20131	1.32709	1.23583
No. of failing tests:	0	1	0	1
Right image				
Grasp evaluation				
α_1:	179.242°	179.974°	172.347°	172.804°
α_2:	179.034°	179.948°	176.797°	179.944°
β_1:	**14.3402°**	9.06514°	**11.0492°**	3.93815°
β_2:	-3.07997°	-7.36927°	-6.9231°	-7.03124°
$\gamma_{1,2}$:	9.4299	10.773	8.62441	9.15255
Global quality value:	1.60491	1.32857	1.30981	1.28766
No. of failing tests:	1	0	1	0
Stability thresholds: α: 160°, β: 10°, γ: 25				
Relative coordinates of the tracked grasp distC: 0.00377836 angleWImin: -86.9834° distGP1: -0.00446712 distGP2: 0.00434245				

Although experiments have been performed considering only the external contour of the objects, this grasp tracking strategy could also be applied to objects having internal contours. Nevertheless, if RRP coordinates are used as the invariant description of the grasp and the object on which the grasp is tracked has more than one internal contour, it would be necessary to determine the correspondence between the internal contours of the current and the reference descriptions of the object.

Chapter 6

Control law: Gripper-to-object positioning

The control law and the generation of target values for the grasp points is explained here. A brief description of the virtual actuators corresponding to the robot and data viewers is also provided.

6.1 General overview

The positioning of the gripper with respect to the object is controlled by the *target generator* and the *control law* filters described in section 3.2. The *control law* filter uses features extracted from the images provided by the stereo pair of cameras –in particular, the position of the grasp points in image space– to guide the gripper towards the object. The target position of the gripper is determined by the target position of the grasp points in the image space of each camera. This target position is computed also from visual data by the *target generator* filter.

6.2 Visual servoing for gripper-to-object positioning

The positioning of the gripper with respect to the object is one of the fundamental steps not only in object manipulation tasks, but also in other tasks such as object exploration. Many works have thus considered this step from a general point of view, without focusing on the details of the task within which the positioning was performed.

Espiau et al. proposed in [55] a general methodology for the design of visual servoing tasks. They used the concept of *task function* [167] to compute the error with respect to the desired relative positioning between the gripper and the object. The concept of *virtual linkage* was used to define the relative position between a sensor –which is attached to the robot in this work– and the object of interest. Yoshimi and Allen [202] use visual servoing to perform first an alignment of a peg with respect to a reference hole and then execute an insertion movement. No calibration data are explicitly required here, and the visual servo control can be performed with an approximation of the interaction matrix. 3D information is estimated through the movement of the image features after known movements of the robot are performed. Papanikolopoulos and Smith [142] use window-based techniques to more robustly track the image features. Emphasis is put in this work in the selection of features that do not make the interaction matrix singular; the problem of occlusion of features is also considered. In addition, the feature tracking method is designed to ensure that the tracked features belong to the object on each image.

In [42], Cowan and Koditschek consider the positioning with respect to a planar convex object lying on a planar surface. Nevertheless, the image plane is observed perpendicularly so that it is reduced to a line, and the control task only involves points that move along this line. The control law is designed following a *sequential composition* approach, in which a palette of sub-controllers and an image-based switching law is defined. Chesi et al. [33, 34] consider the positioning with respect to a set of planar objects lying on the same plane. They compute an homography between the contours extracted from a current and the target view of the objects, and use this homography in the computation of the control law. An eye-in-hand configuration with a single camera is used in that work. The target position between the robot and the objects is defined manually. Berry et al. describe in [16] a task that is composed of a sequence of three control sub-tasks, although no specific mechanism for switching between sub-tasks is described. The first sub-task positions the robot with respect to the object, while the other two are oriented to perform an exploration of the object. Finally, Martinet and Cervera [121] compare the building of an interaction matrix for an eye-in-hand stereo vision system following the *multi-camera visual servoing approach* of Malis et al. [116] with a simple stacking of the interaction matrices of each camera, in which the relationship between each camera and the end-effector is neglected. As they show, some problems appear in the second case for reaching the equilibrium during positioning and object exploration tasks.

Among the works that perform an actual grasping of the object of interest in addition to the positioning task, Allen et al. [1], use two fixed cameras to compute the 3D position of a moving object –experiments were performed with a toy train moving along a circular path. The object is detected as a blob in each image and the 3D position of its centroid is computed. A predictive control of the robot arm is performed. Once the tracking of the object with the moving arm is stable, the grasp is executed. Grasp execution with a single, stand-alone camera was also considered by Yoshimi and Allen [203]. In this case, both the object and the fingers are observed, and snakes are used to track them. Grasp determination is considered here as an off-line step, performed before the grasp execution. Vision is also used to monitor the stability of the grasp once it has been executed. It is highlighted the need of the fusion of vision and contact/force data for a truly precise control of the grasping task.

A stand-alone uncalibrated stereo head is used by Horaud et al. in [84]. Grasping is considered here as the alignment of two solids in 3D projective space, the end-effector and the object of interest. The grasp determination –defined as the selection of the target relationship between the object and the end-effector– is performed in an off-line stage. The on-line stage, during which the interaction matrix is estimated, corresponds to the execution of the grasp.

In [60], Fujimoto et al. are given the grasp in the form of a grasp polygon –a polygon joining the grasp points–, although no explanation is given on how this polygon is computed. They compute some data that depend on this grasp description and the centroid and minimum inertia axis of the object of interest is computed. These and other data are updated during the on-line stage.

6.3 Analysis of the control law

6.3.1 General analysis

As the control task is defined in image space, it involves moving two points –the grasp points at their current position– towards two target positions, which correspond to the projection in the image plane of the grasp points at the desired relative position between the robot and the object.

The control law has to take into account the restrictions in the d.o.f. imposed by the particular procedure for the translation of the grasp points used by the grasp tracking algorithm. Two cases can be considered:

- **Translation of the grasp using a B-spline or second-order movements to define a reference frame.** As mentioned in section 5.3, this translation can be performed as long as the movement of the object in image space is restricted to:

  - Translations on the image plane
  - Changes of scale
  - Rotations on the image plane

  For simplicity in the specification of the movement of the robot based on these restrictions, it is assumed that the stereo cameras are mounted so that their image planes are parallel to the plane where the object lies. The image plane of each camera is assumed to be defined by two axes X and Y. A coordinate frame is associated to each camera so that its X and Y axis define a plane parallel to the image plane and the Z axis is perpendicular to both planes. Under this configuration, the movement of the gripper of the robot –to which the stereo head is rigidly attached– should comply with the following restrictions:

  - Any translation of the cameras –which can be seen as a composition of translations around the X, Y, and Z axes of the coordinate frames associated to the cameras. It produces translations and changes of scale of the shape of the object on the image plane.
  - Any rotation around an axis perpendicular to the image plane. According to the model of the stereo head described in section 4.1.1, this rotation axis should be parallel to the Z axes of the coordinate frames associated to the cameras. The effect of this movement is a rotation of the shape of the object in the image plane.

  Both the above restrictions keep the image and the object planes parallel. Four d.o.f. are thus available for the definition of the control law.

- **Translation of the grasp using an homography.** The homography is invariant with respect to the 6 d.o.f. of the movement of the camera –and, therefore, of the gripper. The restrictions come from the procedure used to find the point correspondences on the shape of the object that are used to compute the homography. With a procedure that allows 6 d.o.f., the control law could also be applied on all those d.o.f., that is, translations and rotations with respect to the X, Y, and Z axes of the cameras.

The above restrictions will have to be taken into account when designing the output space of the control law. In addition, in any of the above cases, the control law will have to ensure that the line joining the target points towards which the gripper is guided belongs to or is parallel to the object plane.

In addition to the above restrictions in the d.o.f., other restrictions regarding the trackability of the grasp and its observability in its target position have also to be taken into account. These restrictions set limits on which target relative positions between the robot and the object can be considered. Two cases can be distinguished:

- **Full observability of the object is required.** The whole object has to be observed in image space for the location of the grasp points to be possible. This is the case in which the grasp is described with respect to second-order moments or tracked through the computation of homographies. In the case of a description with respect to second-order moments, partial occlusions of the shape of the object –or enlargements, due to to the proximity of other elements– would cause the invariance of these moments along the sequence of images to be lost, and therefore invalidate the description of the grasp. In the case of the homography-based grasp tracking, these occlusions may –although not necessarily– hide some correspondences that could have been used to compute the homography, or lead to the selection of false correspondences.

  In this situation, the target relative position between the object and the robot, as well as the intermediate relative positions induced by the control law should comply with the following requirements:

  - The whole object appears in the field of view of the cameras.
  - There is no overlapping in image space between the object and the gripper or any other element that may appear in the scene.

- **Full observability of the object is not required.** This is the case the tracking of the object and the description of the grasp are based on the use of a model, such as that defined by a B-spline. Several work exist that show that it is possible to track an object even with partial occlusions of the object and/or other perturbances [11, 200]. If the grasp is described with respect to the model, as long as the model could be located in image space, it would be possible to determine the position of the grasp points, even in the case that they happened to be outside the limits of the image.

The above restrictions will have to be considered during the computation of the target position of the grasp, which will have to comply with them. In addition, the control law should be designed so that it does not induce any movement of the robot that causes the projection of the object in image space to violate any of them. Ideally, both the target position of the grasp and the path between the current position and this target should be within the limits imposed by the above restrictions.

Prior to the use of the control law, it should be ensured that these restrictions are met. It is assumed that another control task may have to be executed, in a previous step, which, among other goals, brings the state of the robotic system within these restrictions. The control law defined in this section can then be executed and may achieve its goal –i.e. reaching the target position of the grasp points–, possibly leaving the system within a set of conditions under which it is possible to start another control task. Anyway, the use of this control law should stop before or as soon as one of the restrictions mentioned above fails or cannot be guaranteed.

For simplicity, with respect to the restrictions in the d.o.f., this thesis has considered only the case in which the control law is applied on four d.o.f. Regarding the observability and trackability of the object and the grasp points, the grasp description based on second-order moments has been selected for tracking and, therefore, full observability of the object is required. Nevertheless, only the checking of the observability at the target position has been considered. Therefore, the target grasp will be generated taking into account full-observability restrictions, but no analysis will be performed to ensure that the object is observable along the whole path followed in image space. Finally, the specification of the control tasks corresponding to a previous or a posterior step –or of a set of such tasks– is out of the scope of this thesis and has therefore been left for future work.

6.3.2 Visual control features

The following visual features have been considered for the definition of the control law:

- **Current position of the grasp points in each image.** Each grasp point p_{g_i} is described by two coordinates (u_{g_i}, v_{g_i}) in image space: $(u^r_{g_i}, v^r_{g_i})$ for the right image and $(u^l_{g_i}, v^l_{g_i})$ for the left one. As the computed grasps consists of two points, two pairs of coordinates are available in each image. Therefore, $i \in \{1, 2\}$.

- **Target position of the grasp points in each image.** Each target grasp point $p^*_{g_i}$ is described by two coordinates $(u^*_{g_i}, v^*_{g_i})$ in image space: $(u^{r^*}_{g_i}, v^{r^*}_{g_i})$ for the right image and $(u^{l*}_{g_i}, v^{l*}_{g_i})$ for the left one. There are two pairs of target coordinates in each image, each corresponding to one of the coordinates giving the current position of the grasp points, that is, $i \in \{1, 2\}$.

  Due to the observability restrictions described in section 6.3.1, this target position does not necessarily correspond to the ideal situation of a relative position between the object and the gripper at which the gripper can grasp the object. Instead, it may be a close position to this ideal situation that is compatible with the observability restrictions. As suggested in section 6.3.1, in case that the target grasp points $p^*_{g_i}$ do not correspond to this ideal situation, an additional control task would be required to complete the approximation of the gripper to the grasp points. Anyway, these target coordinates should correspond to the projection of points that belong to the plane where the object lies or to a parallel one.

Therefore, the vector $\mathbf{s_m}$ of image feature parameters is:

$$\mathbf{s_m} = \left[u^l_{g1}, v^l_{g1}, u^l_{g2}, v^l_{g2}, u^r_{g1}, v^r_{g1}, u^r_{g2}, v^r_{g2}\right]^t \tag{6.1}$$

The vector $\mathbf{s_m}$ consists of the coordinates of the points p_{g_i} in image space, expressed in pixels. Based on $\mathbf{s_m}$, a vector $\mathbf{s}$ can be defined with the corresponding coordinates $\mathrm{p}_{g_i} = (x^c_{g_i}, y^c_{g_i})$ ($i \in \{1, 2\}$, $c \in \{l, r\}$) of these points expressed with respect to a 2D coordinate frame associated to the image plane:

$$\mathbf{s} = \left[x^l_{g1}, y^l_{g1}, x^l_{g2}, y^l_{g2}, x^r_{g1}, y^r_{g1}, x^r_{g2}, y^r_{g2}\right]^t \tag{6.2}$$

where:

$$x_{g_i}^c = \frac{d_u^c(u_{g_i}^c - u_0^c)}{F_u^c} \tag{6.3}$$

$$y_{g_i}^c = \frac{d_v^c(v_{g_i}^c - v_0^c)}{F_v^c} \tag{6.4}$$

u_0^c, v_0^c, d_u^c, d_v^c are intrinsic camera parameters of camera c, described in table 4.1, and F_u^c and F_v^c correspond to the definitions given by equations 4.9 and 4.10. This vector $\mathbf{s}$ is used as the vector of control features to be provided as input to the control law. $\mathbf{s}^*$ is the vector of target values for these control features.

6.3.3 Output space of the control law

As it has been indicated in section 6.3.1, the control law has been considered only for the case of 4 d.o.f. In the case, the d.o.f. available are: translations along the X, Y, and Z axes, and a rotation around the Z axis. As the vision system considered in this thesis is composed of two cameras, a transformation is applied internally by the control law filter between each camera and the gripper of the robot, so that the controllable d.o.f. are expressed with respect to a frame attached to the gripper, as required by the virtual actuator in charge of the movement of the robot (see section 6.6). The definition of this transformation is based on the camera configuration parameters described in section 4.1.2.

In this thesis, the robot is considered as a velocity-controlled device. Therefore, the control law will produce a vector $\dot{\mathbf{r}}$ of velocities –the *velocity screw*, or *kinematic screw*. Considering the restrictions mentioned in section 6.3.1, the velocity screw for the 4 d.o.f. case is:

$$\dot{\mathbf{r}} = [V_x, V_y, V_z, \omega_z]^t \tag{6.5}$$

where V_x, V_y and V_z are translational velocities along axes X, Y, and Z, respectively, and ω_z the angular velocity corresponding to the rotations around the Z axis.

A velocity screw $\dot{\mathbf{r}}_6$ for the 6 d.o.f. case would include all translational and rotational velocities:

$$\dot{\mathbf{r}}_6 = [V_x, V_y, V_z, \omega_x, \omega_y, \omega_z]^t \tag{6.6}$$

where ω_x, ω_y are rotational velocities around axes X and Y, respectively.

6.4 Computation of target values for the grasp points

The computation of the target positions of the grasp points in the image space of each camera is carried on by the *target generator* filter introduced in section 3.2. For this computation, a model is proposed in this thesis that describes the target position of the grasp points in the image and 3D coordinate spaces of each camera with respect to a landmark that is given as a reference position. This description is based on a set of parameters that define the target relative location of the grasp with respect to the landmark. The landmark has a description in both the image of each camera, from which the corresponding 3D description with respect to each camera can be recovered using the parameters of that camera. The meaning of this description depends on the context in which the tracking of the object is going to be executed.

In addition to producing the target grasp, the *target generator* filter also verifies its feasibility by checking the observability restrictions on the target position of the object with respect to the cameras. Additional verifications can be performed depending on the information provided by the landmark and other parameters available to the filter.

This filter requires as input the current coordinates of the grasp points in the image of each camera, a description of the object these points belong to and an indication of the status of the grasp tracking. The output is a target position of the grasp points in each image that complies with the observability restrictions stated in section 6.3.1. In addition, the filter also requires, as parameters, calibration data regarding the image acquisition sensor (see section 4.1) and the description of the landmark in the image space of each camera.

The *target generator* filter uses the model mentioned above to compute a target 3D description of the grasp with respect to the cameras. This 3D target position is then projected onto the image space of each camera, in order to obtain the pixel coordinates of the grasp points that the filter produces as output. The feasibility of the target grasp is then verified using the computed 2D and 3D descriptions.

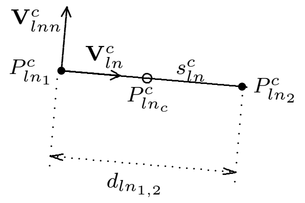
In the current design of this filter, the tracking status given as input is used to decide when the target position of the grasp has to be computed. In particular, this computation is performed only when the tracking status indicates that a new grasp –instead of a previously computed one– has been selected, or when, with a already computed grasp being tracked, that target position has not been computed yet. Otherwise, the last computed target position is produced as output. Consequently, no new target position is computed during the tracking of the same grasp. Only when the tracked grasp is discarded and a new one is selected –either computed or taken from the tracking window–, a new target position of the grasp points is computed.

6.4.1 Model of a landmark for the target grasp

The purpose of the actual landmark used in this thesis is related to the observability restrictions mentioned in section 6.3.1, in particular, to the avoidance of the overlapping in image space between the object and the fingers of the gripper –in case that they could be observed by any of the cameras. With this purpose, a segment, called *landmark segment* has been defined as landmark.

In the image space of each camera, this segment belongs to a line that provides a limit before which the lack of overlapping between object and fingers can be ensured. In fact, it can be considered that the landmark segment divides the image plane of each camera in two subplanes, with the fingers –if observable– appearing in one of these two subplanes only. Let the *free-space image subplane* be the subplane on which the landmark segment ensures that the projection of the fingers does not appear and the *overlapping image subplane* be the other subplane. The overlapping with elements other than the fingers, that is, elements appearing on the free-space image subplane, is not checked.

In the 3D space of the frame c ($c \in \{l, r\}$) each camera, the landmark segment lies on a plane that is parallel to the image planes of the cameras. Its depth along the optical axis of each camera –which, according to the camera model described in section 4.1.1, corresponds to the Z axis of the coordinate frame associated to that camera– specifies the target depth of the grasp points along this axis. Therefore, all points belonging to the plane where the landmark segment lies feature the same value for their Z coordinate with respect to any of



s^c_{ln}:	Landmark segment
$P^c_{ln_1}$ and $P^c_{ln_2}$:	Extrema of the landmark segment
$P^c_{ln_c}$:	Central point of the landmark segment
$d_{ln_{1,2}}$:	Length of the landmark segment
$\mathbf{V}^c_{ln}$:	Direction vector of the landmark segment
$\mathbf{V}^c_{lnn}$:	Direction vector of the normal to the landmark segment

Figure 6.1: Model of the landmark segment. 3D coordinates are related to a camera frame c ($c \in \{l, r\}$).

the camera frames and vectors defined on this plane or on a parallel one have the general form $\mathbf{V}^c = (V^c_x, V^c_y, 0)$. As it does in the case of the image plane, the line containing the 3D landmark segment also divides the 3D plane where it lies in two subplanes, the projections of which in the image space of each camera correspond to the *free-space image subplane* and the *overlapping image subplane*. Let the *free-space 3D subplane* be the 3D subplane projected as the *free-space image subplane* and let the *overlapping 3D subplane* be the 3D subplane projected as the *free-space image subplane*.

The 3D location of this segment with respect to the other two axes of the camera frame depends on its location in image space with respect to the fingers of the gripper when observed from the above target depth. In case that the fingers cannot be observed, the landmark segment should be located near the border of each image that would be closer to the projection of the fingers, had this border not existed.

The 3D length of the landmark segment indicates the maximum allowed 3D distance between the grasp points should not be longer than the maximum opening of the fingers of the gripper. Therefore, the observation of the landmark segment is a mechanism to determine the maximum allowed distance between the grasp points in image space.

A description of the proposed 3D model for the landmark segment is shown in figure 6.1. The coordinates of the components of this model are defined in the 3D space of any of the cameras c ($c \in \{l, r\}$). According to this model, the landmark segment, s^c_{ln}, is defined by two points, $P^c_{ln_1}$ and $P^c_{ln_2}$. In addition to these two points, the proposed model also includes several additional values, with the aim of providing a common basic set of landmark-related elements to the models considered for the computation of the target grasp: (1) $P^c_{ln_c}$ is the central point of this segment; (2) $d_{ln_{1,2}}$ is the length of the landmark segment, that is, the distance between points $P^c_{ln_1}$ and $P^c_{ln_2}$; (3) $\mathbf{V}^c_{ln}$ is the direction vector of this segment, computed considering the direction from the first to the second point; and (4) $\mathbf{V}^c_{lnn}$ is the direction vector normal to the landmark segment, computed so that this vector points towards the subplane where the fingers of the gripper does not appear. All vectors are assumed to be normalized, that is, $\|\mathbf{V}^c_{ln}\| = \|\mathbf{V}^c_{lnn}\| = 1$.

Figure 6.2 shows an example of the relationship between the gripper of a robot arm and a shape that could be used to specify a landmark segment. In this case, the landmark segment is provided by the side of this rectangular shape that is closest to the border of the image where projection of the fingers of the gripper appears.

The landmark segment can be previously known or learned. In either case, this segment

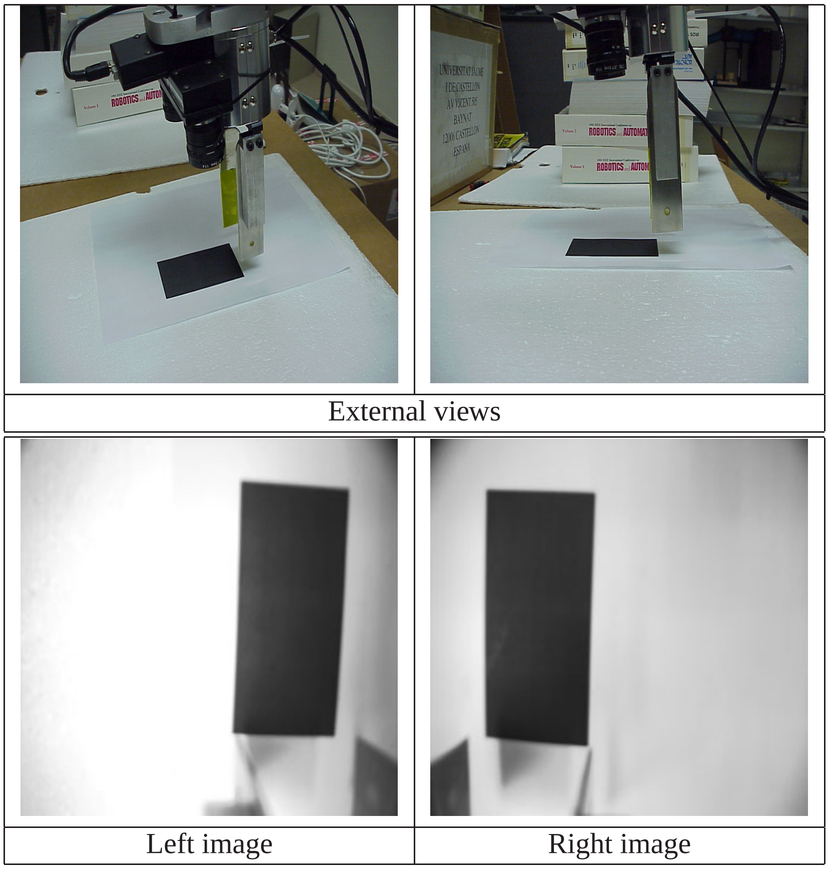


Figure 6.2: Sample shape for the specification of a landmark segment.

will be expressed in the image space of each camera, as required by the target generator filter. The projections $p^c_{ln_1}$ and $p^c_{ln_2}$ image space of the 3D points $P^c_{ln_1}$ and $P^c_{ln_2}$, respectively, will be given as input to this filter, which can use the parameters of the camera to recover the above information regarding this segment.

A simple off-line procedure is proposed in this thesis to learn a description of the landmark segment, expressed as the set $\{p^c_{ln_1}, p^c_{ln_2}\}$, from the images of the template shown in figure 6.2.

6.4.2 Model of the target grasp

The proposed model of the target grasp specifies the set of parameters that have to be computed in order to determine the target position of the grasp points. This model defines a 3D relationship between the target grasp points and the landmark that is given as a reference position. As such a landmark, this model uses the *landmark segment* described in section 6.4.1.

As it is mentioned in section 6.4.1, the line containing the projection of the landmark segment onto the image space of each camera divides this space in two subspaces, the *free-space image subplane* and the *overlapping image subplane*. Regardless of whether the fingers of the robot are observable or not, it is ensured that their projection does not appear in the *free-space image subplane*. Considering this fact, the aim of the target grasp model proposed in this section is to define a relative position between the landmark and the grasp segments that ensures that the projection of the object when that relative position is achieved falls entirely on the *free-space image subplane*.

The components of this model are defined in the 3D space of any of the camera frames $c\,(c \in \{l, r\})$. Nevertheless, through the appropriate coordinate transformations, they could be expressed with respect to any other frame m. All these components lie on the same plane, which is the plane where the landmark segment and its associated set of related elements lie. As it was mentioned in section 6.4.1, all points in this plane have the same depth with respect to the camera frames, and vectors defined on this plane have the general form $\mathbf{V}^c = (V_x^c, V_y^c, 0)$. The 3D point coordinates extracted from this model can be projected, using the camera parameters, onto the image space of each camera in order to obtain their corresponding pixel coordinates.

A description of this model is given in figure 6.3. s_{ln}^c is the landmark segment and $P_{ln_1}^c$, $P_{ln_2}^c$, $P_{ln_c}^c$, $\mathbf{V}_{ln}^c$ and $\mathbf{V}_{lnn}^c$ are some of its related components, which are described in section 6.4.1. According to this model, the target grasp points are $P_{g_1}^{c^*}$ and $P_{g_2}^{c^*}$, and define a *target grasp segment* $s_g^{c^*}$, the central point of which is $P_{g_c}^{c^*}$. In this model, the landmark segment and the target grasp segment are parallel and are aligned at their central points, $P_{ln_c}^c$ and $P_{g_c}^{c^*}$, that is, $P_{g_c}^{c^*}$ belongs to the normal line l_{lnn}^c to the landmark segment at $P_{ln_c}^c$. The distance $d_{ln,g}$ between points $P_{ln_c}^c$ and $P_{g_c}^{c^*}$ coincides with the distance between the landmark segment and the grasp segment, and $d_{g_{1,2}}$ is the distance between the grasp points. The direction vector of the target grasp segment is parallel to the same direction vector as the landmark segment, that is, $\mathbf{V}_{ln}^c$. Whether it follows the same direction or the opposite one, depends on which target grasp point is $P_{g_1}^{c^*}$ and which one is $P_{g_2}^{c^*}$. The indication of this direction is given by a sign parameter, $side_{g_{1,2}}$. For the determination of this direction, the normal line l_{lnn}^c to the grasp and the landmark segments is used as a reference. If $P_{g_1}^{c^*}$ has to be on the same side as $P_{ln_1}^c$, then $side_{g_{1,2}}$ is given the value -1; otherwise, its value is 1.

Therefore, the parameters that have to be provided as input to this model of the target grasp in order to obtain the target grasp points are those highlighted in a box in figure 6.3: $d_{ln,g}$, $d_{g_{1,2}}$, and $side_{g_{1,2}}$. Note that the distance $d_{ln,g}$ between the landmark and the grasp segment has to be selected so that, when the grasp points are placed at the target positions $P_{g_1}^{c^*}$ and $P_{g_2}^{c^*}$, the projection of the object in the image space of each camera must lie entirely in the *free-space image subplane* of that camera, that is, the entire object must be in the *free-space 3D subplane* defined by the landmark segment.

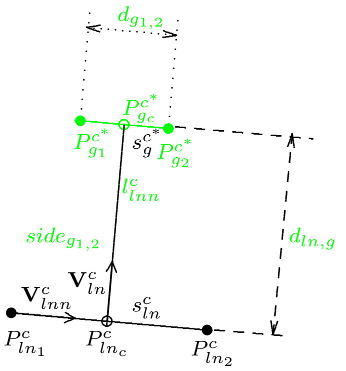
Using the above model parameters and the 3D description of the landmark segment, the target grasp points $P_{g_1}^{c^*}$ and $P_{g_2}^{c^*}$ can be obtained as:

$$P_{g_c}^{c^*} = P_{ln_c}^c + d_{ln,g} \cdot \mathbf{V}_{lnn}^c \qquad (6.7)$$

$$P_{g_1}^{c^*} = P_{g_c}^{c^*} + side_{g_{1,2}} \cdot \frac{d_{g_{1,2}}}{2} \cdot \mathbf{V}_{ln}^c \qquad (6.8)$$

$$P_{g_1}^{c^*} = P_{g_c}^{c^*} - side_{g_{1,2}} \cdot \frac{d_{g_{1,2}}}{2} \cdot \mathbf{V}_{ln}^c \qquad (6.9)$$

Let $P_{g_i}^{c^*} = (X_{g_i}^{c^*}, Y_{g_i}^{c^*}, Z_{g_i}^{c^*})$ the components of the 3D coordinates of target grasp point i



$s^{c^*}_g$: Target grasp segment, with direction vector parallel to $\mathbf{V}^c_{ln}$

$P^{c^*}_{g_1}$ and $P^{c^*}_{g_2}$: Target grasp points

$P^{c^*}_{g_c}$: Central point of the target grasp segment

l^c_{lnn}: Normal line to s^c_{ln} at $P^c_{ln_c}$ following vector $\mathbf{V}^c_{lnn}$

$d_{ln,g}$: Distance between the landmark segment and the grasp segment

$d_{g_{1,2}}$: Distance between the grasp points

$side_{g_{1,2}}$: Indication of whether $P^{c^*}_{g_1}$ is on the same side of l^c_{lnn} as $P^c_{ln_1}$ ($side_{g_{1,2}} = -1$) or not ($side_{g_{1,2}} = 1$)

Figure 6.3: Model of the target grasp. 3D coordinates are related to a camera frame c. The input data required by this model are enclosed in a box.

with respect to camera frame c and $p^{c^*}_{g_i} = (u^{c^*}_{g_i}, v^{c^*}_{g_i})$ the pixel components of their corresponding projection in the image space of that camera ($i \in \{1, 2\}$). Following expressions 4.14 and 4.15, these pixel components can be computed from points $P^{c^*}_{g_i}$ as:

$$u^{c^*}_{g_i} = u^c_0 + \frac{F^c_u X^{c^*}_{g_i}}{d^c_u Z^{c^*}_{g_i}} \qquad (6.10)$$

$$v^{c^*}_{g_i} = v^c_0 + \frac{F^c_v Y^{c^*}_{g_i}}{d^c_v Z^{c^*}_{g_i}} \qquad (6.11)$$

6.4.3 Model of the bounding box of the target object

The proposed model of the bounding box of the target object provides a rough estimation of the surface occupied by the projection of the object in the image space of each camera when the grasp points have been placed at the target positions $P^{c^*}_{g_1}$ and $P^{c^*}_{g_2}$ computed by the target grasp model. This estimation can be used by a conservative –although relatively simple– algorithm to evaluate the observability of the object at its target position, avoiding the need of a more complex procedure for the 3D reconstruction and image projection of the object.

The proposed model specifies the set of parameters that have to be computed in order to obtain an estimation of the bounding box of the object at the above target position. This model defines a 3D relationship between this target bounding box, the target grasp points and the landmark segment, so that the bounding box lies on the *free-space 3D subplane* defined by the landmark segment. In this way, it is ensured that no overlapping will occur between the object and the fingers of the robot at their target relative position.

The components of this model are defined in the 3D space of any of the camera frames c ($c \in \{l, r\}$). All these components lie on the same plane, which is the plane where the landmark segment lies. As it was mentioned in sections 6.4.1 and 6.4.2, all points in this plane have the same depth with respect to the camera frames, and vectors defined on this plane have the general form $\mathbf{V}^c = (V_x^c, V_y^c, 0)$. The 3D point coordinates extracted from this model can be projected, using the camera parameters, onto the image space of each camera in order to obtain their corresponding pixel coordinates.

In the proposed model, the bounding box –also called *aligned bounding box*– is aligned with the grasp segment, so that two sides of this box are parallel to the grasp segment and the other two, perpendicular. Regardless of the actual dimensions of this bounding box, let the *width segments* of this bounding box be the two sides that are parallel to the grasp segment, and the *height segments* the two perpendicular sides. Let the *base segment* of this bounding box the width segment that is closer to the grasp segment.

In addition, each side is tangent to the external contour of the object as some point. Therefore, this bounding box provides a rectangular border of the object in the directions parallel and perpendicular to the grasp segment.

A description of this model is given in figure 6.4. s_{ln}^c, $P_{ln_1}^c$, $P_{ln_2}^c$, $P_{ln_c}^c$, $\mathbf{V}_{ln}^c$, $\mathbf{V}_{lnn}^c$ are components related to the landmark segment, and are described in section 6.4.1. The target grasp segment $s_g^{c^*}$ and its related components –$P_{g_1}^{c^*}$, $P_{g_2}^{c^*}$, $P_{g_c}^{c^*}$, l_{lnn}^c, $d_{ln,g}$, $d_{g_{1,2}}$, and $side_{g_{1,2}}$– are described in section 6.4.2. In this model, the target base segment, $s_{bl}^{c^*}$, belongs to the same line as the landmark segment. The director vector of the target base segment coincides with that of the landmark segment and the grasp segment.

The intersection between the target base segment and the normal line l_{lnn}^c, to the grasp segment at its center coincides with the center of the landmark segment. This intersection splits the target base segment in two parts, of lengths w_{bb_1} and w_{bb_2}. Points $P_{bb_i}^{c^*}$ ($i \in \{1, 2, 3, 4\}$) are the corners of the target bounding box.

Points $P_{bb_1}^{c^*}$ and $P_{bb_2}^{c^*}$ lie on the same side of line l_{lnn}^c as the target grasp point $P_{g_1}^{c^*}$, and points Points $P_{bb_3}^{c^*}$ and $P_{bb_4}^{c^*}$ on the same side as $P_{g_2}^{c^*}$. The target base segment is defined therefore by points $P_{bb_1}^{c^*}$ and $P_{bb_4}^{c^*}$. This side information is provided by the same sign parameter $side_{g_{1,2}}$ considered in the model of the target grasp (see section 6.4.2).

w_{bb_1} is therefore the distance between $P_{bb_1}^{c^*}$ and the center of the landmark segment, that is, the intersection of line l_{lnn}^c with the base segment. Analogously, w_{bb_2} is therefore the distance between $P_{bb_4}^{c^*}$ and the center of the landmark segment. Finally, h_{bb} is the height of the target bounding box.

Therefore, the parameters that have to be provided as input to this model of the bounding box in order to obtain the four corners of the bounding box in its target position are those highlighted in figure 6.4: w_{bb_1}, w_{bb_2}, h_{bb} and $side_{g_{1,2}}$. Note that the same value of $side_{g_{1,2}}$ must be used for both the target grasp and the target bounding box models.

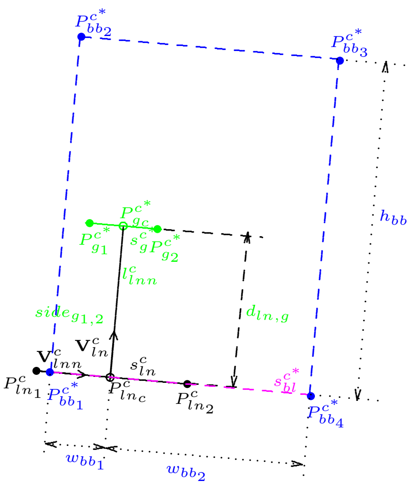
Using the above model parameters and the 3D description of the landmark segment, the coordinates $P_{bb_i}^{c^*}$ ($i \in \{1, 2, 3, 4\}$) of the corners of the bounding box can be computed as:

$$P_{bb_1}^{c^*} = P_{ln_c}^c + side_{g_{1,2}} \cdot w_{bb_1} \cdot \mathbf{V}_{ln}^c \qquad (6.12)$$

$$P_{bb_4}^{c^*} = P_{ln_c}^c - side_{g_{1,2}} \cdot w_{bb_2} \cdot \mathbf{V}_{ln}^c \qquad (6.13)$$

$$P_{bb_2}^{c^*} = P_{bb_1}^{c^*} + h_{bb} \cdot \mathbf{V}_{lnn}^c \qquad (6.14)$$

$$P_{bb_3}^{c^*} = P_{bb_4}^{c^*} + h_{bb} \cdot \mathbf{V}_{lnn}^c \qquad (6.15)$$



$s^{c^*}_{bl}$: Target base segment, with direction vector parallel to $\mathbf{V}^c_{ln}$

$P^{c^*}_{bb_i}$: Corners of the bounding box ($i \in \{1, 2, 3, 4\}$)

w_{bb_1}: Distance from the center of the landmark segment to point $P^{c^*}_{bb_1}$

w_{bb_2}: Distance from the center of the landmark segment to point $P^{c^*}_{bb_4}$

h_{bb}: Height of the target bounding box

$side_{g_{1,2}}$: Same sign parameter used for the model of the target grasp. Indication of whether $P^{c^*}_{bb_1}$ is on the same side of l^c_{lnn} as $P^c_{ln_1}$ ($side_{g_{1,2}} = -1$) or not ($side_{g_{1,2}} = 1$)

Figure 6.4: Model of the target bounding box. 3D coordinates are related to a camera frame c. The input data required by this model are enclosed in a box.

Let $P^{c^*}_{bb_i} = (X^{c^*}_{bb_i}, Y^{c^*}_{bb_i}, Z^{c^*}_{bb_i})$ the components of the 3D coordinates of each corner of the target bounding box with respect to camera frame c and $p^{c^*}_{bb_i} = (u^{c^*}_{bb_i}, v^{c^*}_{bb_i})$ the pixel components of their corresponding projection in the image space of that camera ($i \in \{1, 2, 3, 4\}$). Following expressions 4.14 and 4.15, these pixel components can be computed from points $P^{c^*}_{bb_i}$ as:

$$u^{c^*}_{bb_i} = u^c_0 + \frac{F^c_u X^{c^*}_{bb_i}}{d^c_u Z^{c^*}_{bb_i}} \tag{6.16}$$

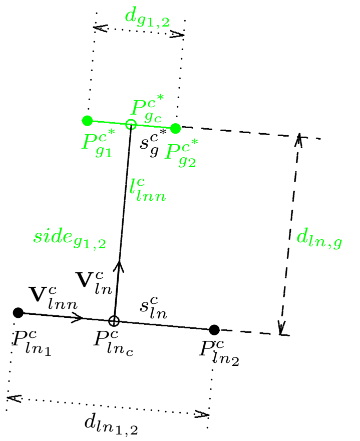
$$v^{c^*}_{bb_i} = v^c_0 + \frac{F^c_v Y^{c^*}_{bb_i}}{d^c_v Z^{c^*}_{bb_i}} \tag{6.17}$$

6.4.4 Model for the feasibility test

The feasibility of the target grasp computed using the model described in section 6.4.2 is evaluated based on the following tests:

- **The grasp-size test.** The 3D distance between the grasp points should not be longer than the length of the landmark segment.

- **The observability test.** According the restrictions indicated in section 6.3.1, the object should be completely observable in the image space of each camera when the target 3D position of the grasp points with respect to the camera frames –which is computed following the model described in section 6.4.2– is achieved.



Data required:
$d_{g_{1,2}}$: Distance between the grasp points
$d_{ln_{1,2}}$: Length of the landmark segment

Test:
$d_{g_{1,2}} \leq d_{ln_{1,2}}$

Figure 6.5: Model for the evaluation of the grasp-size test. 3D coordinates are related to a camera frame c.

Should any of the above tests fail, the result of the feasibility evaluation would be negative.

The grasp-size test

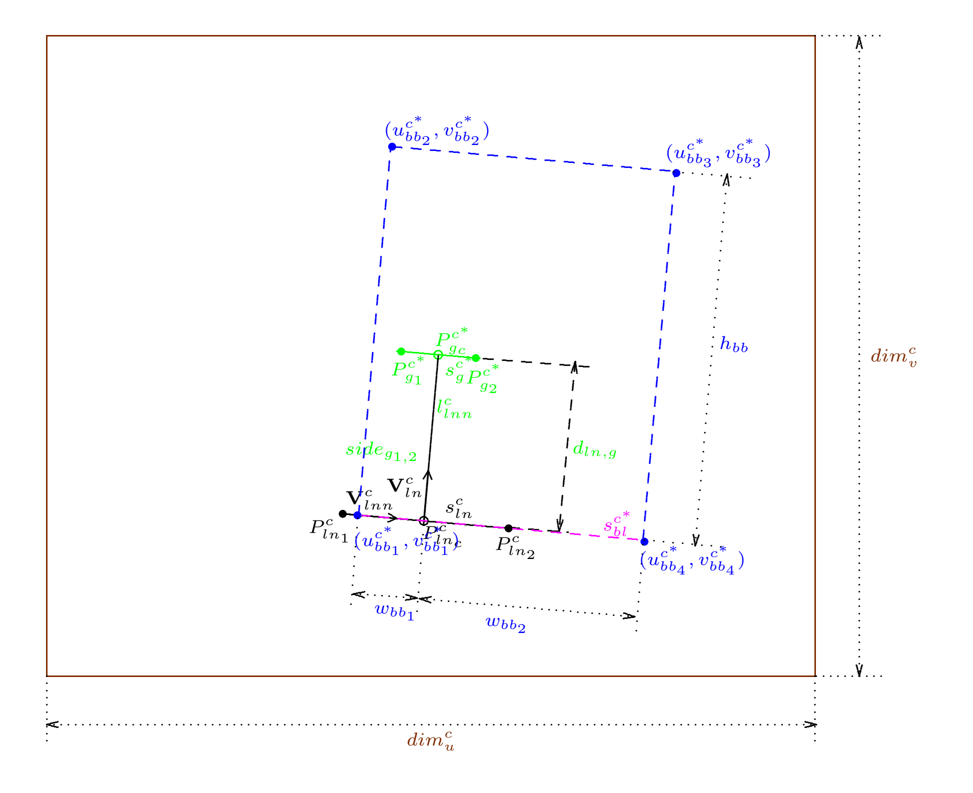
A description of the model used for the evaluation of grasp-size test is given in figure 6.5. The data required by this test are considered expressed in the 3D space of one of the camera frames c. This test uses the distance $d_{g_{1,2}}$ between the grasp points –which is also used in the computation of the target grasp– and the distance $d_{ln_{1,2}}$ between the extremes of the landmark segment –which is part of the data included in the model of this segment. The test is evaluated positively if $d_{g_{1,2}}$ is smaller or equal than $d_{ln_{1,2}}$ and negatively otherwise.

The observability test

Following the specifications given in section 6.3.1, this test is evaluated in the image space of each camera. An strict application of this test would require having not only the 3D target position of the grasp points, but also of all the contour points. However, this would be difficult and time-consuming, due to the lack of already available correspondences between the left and right images of the object other than the grasp points.

To simplify, a bounding box of the object at the target position is computed using the model described in section 6.4.3. This model ensures that there is no overlapping between the object and the fingers in image space, since it produces a target position of the object on the *free-space image subplane*. Therefore, it is only checked if the target bounding box lies within the boundaries of the image.

A description of the model used for the evaluation of this test is given in figure 6.6. As input data, this model uses the pixel coordinates $p_{bb_i}^{c*} = (u_{bb_i}^{c*}, v_{bb_i}^{c*})$ $(i \in \{1, 2, 3, 4\})$ of the corners of the target bounding box in each image –obtained using the model described in section 6.4.3– and the image parameters related to the size of the image, dim_u^c and dim_v^c –described in table 4.3. The test is evaluated positively if the coordinates of the corners of



Data required:

$(u^{c^*}_{bb_i}, v^{c^*}_{bb_i})$:	Coordinates of the corners of the target bounding box ($i \in \{1, 2, 3, 4\}$)
dim^c_u:	Size of the image along the direction of the u coordinate.
dim^c_v:	Size of the image along the direction of the v coordinate.

Test:

$u^{c^*}_{bb_i} \leq dim^c_u$ and $v^{c^*}_{bb_i} \leq dim^c_v$

Figure 6.6: Model for the evaluation of the observability test. Pixel coordinates are related to the image space of camera c.

the target bounding box are not larger than the image dimensions. Otherwise, a negative evaluation is produced.

6.4.5 Computation of the landmark segment

This section describes a simple procedure to obtain the coordinates in image space of the extremes of a landmark segment, which are the description required by the *target generator* filter regarding this segment. This procedure is intended to be executed as an off-line step.

The proposed procedure obtains the coordinates of the landmark segment from the template shape shown in figure 6.2. This shape is a rectangle one of the sides of which has approximately the same length as the maximum opening of the fingers of the gripper. As shown in figure 6.2, this shape has to be observed from a relative position with respect to the cameras so that one of these sides is observed as close as possible to the fingers at the

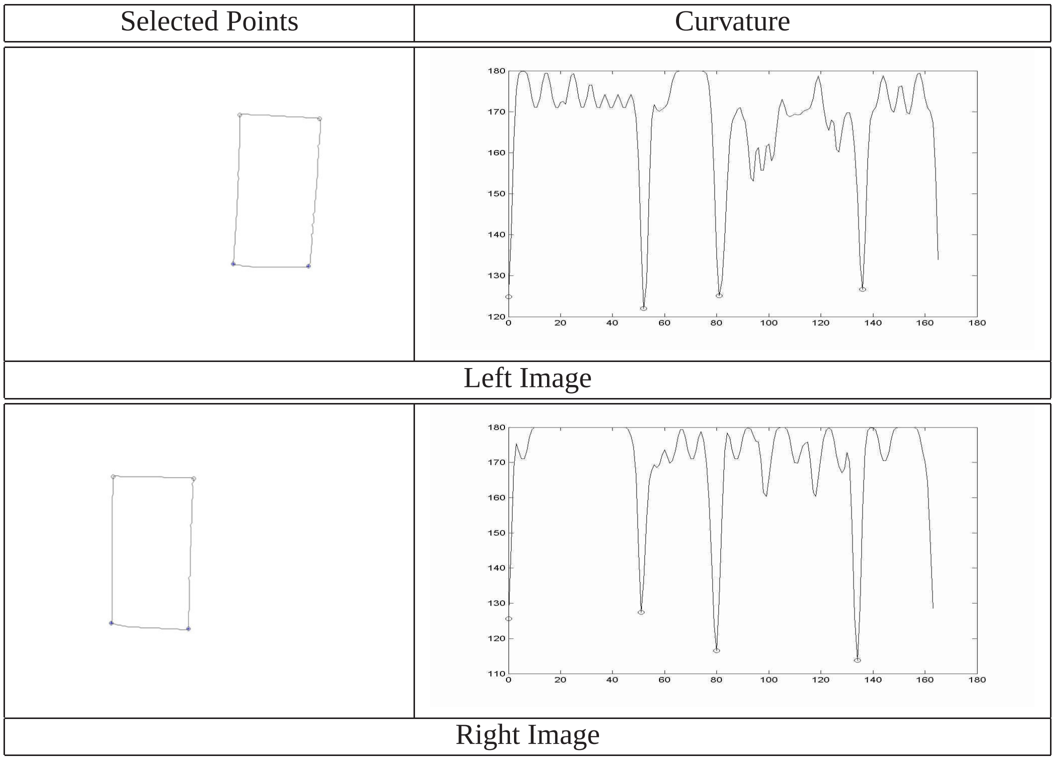


Figure 6.7: Computation of the landmark segment. Corners are shown as circles (o). The extremes of the landmark segment are selected with asterisks (*).

desired depth. This depth is assumed to have been determined a priori.

This procedure takes as input the left and right images of the template shape. Either if the fingers can be observed or not, it should also be given the side of the image that is closer –or would be closer if the non-observable fingers could be observed– to the side of the template corresponding to the landmark segment. In this case, the fingers are closer to the bottom side of the image. The output of this procedure is a pair of pixel coordinates $p^c_{ln_i} = (u^c_{ln_i}, v^c_{ln_i})$ $(i \in \{1, 2\})$ in the image space of each camera c $(c \in \{l, r\})$ corresponding to the extremes of the landmark segment.

The computation performed by this procedure is outlined by algorithm 6.1. This algorithm extracts first the contour of the template shape in each image. This contour extraction can be performed using the capabilities of the *basic visual perception* filter described in section 4.2. The next step is to detect the corners in this contour, since they will allow to analyze the sides of the contour. Such a corner detection has been performed through an analysis of the curvature of the contour, as described in sections 4.4.2 and 5.3.3.

Each side of the contour is defined by two consecutive corners. The proposed algorithm selects the side of the contour that is closest to the image side given as a reference –the bottom side of the image, in this case. This closeness is measured based on the distance as the distance from the central point of the contour side to the line defined by the reference side of the image.

The contour side selected in each image through this procedure is considered to be the projection of the landmark segment in that image. The two corners associated to this side are the extremes of the landmark segment.

Algorithm 6.1 Computation of the landmark segment

```
for each pair of compatible regions do
    Extract the contour of the observed shape
    Extract the four corners of this contour
    Considering two consecutive corners as a side of the contour, select the side that is
    closer to the specified side of the image
    Return the corners that define the selected side
end for
```

Figure 6.7 shows the extraction of corners on the contour of the template shape in each image based the analysis of the curvature. The contour side closest to the bottom of the image is also selected.

6.4.6 Computation and feasibility evaluation of the target grasp

Algorithm 6.2 Computation and feasibility evaluation of the target grasp

```
Compute the target grasp
Compute the bounding box corresponding to the target object position
Evaluate the feasibility of the grasp
```

The computation of the target grasp and the evaluation of its feasibility requires the sequential application of the models of the target grasp and the target bounding box, as well as the model corresponding to the feasibility test, –described in sections 6.4.2, 6.4.3 and 6.4.4, respectively– and the obtention of the data they require as input. This process is summarized by algorithm 6.2.

The complexity of the above algorithm lies mainly in the computation of the input data for the target grasp and bounding box, since the feasibility of the grasp can be evaluated directly using the data provided by the target grasp and the bounding box models. The computation of the target grasp and the bounding box using these models is described by algorithms 6.3 and 6.4, respectively.

Tables 6.1, 6.2 and 6.3 show an example of the application of the above algorithms for the computation of the target grasp points and the target bounding box.

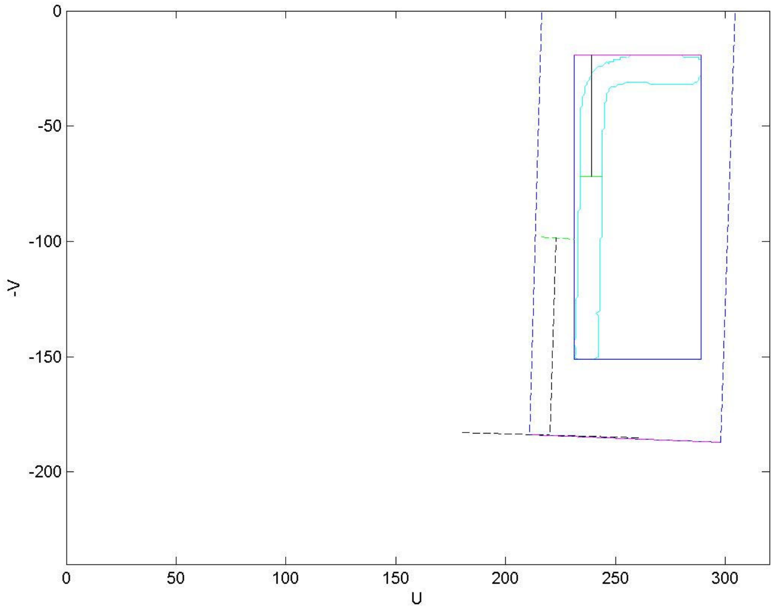
6.5 Computation of the control law

The computation of the control law itself is provided by the *control law* filter described in section 3.2. This filter uses as input a description of the selected grasp for the image provided by each camera of the stereo head, a specification of the target position of the grasp points in each image, and some calibration data regarding the stereo head. Based on this input, the filter produces as output a vector of velocities –or *velocity screw*– to be sent to the robot arm in order to correct the error between the current and the target position of the image points in each camera.

Algorithm 6.3 Computation of the target grasp

```
$\{$Compute the 3D distance between the grasp points, $d_{g_{1,2}}\}$
Compute the 3D coordinates of the grasp from their pixel coordinates in both images
Compute $d_{g_{1,2}}$

$\{$Compute the corners $P^{c^*}_{bb_i}$ ($i \in \{1,2,3,4\}$) of the 3D bounding box aligned with the grasp segment$\}$
Compute the inclination $\alpha_c$ of the grasp line in the image space of camera $c$ ($c \in \{l, r\}$)
Average inclination of the grasp line as $\alpha = (\alpha_l + \alpha_r)/2$
Rotate the object in each image by $-\alpha$ in each image space
Compute the bounding box of each object based on the maximum and minimum values of the components $u$ and $v$ of their pixel coordinates
Rotate the object and the corners of its bounding box in each image by $\alpha$
Use the corners of the rotated bounding box in each image as an estimation of point correspondences and compute their 3D coordinates $P^{c}_{bb_i}$ ($i \in \{1,2,3,4\}$)

$\{$Compute distance between the grasp segment, and the base segment, $d_{bl,g}$, equivalent to the distance $d_{ln,g}$, with the landmark segment at target position ($d_{bl,g} = d_{ln,g}$)$\}$
Select base segment of the bounding box as the closest parallel side to the grasp segment.
Compute $d_{bl,g}$ as the distance from the grasp segment to the base segment

$\{$Compute the sign parameter, $sign_{g_{1,2}}\}$
Build line $l_{gl,bl}$, as the line normal to the grasp segment at the center $P^c_{g_c}$ of this segment, with the direction vector going from $P^c_{g_c}$ towards the base segment
Substitute grasp point 1, $P^c_{g_1}$ on the equation on this line to determine on which side of this line grasp point 1 lies. Make $sign_{g_{1,2}} = -1$ if the result of this substitution is a negative value, and $sign_{g_{1,2}} = 1$ otherwise.

$\{$Apply model to compute the target grasp points$\}$
Apply target grasp model to compute the target grasp points, $P^{c^*}_{g_1}$ and $P^{c^*}_{g_2}$, using $d_{g_{1,2}}$, $d_{bl,g}$ (equivalent to $d_{ln,g}$), and $sign_{g_{1,2}}$
```

Table 6.1: Example of computation of the target grasp points and the target bounding box on an Allen wrench. Current positions are drawn with a solid line and target ones with a dashed line.

Left image


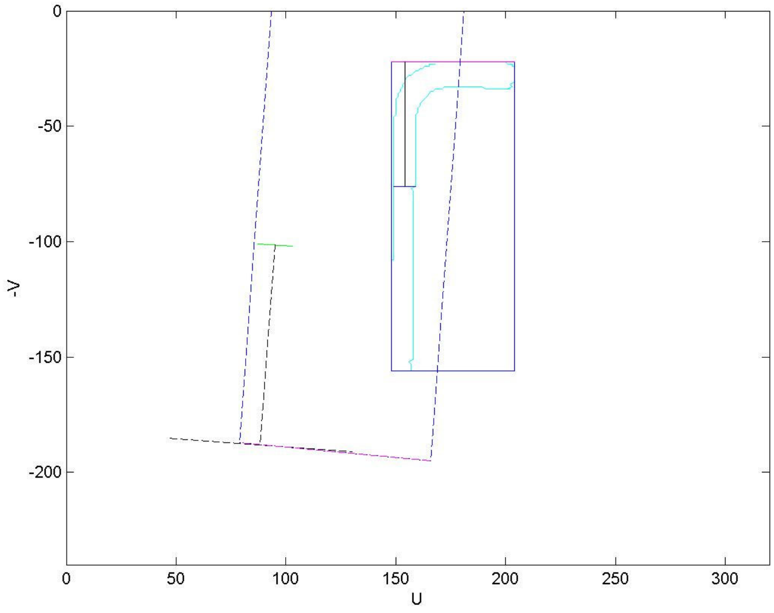
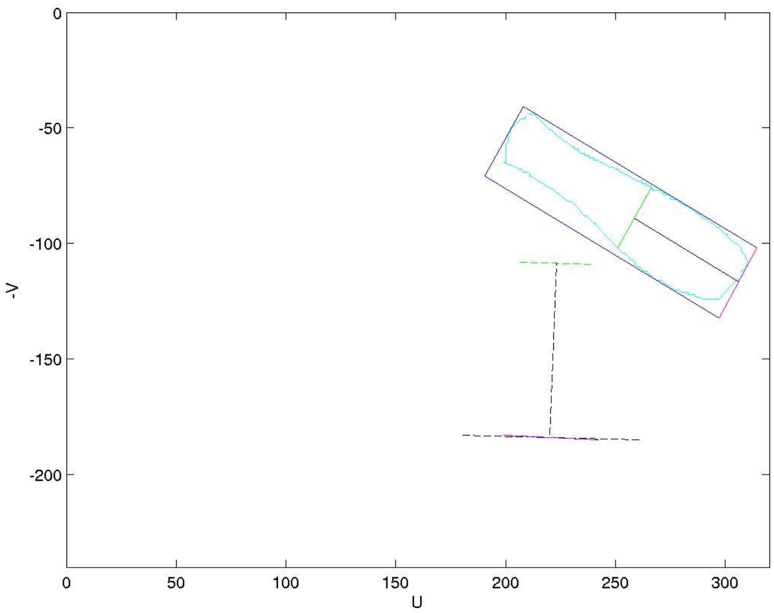
Right image


Table 6.2: Example of computation of the target grasp points and the target bounding box on the handle of a screwdriver. Current positions are drawn with a solid line and target ones with a dashed line.

Left image



Right image

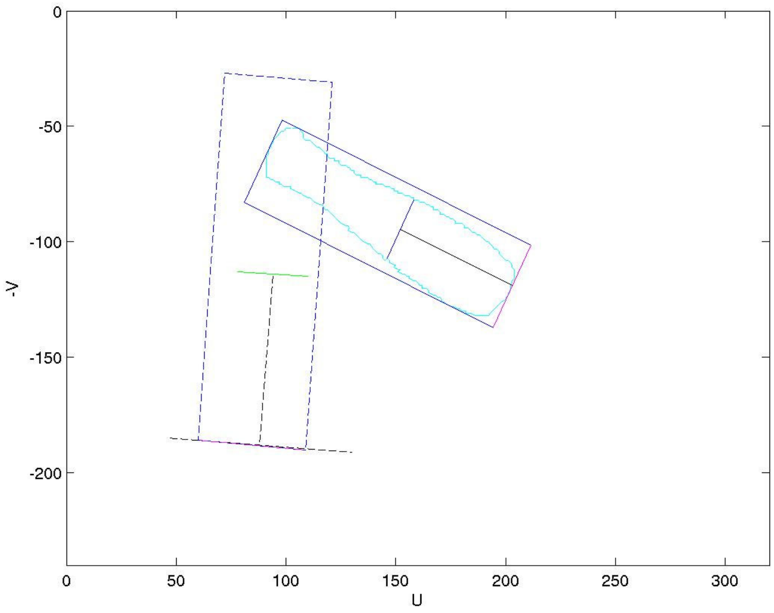
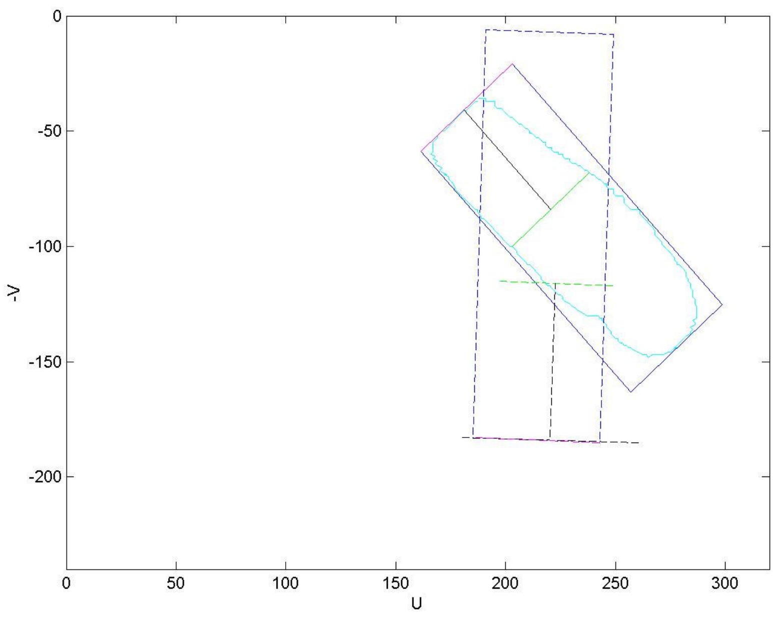
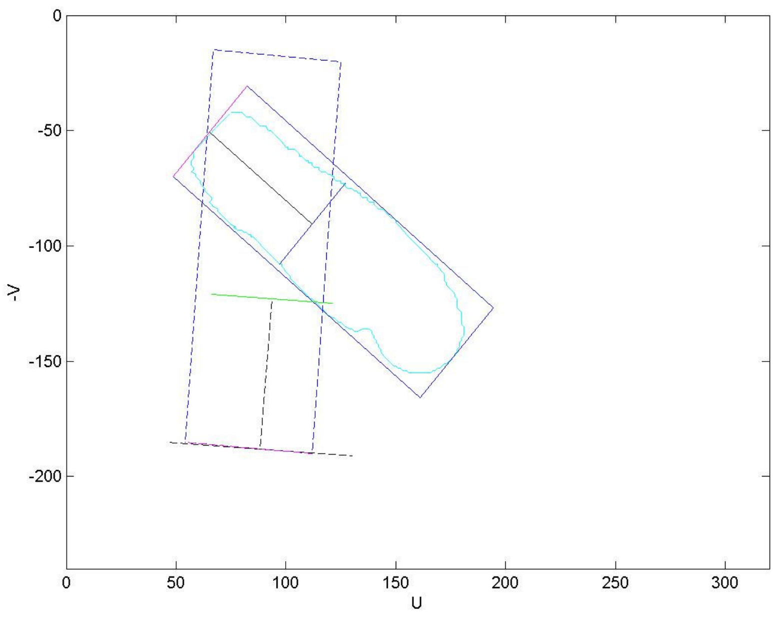


Table 6.3: Example of computation of the target grasp points and the target bounding box on a tool. Current positions are drawn with a solid line and target ones with a dashed line.

Left image

Right image


Algorithm 6.4 Computation of the target bounding box

```
  {Compute distances $w_{bb_1}$ and $w_{bb_2}$}
  Find the intersection $P^c_{bl_t}$ between line $l_{gl,bl}$ and the base segment
  Select extreme 1, $P^c_{bb_1}$, of the base segment as the one that is on the same side as grasp point 1, $P^c_{g_1}$.
  Select extreme 2, $P^c_{bb_2}$, of the base segment as the one that is on the same side as grasp point 2, $P^c_{g_2}$.
  Compute $w_{bb_1}$ as the distance between $P^c_{bl_t}$ and $P^c_{bb_1}$
  Compute $w_{bb_2}$ as the distance between $P^c_{bl_t}$ and $P^c_{bl_2}$

  {Compute $h_{bb}$}
  Compute $h_{bb}$ as the distance between the two sides of the bounding box that are perpendicular to the grasp segment

  {Apply model to compute target bounding box}
  Apply model to compute the corners $P^{c^*}_{bb_i}$ ($i \in \{1, 2, 3, 4\}$) of the target bounding box, using $w_{bb_1}$, $w_{bb_2}$, $h_{bb}$ and $side_{g_{1,2}}$.
```

The computation of the velocity screw with the purpose of correcting the observed error in image space in the position of the grasp points, requires determining the relationship between the velocity screw and the variations it causes in the observations in image space. This relationship, which is the core of the control law, is provided by the *interaction matrix*. In general, the dimensions of the interaction matrix depend on the number of inputs and the number of outputs of the control law. As mentioned in section 6.3, in the task developed in this thesis, 8 control features are used as input (see equation 6.2). Regarding the number of outputs, the restricted case of 4 d.o.f. have been considered. Therefore, in this case, the interaction matrix has dimension 8x4. The development of this matrix and its use within the control law is described in the following sections. In addition, some indications are given for the definition of this matrix and the control law in the complete case of 6 controllable d.o.f.

6.5.1 The interaction matrix

Considering the robot as a velocity controlled device, in IBVS systems, the control law has to compute the necessary velocity screw $\dot{\mathbf{r}}$ that moves the current vector $\mathbf{s}$ of image feature parameters towards the desired vector $\mathbf{s}^*$. Let $\dot{\mathbf{s}}$ the velocity or rate of change of the image feature parameters. A mechanism is needed that can relate $\dot{\mathbf{s}}$ and $\dot{\mathbf{r}}$ is required in order to build the control law. This mechanism is provided by the *interaction matrix* [55]. This matrix can be defined as a linear transformation that maps end-effector velocity to image feature velocities [41]:

$$\dot{\mathbf{s}} = \mathbf{L_s}(\mathbf{r})\dot{\mathbf{r}} \tag{6.18}$$

where $\mathbf{s} \in \mathcal{F} \subseteq \Re^n$, $\mathbf{r} \in \mathcal{T} \subseteq \Re^m$, $\mathbf{L_s}(\mathbf{r}) \in \Re^{n \times m}$, and:

$$\mathbf{L_s}(\mathbf{r}) = \left[\frac{\partial \mathbf{s}}{\partial \mathbf{r}}\right] = \begin{bmatrix} \frac{\partial s_1(\mathbf{r})}{\partial r_1} & \dots & \frac{\partial s_1(\mathbf{r})}{\partial r_m} \\ \vdots & & \vdots \\ \frac{\partial s_n(\mathbf{r})}{\partial r_1} & \dots & \frac{\partial s_n(\mathbf{r})}{\partial r_m} \end{bmatrix} \tag{6.19}$$

This matrix was originally introduced by Sanderson and Weiss [169], who referred to it as the *feature sensitivity matrix*. It has also been referred to as *image Jacobian* [41, 90] or *B matrix* [143, 144].

In order to obtain a velocity screw $\dot{\mathbf{r}}$ from a perceived rate of change $\dot{\mathbf{s}}$ in the image feature parameters, the inverse of the interaction matrix $\mathbf{L_s}(\mathbf{r})$ has to be computed. If $n = m$ and $\mathbf{L_s}$ is nonsingular, the relationship between $\dot{\mathbf{r}}$ and $\dot{\mathbf{s}}$ can be expressed as:

$$\dot{\mathbf{r}} = \mathbf{L_s^{-1}} \dot{\mathbf{s}} \tag{6.20}$$

However, as this is not the general case, a pseudo-inverse $\mathbf{L_s^\dagger}$ can be defined, assuming that $\mathbf{L_s}$ is full rank –that is, $rank(\mathbf{L_s}) = min(n, m)$– [90]:

$$\mathbf{L_s^\dagger} = \begin{cases} \mathbf{L_s^{-1}} & \text{if } n = m \text{ and } \mathbf{L_s} \text{ is nonsingular} \\ (\mathbf{L_s^t} \mathbf{L_s})^{-1} \mathbf{L_s^t} & \text{if } n > m \\ \mathbf{L_s^t} (\mathbf{L_s} \mathbf{L_s^t})^{-1} & \text{if } n < m \end{cases} \tag{6.21}$$

With this definition of $\mathbf{L_s^\dagger}$, the relationship between $\dot{\mathbf{r}}$ and $\dot{\mathbf{s}}$ can be expressed in general as:

$$\dot{\mathbf{r}} = \mathbf{L_s^\dagger} \dot{\mathbf{s}} + (\mathbb{I} - \mathbf{L_s^\dagger} \mathbf{L_s}) \mathbf{b} \tag{6.22}$$

where $\mathbf{b}$ is an arbitrary vector of the appropriate dimension.

In general, $(\mathbb{I} - \mathbf{L_s^\dagger} \mathbf{L_s}) \neq 0$ when $n < m$. In other cases –that is, when $n \geq m$– this expression is evaluated to zero. For $n < m$ and $(\mathbb{I} - \mathbf{L_s^\dagger} \mathbf{L_s}) \neq 0$, all the vectors $(\mathbb{I} - \mathbf{L_s^\dagger} \mathbf{L_s}) \mathbf{b}$ lie in the null space of $\mathbf{L_s}$ and correspond to those components of the object velocity that cannot be observed by the vision system. These vectors are also said to be the *singularities* of the image Jacobian [178]. An analysis of the problems due to the existence of these singularities can be found in [30]. The null space of $\mathbf{L_s}$ is used in hybrid methods (see section 2.1.2) to determine which d.o.f. of the robot can be observable –and therefore, controllable through IBVS– and which cannot [90]. In some IBVS works [16], the visual servo task is complemented with a secondary, non-visual servoed task that uses the non-observable d.o.f.

One of the problems regarding the interaction matrix is the required knowledge of the intrinsic and extrinsic parameters of the camera(s) –which include depth information [104]. For this reason, many published works have used a constant interaction matrix that has been computed at the target pose –that is, the pose at equilibrium– of the robot [16, 55, 121]. Nevertheless, this approach has the drawback that the convergence is only ensured in a neighborhood around the equilibrium pose [41, 104]. Some works have proposed to perform an on-line estimation of the depth [144], or of the whole interaction matrix [151].

6.5.2 Definition of the interaction matrix

As mentioned in section 6.3.2, eight control features have been selected as input to the control task (see equation 6.2). There are four features associated to each camera $c \in \{l, r\}$ of the stereo pair, which correspond to the coordinates $(x^c_{g_i}, y^c_{g_i})$ of each grasp point i in the image acquired by that camera. In addition, according to section 6.3.3, in the case considered in this thesis, four values are provided in an output vector of velocities (see equation 6.5). In this section, the interaction matrix is defined individually for each camera. The two matrices are then combined in a single, global interaction matrix.

In the case considered here, the part of the interaction matrix corresponding to each grasp point $s^c_{g_i} = (x^c_{g_i}, y^c_{g_i})$ in image c is:

$$L_{s^c_{g_i}} = \begin{bmatrix} \frac{1}{Z^c_{g_i}} & 0 & -\frac{x^c_{g_i}}{Z^c_{g_i}} & -y^c_{g_i} \\ 0 & \frac{1}{Z^c_{g_i}} & -\frac{y^c_{g_i}}{Z^c_{g_i}} & x^c_{g_i} \end{bmatrix} \tag{6.23}$$

where $Z^c_{g_i}$ is the depth of grasp point i with respect to frame associated to camera c. In this submatrix, the first row is associated to $x^c_{g_i}$ while the second to $y^c_{g_i}$. The columns correspond, from left to right, to the velocities V_x, V_y, V_z, and ω_z of the output control vector.

For simplicity, $Z^c_{g_i}$ and all $(x^c_{g_i}, y^c_{g_i})$ have been considered at their target values for building the interaction matrix. This makes the response of the control law more stable, but only in a neighborhood of these target values. Therefore, the use of this control law should start once the relative pose between the robot and the object provides image feature parameters within this neighborhood. With these settings, the interaction matrix given by expression 6.23 becomes:

$$L_{s^c_{g_i}} = L_{s^{c*}_{g_i}} = \begin{bmatrix} \frac{1}{Z^{c*}_{g_i}} & 0 & -\frac{x^{c*}_{g_i}}{Z^{c*}_{g_i}} & -y^{c*}_{g_i} \\ 0 & \frac{1}{Z^{c*}_{g_i}} & -\frac{y^{c*}_{g_i}}{Z^{c*}_{g_i}} & x^{c*}_{g_i} \end{bmatrix} \tag{6.24}$$

The interaction matrix for both grasp points 1 and 2 in each image c is defined as:

$$L_{s^c} = \begin{bmatrix} L_{s^c_{g_1}} \\ L_{s^c_{g_2}} \end{bmatrix} \tag{6.25}$$

The relationship between vectors $\dot{\mathbf{s}}$ and $\dot{\mathbf{r}}$ for both images l and r is defined as [121, 116]:

$$\dot{\mathbf{s}} = \begin{bmatrix} L_{s^l} & 0 \\ 0 & L_{s^r} \end{bmatrix} \begin{bmatrix} {}^lM_e \\ {}^rM_e \end{bmatrix} {}^e\dot{\mathbf{r}} \tag{6.26}$$

where ${}^e\dot{\mathbf{r}} = \dot{\mathbf{r}}$ is the velocity screw expressed with respect to the end-effector –the gripper, in this case– of the robot, and ${}^cM_e$ is the transformation between the velocity of camera c and the velocity of the end-effector.

As mentioned in section 4.1.2, it is assumed that the Z axis and the planes defined by the X and Y axes of all these frames are parallel. Following the model described in this section, the translation component of the transformation from the coordinate frame associated to camera c to the frame associated to the gripper is ${}^ct_e = \{{}^ct_{x_e}, {}^ct_{y_e}, {}^ct_{z_e}\}$ and the corresponding rotation matrix is given by ${}^{c}R_e = {}^{c}R_{z_e}({}^{c}\alpha_e)$. Under these circumstances, the transformation matrix ${}^{c}M_e$ is given by:

$${}^{c}M_e = \left[\begin{array}{cccc} & & & {}^{c}t_{y_e} \\ & {}^{c}R_{z_e}({}^{c}\alpha_e) & & -{}^{c}t_{x_e} \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{cccc} \cos{}^{c}\alpha_e & -\sin{}^{c}\alpha_e & 0 & {}^{c}t_{y_e} \\ \sin{}^{c}\alpha_e & \cos{}^{c}\alpha_e & 0 & -{}^{c}t_{x_e} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \qquad (6.27)$$

The above matrix describes the relationship between velocities in the frames of the cameras and the gripper. The rows correspond, from up to down, to the velocities V_x, V_y, V_z, and ω_z respect to the coordinate frame of camera c, while the columns are associated, from left to right, to the same velocities with respect to the frame of the gripper. This matrix has been obtained from the transformation matrix used by Malis [116] and other related works [121], considering the restrictions mentioned above for the relationship between the cameras and the gripper, and the d.o.f. handled by the control law.

Equation 6.26 produces a resulting interaction matrix L_s that can be defined as:

$$L_s = \left[\begin{array}{cc} L_{s^l} & 0 \\ 0 & L_{s^r} \end{array}\right] \left[\begin{array}{c} {}^{l}M_e \\ {}^{r}M_e \end{array}\right] = \left[\begin{array}{c} L_{s^l} \cdot {}^{l}M_e \\ L_{s^r} \cdot {}^{r}M_e \end{array}\right] \qquad (6.28)$$

so that expression 6.26 becomes:

$$\dot{\mathbf{s}} = L_s \dot{\mathbf{r}} \qquad (6.29)$$

In the case of 6 controllable d.o.f., the vector of velocities produced by the control law would be the one given by expression 6.5. The part of the interaction matrix corresponding to each grasp point $s^c_{g_i} = (x^c_{g_i}, y^c_{g_i})$ in image c –given by expression 6.23 in the case with 4 d.o.f.– would be:

$$L^6_{s^c_{g_i}} = \left[\begin{array}{cccccc} \frac{1}{Z^c_{g_i}} & 0 & -\frac{x^c_{g_i}}{Z^c_{g_i}} & -x^c_{g_i} y^c_{g_i} & 1 + x^{c^2}_{g_i} & -y^c_{g_i} \\ 0 & \frac{1}{Z^c_{g_i}} & -\frac{y^c_{g_i}}{Z^c_{g_i}} & -1 - y^{c^2}_{g_i} & x^c_{g_i} y^c_{g_i} & x^c_{g_i} \end{array}\right] \qquad (6.30)$$

As in expression 6.23, the first row of this submatrix is associated to $x^c_{g_i}$ while the second to $y^c_{g_i}$. The columns correspond, from left to right, to the velocities V_x, V_y, V_z, ω_x, ω_y, and ω_z of the output control vector. The global interaction matrix L^6_s is built in a similar way to expressions 6.25 and 6.28, with the transformation matrix ${}^{c}M^6_e$ for the 6 d.o.f. is [116, 121]:

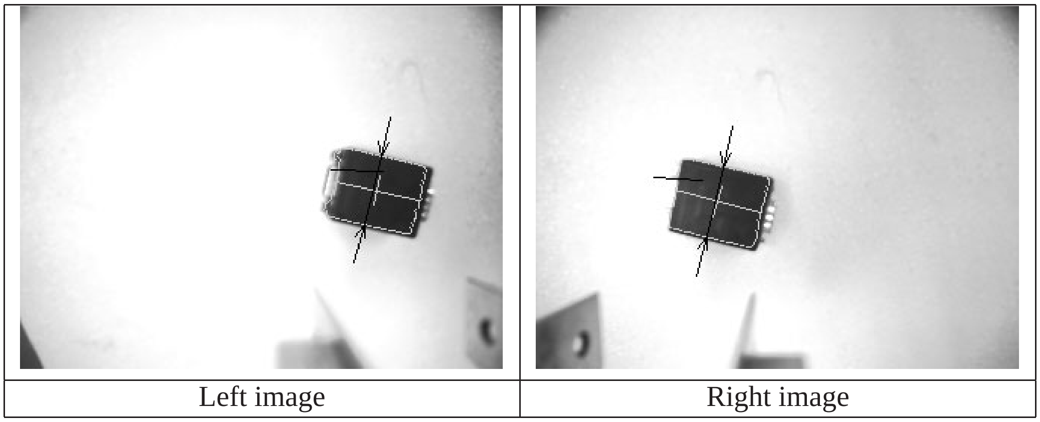
$${}^{c}M^6_e = \left[\begin{array}{cc} {}^{c}R_e & [{}^{c}t_e]_\times \, {}^{c}R_e \\ \mathbb{O}_3 & {}^{c}R_e \end{array}\right] \qquad (6.31)$$

where $\mathbb{O}_3$ is a null square matrix of dimension 3 and $[{}^{c}t_e]_\times$ is the anti-symmetric matrix associated to vector ${}^{c}t_e$.

6.5.3 Design of the control law

A task function $\mathbf{e}$ is defined as:

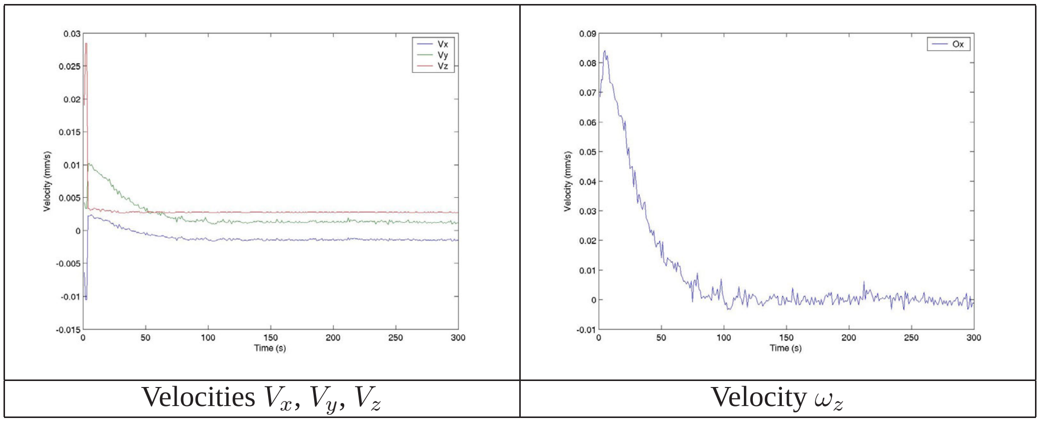
$$\mathbf{e}(t) = L^\dagger_s(\bar{\mathbf{s}}(t) - \mathbf{s}^*) \qquad (6.32)$$



Left image

Right image

Figure 6.8: Initial view and target grasp for a connector.



Velocities V_x, V_y, V_z

Velocity ω_z

Figure 6.9: Evolution of the velocity screw in the execution of a positioning task with respect to a connector.

where $\bar{\mathbf{s}}(t)$ is the smoothed position of the grasp points at time instant t, computed as:

$$\bar{\mathbf{s}}(t) = \frac{\mathbf{s}(t) + \mathbf{s}(t-1)}{2} \tag{6.33}$$

This smoothing has been performed in order to reduce the effect of errors in the position of the grasp points during tracking.

In this thesis, a simple, proportional control law based on this task function has been used. This control law is defined as:

$$\dot{\mathbf{r}}(t) = -\lambda \mathbf{e}(t) = -\lambda L_s^{\dagger}(\bar{\mathbf{s}}(t) - \mathbf{s}^*) \tag{6.34}$$

The definition of the control law would be similar in the case of 6 controllable d.o.f.:

$$\dot{\mathbf{r}}_6(t) = -\lambda L_s^{6^{\dagger}}(\bar{\mathbf{s}}(t) - \mathbf{s}^*) \tag{6.35}$$

Figures 6.8, 6.9 and 6.10 show an example of the execution of this control law for the positioning of the gripper with respect to an object.

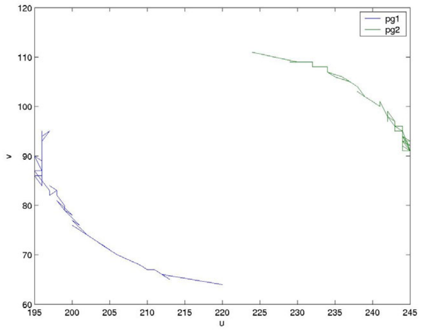


Figure 6.10: Evolution of the position of the grasp points in the execution of a positioning task with respect to a connector.

6.6 Robot movement and data visualization

As it has been already mentioned, in this thesis, the robot is considered as a velocity-controlled device. The sending of movement commands to the robot is performed by the *robot* actuator shown in figure 3.5. This module uses as input a set of velocities expressed with respect to a coordinate frame associated to the end-effector of the robot –the gripper, in this case. As it has been mentioned in section 6.3.3, four d.o.f. related to this frame have been considered in this thesis for the control of the robot: transactions along the X, Y, and Z axes and rotations around the Z axis of the gripper. Therefore, the velocities expected by this module are linear velocities along the X, Y, and Z axes, and the angular velocity around the Z axis. This actuator encapsulates a low-level controller that stabilizes the robot at joint level. Combined with a frame-based controller, as it is the case of this thesis, this module can be used to build a dynamic look-and-move control system.

In addition to the commands to be sent to the *robot* actuator, the proposed behavior for the execution of a grasp also produces several sets of data. These sets can be used for monitoring or debugging the overall operation of the behavior and/or the individual operation of its component filters. With this purpose, some *data viewer* actuators have been developed for the data sets produced by each of the filters. Specialized viewers have been developed for the different data categories –such as images, contours, lines, and grasps– and have been combined appropriately to display the output of each filter. In figure 3.5, the relationship between these actuators and the rest of the task components is shown through a global viewer that encapsulates all developed viewers.

Chapter 7

Conclusions

Final conclusions and some lines for future work are discussed here.

7.1 Summary of the thesis and conclusions

This thesis has considered the problem of the manipulation of objects in a robotic system. In general, this manipulation involves the execution of a sequence of steps and the application of a set of strategies that have often been considered separately in the literature. In addition, an important requirement in many fields of robotics is the ability to perform this manipulation without relying on a predefined model of the object –independently of whether the system has previously used or not a model to identify that object.

The main goal in this thesis has been the definition of a task for the determination, tracking and execution of a grasp on an object without relying on any predefined object model. This task guides a robot arm towards the object, using the visual information provided by a vision system, so that the fingers of the robot are eventually positioned around or close to the points at which the object can be reliably grasped. This thesis has considered the case in which the object is ideally planar –in practice, relatively flat– and the grasp has to be executed with a two-fingered, parallel-jaw gripper. A stereo pair of cameras, mounted on the robot arm, has been used as the vision system.

The above task has been approached as a positioning problem and a visual servoing control loop has been defined to guide the robot towards the object using the data extracted from the images provided by the vision system. The steps to be executed within the control loop, from the extraction of the required visual data to the computation of movement commands for the robot arm, have been identified and each of them has been analysed separately, considering several options in some cases.

In particular, several options have been analysed for the discrimination of the object of interest from the other elements in the observed scene. Procedures for finding the location of stable grasps on the shape of the object and selecting a single one have also been provided. In addition, these grasps have been considered as control features for the definition of the control law that would compute the movement commands for the robot. In order to allow this use of the control points, algorithms have been developed for tracking them along a sequence of images and finding their correspondences within the pair of images provided at each time instant by the vision system. In addition, an analysis has been performed of the restrictions imposed by each particular tracking algorithm on the control law. Moreover, unlike in other control systems, a procedure has been proposed for the automatic computation of the target values for the control features –the grasp points, in this case– used by the control law, and a rule has been defined for the application of this procedure. Additionally, a simple algorithm have been proposed for the off-line computation of some of the parameters required by this procedure.

Finally, an architecture has been defined for the integration of all the above steps within the control loop. The proposed architecture is based on three basic components –virtual sensors, filters, and actuators–, which can be connected to define the control loop. This allows a highly modular design of the control task, easing the development of a distributed implementation of this task and also allowing a smooth integration of new features for supporting more complex tasks. This architecture can be used as a framework for the development of other control tasks of a similar degree of complexity.

The results of the application of the control law have shown the validity of the tracked grasp points as control features, even under noise data extracted from the images provided by the vision system and rough estimations of some of the parameters of the different modules the control loop consists of. The modular design featured by the proposed architecture has made possible to distribute the complexity of the considered control task among its different component modules. This has eased not only the individual analysis and development of each module, but has also made possible to test different strategies in each module without altering the overall design of the control task.

Some of the main contributions of this thesis include: (1) the use of a modular approach to the specification of a control task that provides a basic framework for supporting the concept of behavior; (2) the analysis of several strategies for obtaining a compact representation of the contour of an object; (3) the development of a method for the evaluation and search of a grasp on a planar object for a two-fingered gripper; (4) the specification of different representations of a grasp and the analysis of their use for tracking the grasp between different views of an object; (5) the specification of general algorithms, independent of the actual grasp representation, for the tracking of a grasp along the views of an object obtained from a sequence of single images and a sequence of stereo images; (6) the definition of parametrized models of the target position of the grasp points and of the feasibility of this target grasp, and of an off-line procedure for the computation of some of the reference values required by this model; and (7) the definition of a visual servoing control scheme to guide the gripper of a robot arm towards an unknown object using the grasp points computed for that object as control features and the analysis of the restrictions to be considered for the definition of this control law.

7.2 Publications

The development of this thesis has led to the generation of a number of publications in international conferences. The main publications associated to this thesis cover the following topics:

- **Grasp search and evaluation.**
  - IEEE International Conference on Robotics and Automation (ICRA) (Leuven,

  Belgium, 1998). “Vision-Guided Grasping of Unknown Objects for Service Robots”. Authors: P.J. Sanz, A.P. del Pobil, J.M. Iñesta, and G. Recatalá.

  - Sixth Iberoamerican Conference on Artificial Intelligence (IBERAMIA) (Lisbon, Portugal, 1998). “Computing Contact Stability Grasps of Unknown Objects by Means of Vision”. Authors: A. Morales, P.J. Sanz, G. Recatalá, A.P. del Pobil, and J.M. Iñesta.
  - IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) (Monterey, California, USA, 1999). “Towards a Reactive Grasping System for an Industrial Robot Arm”. Authors: P.J. Sanz, G. Recatalá, V.J. Traver, and A.P. del Pobil.
  - IEEE International Conference on Robotics and Automation (ICRA) (Seoul, South Korea, 2001). “Heuristic Vision-Based Computation of Planar Antipodal Grasps on Unknown Objects”. Authors: A. Morales, G. Recatalá, P.J. Sanz, and A.P. del Pobil.

- **Grasp tracking.**
  - IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Lausanne, Switzerland, 2002). “Visual grasp determination and tracking in 2D dynamic scenarios”. Authors: G. Recatalá, M. Sorg, J. Leupold, P.J. Sanz, and A.P. del Pobil.

- **Computation of polygonal approximations through genetic algorithms.**
  - 11^{th} Scandinavian Conference on Image Analysis (SCIA) (Kangerlussaq, Greenland, Denmark, 1999). “Polygonal Approximations through Genetic Algorithms”. Authors: G. Recatalá and J.M. Iñesta.
  - 15^{th} International Conference on Pattern Recognition (ICPR) (Barcelona, Spain). “Exploring the Performance of Genetic Algorithms as Polygonal Approximators”. Authors: V.J. Traver, G. Recatalá and J.M. Iñesta.

- **Automatic initialization of B-Splines.**
  - 13^{th} Scandinavian Conference on Image Analysis (SCIA) (Göteborg, Sweden, 2003). “Automatic Reconstruction of Silhouettes Using B-Splines”. Authors: S. Glas, G. Recatalá, and M. Sorg.

- **Architectures for robot control.**
  - IEEE International Conference on Systems, Man and Cybernetics (SMC) (Washington D.C. USA, 2003). “Distributed architecture for controlling a manufacturing cell”. Authors: J. Pérez-Aragón and G. Recatalá.
  - IEEE International Conference on Systems, Man and Cybernetics (SMC) (Washington D.C. USA, 2003). “Distributed Agents Control System, a framework for programming distributed agents”. Authors: R. García-Espallargas and G. Recatalá.

An additional publication was published in a national conference:

- **Grasp search and evaluation.**
  - VIII Encuentros de Geometría Computacional (Castelló, Spain, 1999). “Razonamiento Geométrico para la Determinación 2D de Puntos de Agarre” (in Spanish). Authors: P.J. Sanz, V.J. Traver, A. Morales, and G. Recatalá.

7.3 Future work

Work on the grasp-execution task and its framework architecture has opened several lines that could be considered for future work:

- **Architecture for robot control.** The architecture proposed in this thesis has been designed for relatively simple tasks, in which only one action –or *behavior*, as it has been referred to in this work– is executed at a time. This architecture could be extended in several ways:
  - Allowing the execution of several behaviors simultaneously. One or several of the concurrent behaviors could correspond to an intentional action –as such the grasp-execution behavior could be considered–, while the rest could be associated to some automatic action –such as a monitoring of the system in order to avoid hazardous or undesirable situations. In this case, some synchronization mechanism would have to be developed in order to coordinate the concurrent outputs of the behaviors.
  - Allowing the execution of behaviors in a sequence. Complex tasks could be defined as a several behaviors that could be executed in a sequence, or even depending of the state of the system. In this case, a mechanism would have to be defined that would be in charge of the switching between behaviors and of monitoring the conditions under which this switching would be performed.
  - Adding support for active sensors. The virtual sensors described in this thesis are traditional passive sensors. There is no mechanism other than the addition of a filter to on-line modify some parameters or reduce the information provided by a virtual sensor so that only relevant data are produced –for instance, a subwindow of the image space. The set camera-robot when the camera is rigidly attached to the robot can be seen as an active sensor, since movements of the robot modify the perception of the camera. Therefore, one possibility for supporting a sensor in the proposed architecture would be to use a virtual sensor and a virtual actuator. Another possibility could be the use of a virtual filter, since it includes all the interfaces –that is, input, output, and parameter– that the set virtual sensor-virtual actuator has. A filter associated to an active sensor could be considered as a *virtual active sensor* that would join the extremes of a behavior –or simply of a chain of filters–, closing in this way the control loop.
- **Object selection.** The current procedure for the extraction of the contours of an object, based on a previous binarization of a gray-level image, is very sensitive to lighting conditions. Work could be done on the development of more robust object detection procedures, possibly using a combination of different features to detect the

object in image space. New structures for the representation of the shape of the object that could ease this detection should also be analyzed.

- **Grasp search and evaluation.**
  - In this work, a fixed rule has been used to compute a global quality value of a grasp and to select one out of a set of valid grasp. Some research could be carried on a more elaborated definition of this rule based on the definition of several quality metrics that are combined to produce a single overall quality estimation [35, 36]. A learning mechanism could studied to use experience from previously executed grasp in order to obtain a definition and could be optimal for the universe of objects the robot would be manipulating. The works of Morales, Chinellato et al. [36, 129] have proposed several methods to predict the reliability of a grasp from the experience of previously executed ones. Nevertheless, these works have focused only on three-fingered grippers for planar grasp. The adaptation to a two-fingered, parallel-jaw gripper would be necessary.
  - Other grasp selection mechanisms could be studied. For instance, a set of grasps could be initially generated based on a generic set of stability criteria. Other criteria could be applied in a further step in order to reduce this initial set to a set of grasp that would be valid –or simply more suitable– for a given task.
- **Grasp tracking.**
  - The tracking of the grasp points has shown to be quite sensitive to noise during the image processing stages that lead to their computation. This was expected since their obtention requires more data processing than other simpler features –lines or corners, for instance. In fact, the location of the grasp points could be considered as a noisy signal and mechanisms could be explored to reduce the effect of this noise; some of these mechanisms could be:
    - ∗ The use of a smoothing filter on the current location and a list of previous locations.
    - ∗ The use of predictive filters to correct the noise in the current location.
  - Other tracking algorithms could be explored in order to relax the restrictions imposed on the number of degrees of freedom of the control law.
- **Grasp execution.** In this thesis, only visual data have been considered for the execution of a grasp. Nevertheless, this is limited to move the robot to a relative position with respect to the object in which the object can be observed. Other sensor information, such as that provided by contact and force sensors is required to complete the evaluation of the grasp and/or to finish its execution once the robot is in this relative position.

Bibliography

[1] P.K. Allen, A. Timcenko, B.H. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. Technical Report CUCS-034-91, Department of Computer Science, Columbia University, New York, USA, November 1991.

[2] P.K. Allen, A. Timcenko, B.H. Yoshimi, and P. Michelman. Automatic tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, April 1993.

[3] M.A. Arbib. Perceptual structures and distributed motor control. In V.B. Brooks, editor, *Handbook of Physiology: Motor Control*, pages 809–813. The MIT Press, 1981.

[4] M.A. Arbib. Schema theory. In S. Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pages 1427–1443. Wiley-Interscience, New York, USA, 2nd edition, 1992.

[5] R.C. Arkin. Path planning for a vision-based autonomous robot. In *Proc. of the SPIE Conference on Mobile Robots*, pages 240–249, Cambridge (Massachusetts), USA, 1986.

[6] R.C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 264–271, Raleigh (North Carolina), USA, 1987.

[7] R.C. Arkin. Neuroscience in motion: The application of schema theory to mobile robotics. In J.-P. Ewert and M. Arbib, editors, *Visuomotor Coordination: Amphibians, Comparisons, Models and Robots*, pages 649–672. Plenum Press, New York, USA, 1989.

[8] R.C. Arkin. Behavior-based robot navigation for extended domains. *Adaptive Behavior*, 1(2):201–225, 1992.

[9] R.C. Arkin. Modeling neural function at the schema level: Implications and results for robotic control. In R. Ritzmann T. McKenna and R. Beer, editors, *Biological Neural Networks in Invertebrate Neurology and Robotics*, pages 383–410. Academic Press, San Diego (California), USA, 1993.

[10] R.C. Arkin. *Behavior-Based Robotics*. Intelligent Robotics and Autonomous Agents series. The MIT Press, Cambridge (Massachusetts), USA, 1998.

[11] M. Armstrong and A. Zisserman. Robust object tracking. In *Proceedings of the Asian Conference on Computer Vision*, volume I, pages 58–61, Singapore, 1995.

[12] C. Bard, C. Laugier, C. Milési-Bellier, J. Troccaz, B. Triggs, and G. Vercelli. Achieving dexterous grasping by integrating planning and vision-based sensing. *The International Journal of Robotics Research*, 14(5):445–464, October 1995.

[13] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, California, USA, 1987.

[14] G.A. Bekey, H. Liu, R. Tomovic, and W.J. Karplus. Knowledge-based control of grasping in robot hands using using heuristics from human motor skills. *IEEE Transactions on Robotics and Automation*, 9:709–721, December 1993.

[15] A. Bendiksen and G. Hager. A vision-based grasping system for unfamiliar planar objects. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2844–2849, 1994.

[16] F. Berry, P. Martinet, and J. Gallice. Real time visual servoing around a complex object. *IEICE Transactions on Information and Systems*, E83-D(7):1358–1368, July 2000. Special Issue on Machine Vision Applications.

[17] A.S. Besicovitch. A net to hold a sphere. *Math. Gazette*, 41:106–107, 1957.

[18] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 348–353, April 2000.

[19] A. Blake. A symmetry theory of planar grasp. *The International Journal of Robotics Research (Special Issue on Integration Among Planning, Sensing, and Control)*, 14(5):425–444, October 1995.

[20] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, 1998.

[21] A. Blake and A. Yuille. *Active Vision*. MIT, 1992.

[22] C. de Boor. B(asic)-spline basics. In *Fundamental Developments in Computer-Aided Geometric Modeling*, pages 27–49. Academic Press, London, UK, 1993.

[23] E. Boyer and M.-O. Berger. 3D surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3):219–233, 1997.

[24] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, April 1986.

[25] R.A. Brooks. The behavior language: Users' guide. Technical Report 1227, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge (Massachusetts), USA, 1990.

[26] R.A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[27] R.A. Brooks, J.H. Connell, and P. Ning. Herbert: A second generation mobile robot. Technical Report 1016, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge (Massachusetts), USA, January 1988.

[28] A. Castano and S. Hutchinson. Visual compliance: Task-directed visual servo control. *IEEE Transactions on Robotics and Automation*, 10(3):334–342, June 1994.

[29] T. Cham and R. Cipolla. Automated b-spline curve representation incorporating mdl and error-minimizing control point intersection strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):49–53, January 1999.

[30] F. Chaumette. Potential problems of stability and convergence in image-based visual servoing. In D. Kriegman, G. Hager, and S. Morse, editors, *The confluence of vision and control*, volume 237 of *Lecture Notes in Control and Information Sciences*, pages 66–78. Springer-Verlag, 1998.

[31] F. Chaumette, P. Rives, and B. Espiau. Positioning a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 3, pages 2248–2253, Sacramento (California), USA, 1991.

[32] I.M. Chen and J.W. Burdick. Finding antipodal point grasps on irregularly shaped objects. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2278–2283, May 1992.

[33] G. Chesi, E. Malis, and R. Cipolla. Collineation estimation from two unmatched views of an unknown planar contour for visual servoing. In *Proc. British Machine Vision Conference*, pages 224–233, Nottingham, UK, September 1999.

[34] G. Chesi, E. Malis, and R. Cipolla. Automatic segmentation and matching of planar contours for visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 3, pages 2753–2758, San Francisco (California), USA, April 2000.

[35] E. Chinellato. Robust strategies for selecting vision-based planar grasps of unknown objects with a three-finger hand. Msc thesis, Division of Informatics, University of Edinburgh, UK, 2002.

[36] E. Chinellato, R.B. Fischer, A. Morales, and A.P. del Pobil. Ranking planar grasp configurations for a three-finger hand. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1133–1138, Taipei, Taiwan, September 2003.

[37] R. Cipolla and P.J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, Cambridge, UK, 1999.

[38] J. Connell. A behavior-based arm controller. *IEEE Transactions on Robotics and Automation*, 5(6):784–791, December 1989.

[39] J. Connell. A colony architecture for an artificial creature. Technical Report 1151, Massachusetts Institute of Technology, AI Laboratory, August 1989.

[40] P.I. Corke. Visual control of robot manipulators – a review. In k. Hashimoto, editor, *Visual Servoing*, volume 7 of *Robotics and Automated Systems*, pages 1–31. World Scientific, 1993.

[41] P.I. Corke and S.A. Hutchinson. Real-time vision, tracking and control. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 622–629, San Francisco (California), USA, April 2000.

[42] N.J. Cowan and D.E. Koditschek. Planar image based visual servoing as a navigation problem. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 1, pages 611–617, Detroit (Michigan), USA, May 1999.

[43] M.R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, 5:151–165, April 1989.

[44] C. Davidson and A. Blake. Error-tolerant visual planning of planar grasp. In *Proc. 6th Intl. Conf. on Computer Vision*, pages 911–916, 1998.

[45] L. Davis, editor. *Handbook of Genetic Algorithms*. International Thompson Computer Press, 1996.

[46] T. Dean and M. Wellman. *Planning and Control*. Morgan-Kaufmann, San Mateo (California), USA, 1991.

[47] K. Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 705–711, October 1998.

[48] D.F. DeMenthon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, June 1995.

[49] P. Dierckx. *Curve and Surface Fitting with Splines*. Clarendon Press, 1993.

[50] Z. Dodds, M. Jägersand, G. Hager, and K. Toyama. A hierarchical vision architecture for robotic manipulation tasks. In H. Christensen, editor, *Proc. International Conference on Computer Vision Systems*, volume 1542 of *Lecture Notes in Computer Science*, pages 312–330. Springer-Verlag, 1999.

[51] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Transactions on Robotics and Automation*, 14(4):617–622, August 1998.

[52] T. Drummond and R. Cipolla. Visual tracking and control using Lie algebras. In *Proc. Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 652–657, Fort Collings (Colorado), USA, 1999.

[53] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *Proc. 6th European Conference on Computer Vision*, volume 2, pages 20–36, Dublin, Ireland, June/July 2000.

[54] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[55] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–325, June 1992.

[56] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2D objects. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 424–429, Sacramento (California), USA, 1991.

[57] J. Feddema, C. Lee, and O. Mitchell. Model-based visual feedback control for a hand-eye coordinated robotic system. *Computer*, 25(8):21–31, August 1992.

[58] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans. *IEEE Transactions on Robotics and Automation*, 11(1):56–71, February 1995.

[59] O. Fuentes, H.F. Marengoni, and R.C. Nelson. Vision-based planning and execution of precision grasps. Technical Report 546, The University of Rochester, Computer Science Department, New York, 1994.

[60] H. Fujimoto, L.-C. Zhu, and K. Abdel-Malek. Image-based visual servoing for grasping unknown objects. In *Proc. IECON 2000*, pages 876–881, 2000.

[61] A. Galton and R. Meathrel. Qualitative outline theory. In Thomas Dean, editor, *Proc. 16th International Joint Conference on Artificial Intelligence*, pages 1061–1066, Stockholm, Sweden, August 1999.

[62] R. García-Espallargas and G. Recatalá. Distributed agents control system, a framework for programming distributed agents. In *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pages 2563–2568, Washington D.C., USA, October 2003.

[63] E. Gat. ALFA: A language for programming reactive robotic control systems. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1116–1120, Sacramento (California), USA, 1991.

[64] F. Georgsson. Anatomical coordinate system for bilateral registration of mammograms. In J. Bigun and T. Gustavsson, editors, *Image Analysis. 13th Scandinavian Conference, SCIA 2003. Proceedings*, Lecture Notes in Computer Science 2749, pages 353–342. Springer-Verlag, Göteborg, Sweden, June/July 2003.

[65] N. Giordana, P. Bouthemy, F. Chaumette, F. Spindler, J.-C. Bordas, and V. Just. 2D model-based tracking of complex shapes for visual servoing tasks. In M. Vincze and G. Hager, editors, *Robust vision for vision-based control of motion*, chapter 6, pages 67–75. IEEE Press, 2000.

[66] S. Glas. Automatic initialization of B-splines and fitting them to 2D object silhouettes. Master's thesis, Institute for Real-Time Computer Systems, Technische Universität München, Munich (Germany), 2002.

[67] S. Glas, G. Recatalá, and M. Sorg. Automatic reconstruction of silhouettes using B-splines. In J. Bigun and T. Gustavsson, editors, *Image Analysis. 13th Scandinavian Conference, SCIA 2003. Proceedings*, Lecture Notes in Computer Science 2749, pages 239–246. Springer-Verlag, Göteborg, Sweden, June/July 2003.

[68] R.C. González and P. Wintz. *Digital Image Processing*. Addison-Wesley, Reading (Massachusetts), USA, 2nd edition, 1987.

[69] F.C.A. Groan and P.W. Verbeek. Freeman-code probabilities of object boundary quantized contours. *Computer Vision, Graphics, Image Processing*, 7:391–402, 1978.

[70] R. Grupen and T. Henderson. Autochtonous behaviors–mapping perception to action. In T. Henderson, editor, *Traditional and Non-Traditional Robotic Sensors*, volume F-63 of *NATO ASI Series*, pages 867–871. Springer-Verlag, Berlin, Germany, 1990.

[71] A. Guéziec and N. Ayache. Smoothing and matching of 3D space curves. *International Journal of Computer Vision*, 12(1):79–104, 1994.

[72] P. Gullander. *On Reference Architectures for Development of Flexible Cell Control Systems*. PhD thesis, Chalmers Tekniska Högskola (Chalmers University of Technology), Göteborg, Sweden, 1999.

[73] G.D. Hager. A modular system for robust positioning using feedback from stereo vision. *IEEE Transactions on Robotics and Automation*, 13(4):582–595, August 1997.

[74] K. Hashimoto, editor. *Visual Servoing*, volume 7 of *Robotics and Automated Systems*. World Scientific, 1993.

[75] K. Hashimoto, T. Ebine, and H. Kimura. Visual servoing with hand-eye manipulator-optimal control approach. *IEEE Transactions on Robotics and Automation*, 12(5):766–774, 1996.

[76] K. Hashimoto and T. Noritsugu. Performance and sensitivity in visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 2321–2326, Leuven, Belgium, May 1998.

[77] A. Hauck, J. Rüttinger, M. Sorg, and G. Färber. Visual determination of 3D grasping points on unknown objects with a binocular camera system. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 272–278, Kyongju, Korea, 1999.

[78] H. Head and G. Holmes. Sensory disturbances from cerebral lesions. *Brain*, 34:102, 1911.

[79] H. Hexmoor, D. Kortenkamp, R. Arkin, P. Bonasso, and D. Musliner, editors. *Lessons Learned from Implemented Software Architectures for Physical Agents*, AAAI Spring Symposium Series, Stanford (California), USA, March 1995. Working notes.

[80] J. Hill and W.T. Park. Real time control of a robot with a mobile camera. In *Proc. 9th International Symposium on Industrial Robots*, pages 233–246, Washington D.C., USA, March 1979.

[81] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, 1995.

[82] N. Hollinghurst. *Uncalibrated Stereo and Hand-Eye Coordination*. PhD thesis, Trinity Hall, Department of Engineering, University of Cambridge, Cambridge, UK, 1997.

[83] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand eye coordination. *Image and Vision Computing*, 12(3):187–192, April 1994.

[84] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 14(4):525–532, August 1998.

[85] B.K.P. Horn. *Robot Vision*. The MIT Press, Cambridge (Massachusetts), USA, 1986.

[86] I.D. Horswill. Functional programming of behavior-based systems. In *Proc. IEEE Intl. Symposium on Computational Intelligence in Robotics and Automation*, pages 27–34, Monterey (California), USA, November 1999.

[87] J. Hu and H. Yan. Polygonal approximation of digital curves based on the principles of perceptual organization. *Pattern Recognition*, 30(5):701–718, 1997.

[88] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT(8):179–187, 1962.

[89] S.-C. Huang and Y.-N. Sun. Polygonal approximation using genetic algorithms. *Pattern Recognition*, 32:1409–1420, 1999.

[90] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[91] K. Ikeuchi, H.K. Nishihara, B.K.P. Horn, P.G. Sobalvarro, and S. Nagata. Determining grasp configurations using photometric stereo and the PRISM binocular stereo system. *The International Journal of Robotics Research*, 5:46–65, Spring 1986.

[92] J.M. Iñesta. *Algoritmos de Visión Artificial y de Reconocimiento de Patrones para el Estudio Morfométrico del Eje Raquídeo Humano*. PhD thesis, Facultad de Física, Universidad de Valencia, 1994.

[93] J.M. Iñesta, P.J. Sanz, and A.P. del Pobil. An automatic transformation from bimodal to pseudobinary images. In A. del Bimbo, editor, *Image Analysis and Processing*, number 1310 in Lecture Notes in Computer Science, pages 231–238. Springer-Verlag, 1997.

[94] José Manuel Iñesta, Mateo Buendía, and María Ángeles Sarti. Reliable polygonal approximations of imaged real objects through dominant point detection. *Pattern Recognition*, 31:685–699, 1998.

[95] J.W. Jameson and L.J. Leifer. Quasi-static analysis: A method for predicting grasp stability. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 876–883, San Francisco (California), USA, April 1986.

[96] R. A. Jarvis. Automatic grip site detection for robotics manipulators. *Australian Computer Science Communications*, 10(1):346–356, 1988.

[97] Y.-B. Jia. Curvature-based computation of antipodal grasps. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1571–1577, Washington, D.C., USA, May 2002.

[98] L. Kaelbling and S. Rosenschein. Action and planning in embedded agents. In P. Maes, editor, *Designing Autonomous Agents*, pages 35–48. The MIT Press, Cambridge (Massachusetts), USA, 1991.

[99] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2470–2476, Minneapolis (Minnesota), USA, April 1996.

[100] R. Koeppe and G. Hirzinger. Sensorimotor skill transfer of compliant motion. In J. Hollerbach and D. Koditschek, editors, *The Ninth International Symposium of Robotics Research*, pages 239–246, Snowbird (Utah), USA, 1999. Springer-Verlag.

[101] K. Konolige. Small vision systems: hardware and implementation. In Y. Shirai and S. Hirose, editors, *The Eighth International Symposium on Robotics Research*, pages 203–212, Hayama, Japan, October 1997. Springer-Verlag.

[102] J. Kosecka, H. Christensen, and R. Bajcsy. Discrete event modeling of visually guided behaviors. *International Journal on Computer Vision*, 14(2):179–191, March 1995. Special Issue on Qualitative Vision.

[103] D. Kragić and H.I. Christensen. A framework for visual servoing tasks. In E. Pagello, R. Dillman, F. Groen, A. Stentz, and T. Arai, editors, *Intelligent Autonomous Systems 6*, pages 835–842. IOS Press, Venice, Italy, 2000.

[104] D. Kragić and H.I. Christensen. Survey on visual servoing for manipulation. Technical Report ISRN KTH/NA/P-02/01-SE, Department of Numerical Analysis and Computing Science, Stockholms Universitet, Stockholm, Sweden, 2001.

[105] D. Kragić, A. Miller, and P.K. Allen. Realtime tracking meets online grasp planning. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2460–2465, Seoul, Korea, May 2001.

[106] J.C. Latombe. *Robot Motion Planning*. Kluwer, Boston (Massachusetts), USA, 1991.

[107] P. Leitão and A. Quintas. A manufacturing cell controller architecture. In *Proc. of the Flexible Automation and Intelligent Manufacturing Conference*, pages 483–493, 1997.

[108] D. Levine. User guide to the PGAPack parallel genetic algorithm library. Technical Report ANL-95/18, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA, 1996.

[109] T. Lozano-Pérez, J.L. Jones, E. Mazer, and P.A. O'Donnell. *HANDEY: A Robot Task Planner*. Series in Artificial Intelligence. The MIT Press, Cambridge (Massachusetts), USA, 1992.

[110] T. Lozano-Pérez, J.L. Jones, E. Mazer, P.A. O'Donnell, W.E.L. Grimson, P. Tournassoud, and A. Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 843–849, Washington D.C. (USA), March 1987.

[111] Q.-T. Luong and O.D. Faugeras. The fundamental matrix: Theory, algorithms and stability analysis. *International Journal of Computer Vision*, 17:43–75, 1996.

[112] D. MacKenzie, J. Cameron, and R. Arkin. Specification and execution of multiagent missions. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 51–58, Pittsburgh (Pennsylvania), USA, 1995.

[113] C. Malcom and T. Smithers. Programming robotic assembly in terms of task achieving behavioural modules. *Journal of Structural Learning*, 10(2):137–156, 1989.

[114] E. Malis, F. Chaumette, and S. Boudet. Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 1, pages 1352–1359, Leuven, Belgium, May 1998.

[115] E. Malis, F. Chaumette, and S. Boudet. 2D 1/2 visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):234–246, April 1999.

[116] E. Malis, F. Chaumette, and S. Boudet. Multi-cameras visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 4, pages 3183–3188, San Francisco (California), USA, April 2000.

[117] R. Marín, J.S. Sánchez, and P.J. Sanz. Object recognition and incremental learning algorithms for a web-based telerobotic system. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2719–2724, Washington D.C., USA, May 2002.

[118] R. Marín Prades. *The UJI Online Robot: A Distributed Architecture for Pattern Recognition, Autonomous Grasping and Augmented Reality*. PhD thesis, Department of Engineering and Computer Science, Universitat Jaume I, Spain, 2002.

[119] X. Markenscoff, L. Ni, and C.H. Papadimitriou. The geometry of grasping. *The Intl. J. of Robotics Research*, 9(1):61–74, 1990.

[120] D. Marr and T. Poggio. A cooperative computation of stereo-disparity. *Science*, 194(4262):283–287, October 1976.

[121] P. Martinet and E. Cervera. Stacking jacobians properly in stereo visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 717–722, Seoul, Korea, May 2001.

[122] M.T. Mason and J.K. Salisbury. Robot hands and the mechanics of manipulation. In P. H. Winston and M. Brady, editors, *The MIT Press Series in Artificial Intelligence*. MIT Press, 1985.

[123] M.J. Matarić. Behavior-based control: Main properties and implications. In *Proc. of the Workshop on Intelligent Control Systems, International Conference on Robotics and Automation*, Nice, France, May 1992.

[124] M.J. Matarić. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science*, 2(3):82–87, March 1998.

[125] T. Melen and T. Ozanian. A fast algorithm for dominant point detection on chain-coded contours. *Computer Analysis of Images and Patterns*, pages 245–253, 1993.

[126] P.R.S. Mendonça. *Multiview Geometry: Profiles and Self-Calibration*. PhD thesis, University of Cambridge, Cambridge, UK, May 2001.

[127] P.R.S. Mendonça, K.-Y.K. Wong, and R. Cipolla. Epipolar geometry from profiles under circular motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):604–616, June 2001.

[128] D.J. Montana. The condition for contact grasp stability. In *IEEE Intl. Conf. on Robotics and Automation*, pages 412–417, Sacramento (California), USA, 1991.

[129] A. Morales, E. Chinellato, A.H. Fagg, and A.P. del Pobil. Using experience for assessing grasp reliability. In *Intl. Conference on Humanoid Robots (Humanoids 2003)*, Karlsruhe, Germany, October 2003. on CD-ROM.

[130] A. Morales, G. Recatalá, P.J. Sanz, and A.P. del Pobil. Heuristic vision-based computation of planar antipodal grasps on unknown objects. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 583–588, Seoul, Korea, May 2001.

[131] A. Morales, P.J. Sanz, and A.P. del Pobil. Heuristic vision-based computation of three-finger grasps on unknown planar objects. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 1693–1698, Lausanne, Switzerland, October 2002.

[132] A. Morales, P.J. Sanz, G. Recatalá, A.P. del Pobil, and J.M. Iñesta. Computing contact stability grasps of unknown objects by means of vision. In Helder Coelho, editor, *Progreso em Inteligência Artificial. Actas do 6^o Congresso Iberoamericano de Inteligência Artificial*, pages 241–252, Lisboa, Portugal, October 1998. Edições Colibri.

[133] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot. Explicit incorporation of 2D constraints in vision based control of robot manipulators. In P. Corke and J. Trevelyan, editors, *Experimental Robotics VI*, volume 250 of *Lecture Notes in Control and Information Sciences*, pages 99–108. Springer-Verlag, 2000.

[134] M.A. Moussa and M.S. Kamel. An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):239–253, May 1998.

[135] J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. The MIT Press, Cambridge (Massachusetts), USA, 1992.

[136] Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Adisson-Wesley Series in Electrical and Computing Engineering. Reading Mass, Adisson-Wesley, 1991.

[137] V.-D. Nguyen. Constructing force-closure grasps. *The Intl. J. of Robotics Research*, 7(3), 1988.

[138] N. Nilsson. Shakey the robot. Technical Report 323, Artificial Intelligence Center, SRI International, Menlo Park (California), USA, 1984.

[139] F. Noreils and R. Chatila. Plan execution monitoring and control architecture for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(2):255–266, April 1995.

[140] D. Norman and T. Shallice. Attention to action: Willed and automatic control of behavior. In R. Davidson, G. Schwartz, and D. Shapiro, editors, *Consciousness and Self-Regulation: Advances in Research and Theory*, volume 4, pages 1–17. Plenum Press, New York, USA, 1986.

[141] L. Overgaard, B.J. Nelson, and P.K. Khosla. A multi-agent framework for grasping using visual servoing and collision avoidance. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2456–2461, Minneapolis (Minnesota), USA, April 1996.

[142] N. Papanikolopoulos and C. Smith. Computer vision issues during eye-in-hand robotic tasks. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 3, pages 2989–2994, Nagoya, Japan, 1995.

[143] N.P. Papanikolopoulos and P.K. Khosla. Adaptive robot visual tracking: Theory and experiments. *IEEE Transactions on Automatic Control*, 38(3):429–445, 1993.

[144] N.P. Papanikolopoulos, P.K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, 1993.

[145] I. Pavlidis and N.P. Papanikolopoulos. Automatic selection of control points for deformable-model-based target tracking. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2915–2920, Minneapolis (Minnesota), USA, April 1996.

[146] T. Pavlidis and S. L. Horovitz. Segmentation of plane curves. *IEEE Trans. Comput.*, C-23:860–870, 1974.

[147] J. Pérez Aragón. Desarrollo de un sistema de control distribuido para una célula industrial. Final term project, Dept of Computer Science, Universitat Jaume I, Castellón, Spain, September 2003.

[148] J. Pérez-Aragón and G. Recatalá. Distributed architecture for controlling a manufacturing cell. In *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pages 1130–1135, Washington D.C., USA, October 2003.

[149] D. Perrin, C.E. Smith, O. Masoud, and N.P. Papanikolopoulos. Unknown object grasping using statistical pressure models. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1054–1059, San Francisco (California), USA, April 2000.

[150] G.C. Pettinaro. Behavior-based robot program invariance. *Robotica*, 19:217–231, 2001.

[151] J. Piepmeier, G. McMurray, and H. Lipkin. A dynamic quasi-newton method for uncalibrated visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1595–1600, Detroit (Michigan), USA, May 1999.

[152] F. Pla. *Estudios the Técnicas de Análisis de Imagen en un Sistema de Visión para la Recolección Robotizada de Cítricos*. PhD thesis, Facultad de Física, Universitat de València, Valencia, Spain, 1993.

[153] F. Pla, F. Yuste, and F. Ferri. Feature extraction of spherical objects in image analysis. an application to citrus robotic harvesting. *Computers and Electronics in Agriculture*, 8:53–70, 1993.

[154] J. Ponce, D. Stam, and B. Faverjon. On computing force-closure grasps of curved two dimensional objects. *The Intl. J. of Robotics Research*, 12(3):263–273, June 1993.

[155] J. Pretlove and G. Parker. The development of a real-time stereo vision system to aid robotic guidance in carrying out a typical manufacturing task. In *Proc. 22nd ISIR Intelligent Robots and Vision Automation Conference*, pages 21–24, Detroit (Michigan), USA, October 1991.

[156] K. Price. Image segmentation: A comment on studies in global and local histogram-guided relaxation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:247–249, 1984.

[157] D. Proffit and D. Rosen. Metrication errors and coding efficiency of chain-encoding schemes for the representation of lines and edges. *Computer Graphics and Image Processing*, 10:318–332, 1979.

[158] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Comput. Graphics Image Process.*, 1:251–256, 1972.

[159] G. Recatalá and J.M. Iñesta. Polygonal approximations through genetic algorithms. In B.K. Ersbøll and P. Johansen, editors, *Proc. 11th Scandinavian Conference on Image Analysis*, volume 2, pages 707–714, Kangerlussuaq (Greenland), Denmark, June 1999.

[160] G. Recatalá, M. Sorg, J. Leupold, P.J. Sanz, and A.P. del Pobil. Visual grasp determination and tracking in 2D dynamic scenarios. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 25–30, Lausanne, Switzerland, October 2002.

[161] E. Rimon and A. Blake. Caging 2D bodies by 1-parameter two-fingered gripping systems. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1458–1464, Minneapolis (Minnesota), USA, April 1996.

[162] A. Rizzi and D. Koditschek. Preliminary experiments in spatial robot juggling. In R. Chatila and G. Hirzinger, editors, *Proc. 2nd International Symposium on Experimental Robotics*, volume 190 of *Lecture Notes in Control and Information Sciences*, pages 282–298, Toulouse, France, June 1991. Springer-Verlag.

[163] A. Rosenfeld and E. Johnston. Angle detection of digital curves. *IEEE Transactions on Computers*, C-22:875–878, 1973.

[164] A. Rosenfeld and J. S. Wezska. An improved method of angle detection on digital curves. *IEEE Trans. Comput.*, C-24:940–941, 1975.

[165] P. L. Rosin. Techniques for assessing polygonal approximation of curves. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-19:659–666, 1997.

[166] P.L. Rosin and G.A.W. West. Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–1153, December 1995.

[167] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control. The Task Function Approach*. Clarendon Press, Oxford, UK, 1991.

[168] A.C. Sanderson and L.E. Weiss. Image-based visual servo control using relational graph error signals. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 1074–1077, 1980.

[169] A.C. Sanderson, L.E. Weiss, and C.P. Neumann. Dynamic sensor-based control of robots with visual feedback. *IEEE Transactions on Robotics and Automation*, 3(5):404–417, October 1987.

[170] U. Sang, K. Lee, Y.S. Chung, and R.H. Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics and Image Processing*, 52(2):171–190, 1990.

[171] P.J. Sanz. *Razonamiento Geométrico Basado en Visión para la Determinación y Ejecución del Agarre en Robots Manipuladores*. PhD thesis, Universitat Jaume I, Spain, 1996.

[172] P.J. Sanz, A.P. del Pobil, J.M. Iñesta, and G. Recatalá. Vision-guided grasping of unknown objects for service robots. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 3018–3025, Leuven, Belgium, May 1998.

[173] P.J. Sanz, J.M. Iñesta, M. Buendia, and M.A. Sarti. A fast and precise way for computation of moments for morphometry in medical images. In *Proc. of the V Symposium on Biomedical Engineering*, pages 99–100, Santiago de Compostela, Spain, September 1994.

[174] P.J. Sanz, J.M. Iñesta, and A.P. del Pobil. Planar grasping characterization based on curvature-symmetry fusion. *Applied Intelligence*, 10:25–36, 1999.

[175] P.J. Sanz, G. Recatalá, V.J. Traver, and A.P. del Pobil. Towards a reactive grasping system for an industrial robot arm. In *Proc. IEEE Intl. Symposium on Computational Intelligence in Robotics and Automation*, pages 1–6, Monterey (California), USA, November 1999.

[176] R. Schaad. *Representation and Execution of Situated Action Sequences*. PhD thesis, Universität Zürich, Zurich, Switzerland, 1998.

[177] M. Seitz. Towards autonomous robotic servicing: Using and integrated had-arm-eye system for manipulating unknown objects. *Robotics and Autonomous Systems*, 26(1):23–42, August 1999.

[178] R. Sharma and S. Hutchinson. On the observability of robot motion under active camera control. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 162–167, May 1994.

[179] G.C. Shephard. A sphere in a crate. *J. London Math. Society*, 40:433–434, 1965.

[180] R. Shiffrin and W. Schneider. Controlled and automatic human information processing: Ii. *Psychological Review*, 84:127–190, 1977.

[181] K.B. Shimoga. Robot grasp synthesis: A survey. *The Intl. J. of Robotics Research*, 15(3):230–266, June 1996.

[182] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973.

[183] M. Sorg and J. Leupold. Biologically motivated tracking of an object in motion through cue-based directed search. In *Proc IX. Spanish Symposium on Pattern Recognition and Image Analysis*, volume 2, pages 261–266. Universitat Jaume I, 2001.

[184] J. Spooring and R. Arps. Representing contours as sequence of one dimensional functions. In *Proc. of the Fourth Asian Conference on Computer Vision*, pages 742–746, Taipei, Taiwan, January 2000.

[185] K. Stanley, J. Wu, A. Jerbi, and W.A. Gruver. A fast two dimensional image based grasp planner. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 266–271, Kyongju, Korea, 1999.

[186] S.A. Stansfield. Robotic grasping of unknown objects: A knowledge-based approach. *The Intl. J. of Robotics Research*, 10(4):314–326, August 1991.

[187] C. Steger. On the calculation of arbitrary moments of polygons. Technical Report FGBV-96-05, Forschungsgruppe Bilderverstehen, Informatik IX, Technische Universität München, Munich, Germany, October 1996.

[188] K. Tarabanis, P. Allen, and R.Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[189] M. Taylor, A. Blake, and A. Cox. Visually guided grasping in 3D. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 761–766, 1994.

[190] C. H. Teh and R. T. Chin. On the detection of dominant point on digital curves. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-11:859–872, 1989.

[191] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

[192] M. Tonko, J. Schurmann, K. Schafer, and H.-H. Nagel. Visual servoed gripping of a used car battery. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, volume 1, pages 49–54, Grenoble, France, September 1997.

[193] V.J. Traver, G. Recatalá, and J.M. Iñesta. Exploring the performance of genetic algorithms as polygonal approximators. In A. Sanfeliu et al., editor, *Proc. 15th Intl. Conf. on Pattern Recognition*, volume 3, pages 774–777, Barcelona, Spain, September 2000.

[194] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[195] J. Velásquez. When robots weep: Emotional memories and decision-making. In *Proc. of the Fifteenth National Conference on Artificial Intelligence*, pages 70–75, Madison (Wisconsin), USA, 1998.

[196] L.E. Weiss, A.C. Sanderson, and C. Neumann. Dynamic visual servo control of robots: An adaptive image-based approach. *IEEE Journal on Robotics and Automation*, 3(5):404–417, October 1987.

[197] S. Wijesoma, D. Wolfe, and R. Richards. Eye-to-hand coordination for vision-guided robot control applications. *The International Journal of Robotics Research*, 12(1):65–78, February 1993.

[198] W. Wilson. Visual servo control of robots using kalman filter estimates of robot pose relative to work-pieces. In K. Hashimoto, editor, *Visual Servoing*, pages 71–104. World Scientific, 1994.

[199] W. Wilson, C.W. Hulls, and F. Janabi-Sharifi. Robust image processing and position-based visual servoing. In M. Vincze and G. Hager, editors, *Robust Vision for Manipulation*, IEEE Series, pages 163–220. SPIE, 2000.

[200] P. Wunsch and G. Hirzinger. Real-time visual tracking of 3D objects with dynamic handling of occlusion. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 2868–2873, Albuquerque (New Mexico), USA, April 1997.

[201] P.-Y. Yin. A new method for polygonal approximation using genetic algorithms. *Pattern Recognition Letters*, 19:1017–1026, 1998.

[202] B.H. Yoshimi and P.K. Allen. Active, uncalibrated visual servoing. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, volume 4, pages 156–161, San Diego (California), USA, May 1994.

[203] B.H. Yoshimi and P.K. Allen. Visual control of grasping and manipulation tasks. In *Proc. IEEE International Conference on Multisensor Fusion and Integration of Intelligent Systems*, pages 575–582, 1994.

[204] M.F. Zakaria, L.I. Vroomen, P.J.A. Zsombor-Murray, and J.M.H.M. van Kessel. Fast algorithm for the computation of moment invariants. *Pattern Recognition*, 20(6):639–643, 1987.

[205] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Technical Report 2273, INRIA, Sophia-Antipolis, France, May 1994.

[206] H. Zhuang, Z. Roth, and R. Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation of the form AX=YB. *IEEE Transactions on Robotics and Automation*, 10(4):549–554, August 1994.

Index