Universitat Politècnica de Catalunya

Degree's final project

Computing specialization

Integration of task and motion planning for robotics

Author:
Alejandro Suárez Hernández

Director:
Carme Torras Genís
Co-director:
Guillem Alenyà Ribas
Tutor:
Javier Béjar Alonso

April 2016

Abstract

We describe a research project in which we explore the effectiveness of an approach for integrated symbolic and geometric planning in robotics. We target to solve an assembling-like problem with two robot arms. The scenario we propose involves two *Barrett Technology*'s WAM robots that work cooperatively to solve a game for kids. This experiment has a double purpose: setting out a practical challenge that guides our work; and acting as a means to visually validate and show the obtained results. We also cover Project Management aspects such as the temporal planning and the economic, social and environmental analysis.

Describimos un proyecto de investigación en el cual exploramos la efectividad de una estrategia de integración de planificación simbólica y geométrica en el área de la robótica. Nos proponemos resolver un problema equiparable a una tarea de ensamblado mediante dos brazos robot. El escenario que planteamos involucra dos robots WAM de la empresa Barrett Technology trabajando cooperativamente para resolver un juego dirigido a un público infantil. El experimento cumple dos misiones: plantearnos un reto práctico que nos ayude a orientar y guiar nuestro trabajo; y proporcionar un medio visual de demostrar y validar los resultados obtenidos. Adicionalmente cubrimos aspectos típicos de la gestión de proyectos tales como la planificación temporal y el análisis social, económico y medioambiental.

Descrivim un projecte d'investigació on explorem l'efectivitat d'una estatègia d'integració de planificació simbólica i geomètrica en l'àmbit de la robòtica. Ens proposem resoldre un problema equiparable a una tasca de muntatge. L'escenari que plantegem té dos robots WAM de l'empresa Barrett Technology treballant cooperativament per resoldre un joc pels nens. L'experiment compleix dues missions: plantejar-nos un repte pràctic que ens ajudi a orientar la nostra feina; i proporcionar una forma visual de mostrar i validar els resultats obtinguts. A més a més presentem aspectes típics de la gestió de projectes com per exemple la planificació temporal i l'anàlisi social, econòmic i ambiental.

Acknowledgments

Dicen que no hay mejor lengua para maldecir que la materna. En este día y hora todavía es rara la adición de una sección de maldiciones en los artículos y trabajos académicos, así que me gustaría extender la aplicabilidad de esta premisa a los agradecimientos.

Qué menos que empezar dándole las gracias a Carme, Guillem y Javier, que han actuado respectivamente como la directora, el codirector y el tutor del proyecto. Sus comentarios, sugerencias y revisiones han contribuido a que este trabajo haya salido salido adelante. A los tres, ¡muchas gracias!

También querría agradecerle a mi padre todo su apoyo, aunque me haya substraído tantas horas de trabajo con sus llamadas por teléfono para comprobar qué tal me iba.

Gracias también a todos mis amigos, por estar ahí de una forma u otra. En estricto orden aleatorio: Alfredo [sic], Asier, Isaac, Alberto y Núria. Aprovecho para incluir a los integrantes del laboratorio de Manipulació i Percepció, del IRI: Sergi, Nicola y Javier.

Como siempre: nos vemos donde se juntan los caminos.

Alejandro

Contents

List of Figures

List of Tables

Code snippets

Part I

Project management documentation

1 Scope and contextualization

This chapter introduces our project to the reader. Here we provide some background about the involved topics and the state of the art. The project's objectives are also stated. We make clear which is the central line of work, what areas are secondary and which fall outside our scope. Finally, we talk about the most important limitations and risks of the project.

1.1. Introduction

We would like *robots* to gain more autonomy in order for them to perform increasingly complex tasks in diverse domains such as household chores, assistance to old people, maintenance, or even entertainment. *Robotics* is a highly multidisciplinar area which gathers Physics, mechanical engineering, electric engineering, electronic engineering and computer science. Thereby it benefits from the advances and research done in each of these fields.

The project described in this document falls mainly into the field of computer science, and more specifically into artificial intelligence (AI from now on) and algorithmics. It attempts to contribute in the creation of a robust *planning system* which takes into account the inherent difficulties that are present when performing tasks with a robot. These difficulties are namely the uncertainty about the outcomes of the robot's actions, the variability in the measures taken by the sensors and the high computational complexity of the underlying problem: navigating through a world with many objects and interaction possibilities. We study the advantages of filling the gap between *motion planning* (i.e. continuous world representation and geometric algorithms for calculating paths and trajectories) and *task planning* (i.e. planning in an abstract level in which we do not take into account the geometric constraints). These concepts are all reviewed in more detail at section 1.2.1. On a less theoretical level, we will work on an experimental set up that could be considered analogous to an assembling problem. The goal of this experiment is to demonstrate the fruits of the conducted research. For more details, check section 1.3

1.2. Contextualization

In this section we provide an overview of several concepts that are required to fully understand this work and the motivations behind it. This overview includes a definition of the concepts and some background on the topic. Moreover, we offer a justification of the project suitability in its field and its usefulness for future work. Finally, the involved actors are presented.

1.2.1. Related concepts

Robotics

We can find the origin of the term in the Czech noun "robota" which means "labor"[1]. If we were to study in detail the advances and applications of robotics since its origins in the 1950's, we would certainly need a separate article devoted to it. Even defining a robot is a non-trivial task. The International Organization for Standardization describes robots as "automatically controlled, reprogrammable multipurpose manipulators with three or more axes", while The Robot Institute of America defines them as "reprogrammable, multifunctional manipulators designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks". These two definitions are very appropppriate in the industry field in which we can find the roots of robots. Over the years the applications of robots has extended to space exploration (e.g. the Mars Rover), military and law enforcement (e.g. landmine sweepers), medicine (e.g. hip replacement) and entertainment (e.g. Lego MINDSTORMS). There are more visionary definitions and depictions that grant robots human characteristics such as talking and complex human-like behaviour. These are typically given by science fiction writers such as Isaac Asimov. In [Hockstein et al., 2007] and [Stone, 2004] we can find this and much more detailed information about the history of robotics, the influence of mythology and other sociocultural precedents.

Task planning

The first thing we must note is that task planning or simply planning is not inherently related to robotics. Instead, this technique could be studied from a purely algorithmic point of view, and can be used for solving problems without interacting with the physical world. Planning, as understood by practitioners of AI, is an alternative way of solving certain *discrete* problems without incurring in ad-hoc algorithmic solutions. Additionally, it can be viewed as the problem itself, and consequently its computational complexity can be also analyzed. The answer, though, is not unique because is highly dependent on the considered paradigm, the allowed features and the assumptions made over the domain. A planning system, or just planner, typically takes the following input: a *domain*, i.e. the description of the problem and the possible *actions*; and a *problem instance*, i.e. the *initial state* and the *desired (goal) state*. Its output is a sequence of actions that, if applied from the initial state lead to the goal. Alternatively it should tell whether such sequence does not exist.

We can consider, for instance, the problem of "The Towers of Hanoi". A possible ad-hoc solution for the version of the problem in which the pieces are initially located in the first stick could be the classic recursive algorithm. As an alternative, a Breadth First Search navigating through adjacent configurations can also solve the version of the problem in which we start from any arbitrary configuration. A planner is a multipurpose application that receives the description of the problem, in this case the rules of "The Towers of Hanoi", the initial configuration and the end configuration, and uses a general algorithm to solve the problem defined by this three elements. The price to pay is, usually, efficiency.

One of the most widely known paradigms are the ones based on STRIPS (*STanford*

[1]Source: `http://web.archive.org/web/20150415062618/http://capek.misto.cz/english/robot.html`

Research Institute Problem Solver) which constitute what it could be called "Classical planning". These planners use a conjunction of first-order logic well formed formulas to describe the world state. In the domains, each action have associated a list of preconditions and a list of effects over the world state. STRIPS is described in detail in [Fikes and Nilsson, 1972]. The computational complexity of the problems written following the STRIPS convention is studied in [Erol et al., 1995]. On the other hand we have HTNs (*Hierarchical Task Networks*), in which the states are described in an STRIPS-like manner. The difference is that the planner seeks to accomplish tasks, and the domain instructs the planner on how to divide such tasks into smaller subtasks, until arriving to primitive actions (like the ones seen in strips). The operation and complexity of HTNs are described in more detail in [Erol et al., 1996]. It is worth mentioning that, as rule of thumb, the more expressivity we allow in the description of the domains, the greater the complexity of the planning problem. In fact, planning is semidecidable or undecidable under several circumstances.

Motion planning

The task planners described in the previous section are good for deciding sequences of tasks, without worrying about geometrical constraints nor calculating paths and routes. However, the planning problem becomes significantly different when an agent (in our case a robot) has to move in a geometric space. Now, we must provide the adequate instructions so the robot can go from an initial *configuration* in a 2D or a 3D *world* to a final configuration avoiding *obstacles*. When we talk about configuration, we mean the position of the robot in the world (including rotations) as well as the relative pose of its elements (e.g. a movable limb), if applicable. The problem can be now viewed as navigating through a high-dimensional *continuous* world instead of a discrete one. The collection of modeling techniques and algorithms with this goal in mind is what we call motion planning.

The first topic that has to be approached is the *world representation*. It is very important to define an *implicit representation* of the world since the state space is uncountably infinite. There are a vast amount of topics that has been explored to build a solid conceptual infrastructure. These include geometric modeling, both of the robot and of the obstacles; rigid-body transformations, namely translations and rotations; non-rigid transformations; kinematic chains, i.e. combination of joints and links; and topology. All these topics are combined to define and characterize the *configuration space* (C-space) and its properties. This configuration space is the set of all the geometric transformations that can be applied to a robot, or more intuitively, all the positions (including rotations) that the robot can take without intersecting with an obstacle.

Secondly, it is necessary to tackle how the navigation between configuration states is made. Here, the concept of *completeness* enters. An algorithm is said complete if, for every input to the motion planning it correctly reports whether there is a solution and return it in a finite amount of time. Some approaches are based in using a sampling scheme and, thus, are not complete. In these cases, the notion of completennes is sacrificed and other concepts, like *resolution completeness* or *probabilistic completeness* are used instead. There are also complete approaches that build a discrete representation of the world, which (perhaps surprisingly) remains faithful to the original problem.

There exist a huge quantity of material about motion planning, so it is unapprochable to deal with it all in this section. [LaValle, 2006] is an excellent source of information that covers the matter with great detail and from the fundamentals.

1.2.2. Project justification

The idea behind this project originated from the result of research tasks conducted between March and May 2015 as part of the grant conceded by the CSIC (*Consejo Superior De Investigaciones Científicas*[2]) by means of its "Becas Intro 2014" program. The selected research center was the IRI (*Institut de Robòtica i Informàtica Industrial*[3]). Therefore, this project constitutes the culmination of the previous efforts.

The interest of the project is that it contributes to the infrastructure of a continuously developing area such as robotics. Initial applications of robotics, concretely industry, targeted performing repetitive tasks - the so called "Three D" missions (i.e. Dull, Dirty and Dangerous). These tasks are simple enough to be defined programatically, as an algorithm. However, as we extend the range of application of robots, we want them to gain autonomy. It is fairly cumbersome to design an algorithm that explicitly handles each possible input and outcome for all the applications. Planners, in this sense, provide a greater amount of generality and, as a tool, contribute to systematize the process of generating solutions to robotic problems. Moreover, they serve to segregate the problem, its description and the platform which solves it. We can work separately in improving planners, and every system based on them would take benefit. In addition, domain descriptions and problems could be shared between communities.

Until the day, (task) planners have been mostly focused in solving abstract problems like the "Blocks world", elevator control or traffic lights management. While some of this problems could prove to be useful in real applications, they are not used to interact and manipulate in a way we would like a robot to do. One of the greatest handicaps is the gap between task and geometric planning. Several authors recognize this difficulty (like [Kaelbling and Lozano-Pérez, 2013]) and tackle it. Therefore, we think that the project is justified and presents great opportunities.

1.2.3. Involved actors

Targeted audience

The project presented in this document falls mainly in the area of research and experimentation, so its results are not a concrete product to be sold, at least not in short term (see chapter 3 about the economic aspects of the project). Therefore, its targeted audience are other **researchers** and **teachers** rather than possible purchasers. A **private company** or an **university** could also be interested in using this work for the implementation of a commercial product or an academic tool, respectively.

Supervision and guidance

As it has been previously said, the **IRI** takes part in this work. More specifically, the work is supervised by **C. Torras Genís** [4] and **G. Alenyà Riba** [5], from the Perception and Manipulation group. They act, respectively, as the director and co-director of this project.

[2]Website: `http://www.csic.es/`

[3]Website: `http://www.iri.upc.edu/`

[4]Ph.D. in Computer Science from the Universitat Politècnica de Catalunya and member of the CSIC. Bio: `http://www.iri.upc.edu/people/torras/`

[5]Ph.D. from the Universitat Politècnica de Catalunya (Doctor Europeus) and member of the CSIC. Bio: `http://www.iri.upc.edu/people/galenya/`

J. Béjar Alonso [6] also provides guidance for the development of this project as senior professor in the UPC. In addition, we count with the help of **J. Berbegal Mirabent** [7] for feedback and suggestions in regard to the project management documentation.

Development

Finally, **I (A. Suárez Hernández)** am the only person acting as the developer of this project. I am the responsible of gathering all the relevant bibliography for research, of the documentation, design and implementation.

Although I am the only person working in the development process, I perform different roles, and therefore the cost of the human resources is calculated according to the amount of time I spend impersonating each one. This is further discussed in section 3.1.1.

1.3. Problem formulation and objectives

We propose an experimental set-up with the following purpose: introducing pieces whose bases have different geometric shapes in a sphere with cavities that match those pieces (see fig. 1.1). We will focus in doing that using two robot arms that have to act cooperatively following a plan. We shall provide a planning system with geometric and symbolic reasoning capabilities that can compute such a plan. The pieces will be laying initially on a table. One of the robots has to grab the pieces and insert them in the sphere, while the other one has to rotate the sphere in order to facilitate the task. The sphere will be attached to this last robot as an end-effector. Our ultimate goal is to be able of introducing all the pieces, including the most difficult ones (like the trapezoid that can be inserted in only one manner). In this project we will tackle the most simple scenario in which pieces are resting on their base. We we would like the result to be easily extensible for solving this problem even if the pieces start laying on rather complicated poses (stacking and/or on one of their sides). Previous works like those in section 1.5.3 shall come handy. In a deeper level, if this experiment is succesful, it will constitute an additional testimony of the usefulness of planners that combine motion and symbolic reasoning for complex real-life problems. It also presents an opportunity to gather together research made on the topic that has not met yet, and to contribute to robotics with additional results and ideas.

1.4. Scope

Our task will consist mostly of programming and testing. We will be working mainly in the planning system, but we cannot avoid other related areas like perception. Therefore, we shall make the most of currently existing software. Our intention, though, is not to avoid completely writing new procedures for perception. If a problem from this area arises in the course of our work and the existing tools do not completely solve them or have to be slightly adapted, we will deal with them as long as the do not take much time that could be dedicated to the central line of work. The same goes for the geometric representation of

[6] Ph.D. in Computer Science from the Universitat Politècnica de Catalunya and assistant professor there. Bio: `http://www.cs.upc.edu/~bejar/investigacion/investigacion.html`

[7] Ph.D. in Business Administration from the Universitat Politècnica de Catalunya and associate professor at the Universitat Internacional de Catalunya. Bio: `https://www.linkedin.com/pub/jasmina-berbegal-mirabent/42/104/b54`

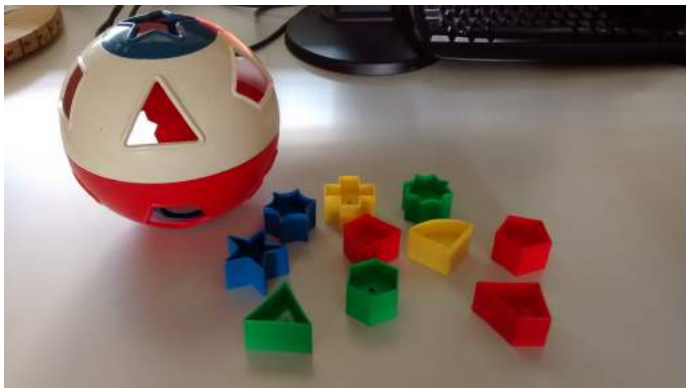

Figure 1.1: The experiment focuses on this game kit. It can be seen as a sphere with several cavities. It comes with a set of pieces, and each one can fit through one and only one of the cavities.

the world and the robots and for automatic learning.

In this sense, we shall employ the ROS environment and several of its packages because they allow fast development and provides a set of standarized, powerful and tested tools and libraries. We are going to make use of simulations for testing our algorithms before executing them in the robot. To do that we have at our disposal the Gazebo simulation tool [8]. In case we need to create some 3D models for the simulation or for another task, we shall employ utilize FreeCAD[9], a 3D CAD suite.

The robots themself are already given. We will make use of two WAM robot arms [10] located at the laboratory of the Perception and Manipulation group. Figure 1.2 shows a picture of one of these robots. We will not deal in much detail with the theory behind step motors and other mechanical topics involved in the construction of our robot, and instead consider them from a programmer perspective. Although the WAM robots are stationary (i.e. their base is fixed in one place), our intention is that the strategy described in this project can be extrapolated to mobile robots.

Figure 1.2: The WAM robot arms located at the IRI Perception & manipulation lab. We have simbolically identified the one with the sphere as CATCHER and the one with the gripper as PICKER.

[8] `http://gazebosim.org/`

[9] `http://www.freecadweb.org/`

[10] `http://www.barrett.com/products-arm.htm`

1.4.1. Potential future applications

As regards to long term aspirations, we expect that the results obtained here will be useful for implementing butlers or household assistants for old people. [Beetz et al., 2011] is a great example of this kind of effort. Another interesting field of application is space exploration. An unmanned exploration vehicle could indeed benefit from this and related works.

The experiment described in section 1.3 could be seen as an assembly taks. Because of this, we think that industry is also a pontentially good candidate for using our work.

1.5. State of the art

This section reviews some of the last related works in the relevant fields. Unlike section 1.2.1, here we mention some of the latest and/or more mature and widely used works in the diverse topics that have already been explored.

1.5.1. Task planning

There is a huge quantity of planners and a lot of research being made in different directions. The ICAPS (*International Conference of Automatic Planning and Scheduling*)[11] celebrates periodically the IPC (*International Planning Contest*) where the most advanced planners participate. With this challenge, the ICAPS seeks to incentivate progress in this area. In addition, they ask for new relevant papers in order to present them in the ceremony. The progress is achieved making the source code of all the contestants available for everyone. This way, they can ensure that future contestants have access to the code of the winner, take its ideas and set the bar higher. No wonder, some of the most advanced planners can be found among the participants of the contest. The winner of the probabilistic discrete track in the MDP (*Markov Decision Problem*) [12] category for the year 2014 was PROST 2014. It is a slightly improved version of a previous planner called simply PROST which won the probabilistic track of the 2011 IPC as well. PROST is based on a UCT-based MCTS[13]. The 2011 version is explained in greater detail at [Keller and Eyerich, 2012]. On the other hand we have Gourmand, or G-Pack as it was known in the 2014 competion of the MDP category. G-Pack was second in the 2014 IPC, but when it first came out just after the 2011 contest ended, it outperformed PROST in all the 2011 problem intances. Gourmand uses LRTDP (*Labeled Real Time Dynamic Programming*, see more in [Bonet and Geffner, 2003]). Gourmand can be viewed with greater detail in [Kolobov et al., 2012].

We will also talk the planners that have participated in the POMDP (*Partially Observable MDP*)[14] category, which could be considered more appropriate for being applied in robotics, since the information gathered by the sensors and the robot actuators (e.g. motors) have certain degree of variability and can introduce uncertainty in the state of the robot. The 2014 winner of the contest was POMDPX_NUS by Nan Ye, Kegui Wu, Meng Zhang, David Hsu and Wee Sun Lee. The authors combined the work made on

[11] Their website can be found at `http://www.icaps-conference.org/`

[12] Discrete process with probabilistic outcomes, but no uncertainty about the current state (i.e. full observability)

[13] UCT stands for Upper Confidence Bound and MCTS stands for Monte Carlo Tree Search. For the interested reader, one of the most exhaustive reference sites about Monte Carlo Tree Search is `http://mcts.ai/index.html` by Cameron Browne (Imperial College London)

[14] Similar to MDP, with the difference that there also exists uncertainty about the current state.

two previous state-of-the-art algorithm, which are explained in [Somani et al., 2013] and [Silver and Veness, 2010].

The ICAPS helps to spread new planning language specifications too. The objective of such languages is, on the one hand, standarizing the way in which problems are described. Since the languages determines the expressivity power, new ones serve to push the limits further and incentivate that planners support increasingly advanced characteristics. The last of these languages is RDDL (Relational Dynamic Influence Diagram Language), by S. Sanner. Its aim is to facilitate the inclusion of both concurrency and stochastic effects in the problem domains. [Sanner, 2010] is a description of the language with a few examples of its usage.

1.5.2. Motion planning

As stated in section 1.2.1, there exist several approaches that deal with the problem of motion planning. One of the most widely known and effective algorithms is RRT (*Rapidly Exploring Random Trees*), and it belongs to the family of Sampling-Based Motion Planning. A parameter-less version of RRT is explained in [LaValle, 2006]. The paper that first described RRT (with a step size parameter) is [LaValle, 1998]. If the robot accepts multiple queries and the configuration space is assumed to be the same (i.e. there shall not be new obstacles) it makes sense to build what is called a *roadmap*. In [Bohlin and Kavraki, 2000] it is described how the lazy PRM (Probabilistic Road Mapping) does so.

We would also like to mention the Robot Operating System or simply ROS [15]. ROS (described further in [Quigley et al., 2009]) is not a typical operating system like GNU/Linux or Unix would be. Instead, it is a software framework for writing robot software. It gathers several libraries, tools and conventions in an attempt to make prototyping and development faster and easier. Among other things, provides languages and packages for describing the geometric structure of a robot, methods for keeping track of the variations of the robot coordinate frame over time, utilities for estimating the pose of the robot and a packet called OMPL (*Open Motion Planning Library*) that has a collection of many state-of-the-art algorithms. This collection includes many versions of RRT and PRM. More about OMPL can be found in [Şucan et al., 2012]. ROS and OMPL are both free software.

1.5.3. Related work

Here we describe some of the previous work that has lead to successful experiments involving a robot performing miscellaneous tasks.

In the first place [Kaelbling and Lozano-Pérez, 2013] has been one of the most invaluable sources of inspiration. It describes an strategy for integrating motion and task planning in belief space. The notion of HPN (Hierarchical Planning in the Now), which is explained more deeply in [Kaelbling and Lozano-Pérez, 2011] is intensively used. It roughly consists of decomposing tasks at multiple level of abstraction (e.g. a task "Make a telephone call" has a higher level of abstraction than "Grab the speaker" and "Dial a number"), so its plans are always aimed at achieving small objectives each time (short horizon). Also, the C-space is a space of distributions of probability instead of a typical geometric space. This tries to model the uncertainty about the current state, so the position of the robot and the surrounding objects is not taken for sure, but with certain probability.

[15] Website: `http://www.ros.org/`

From the same authors we have [Lozano-Pérez and Kaelbling, 2014]. The authors propose a methodology based on performing task planning while postponing the calculation of routes. The sequence of actions obtained this way is the plan skeleton. They use a CSP (Constraint Satisfaction Problem) to force that the pose of the robot and the objects in the world is adequate so the primitive actions *Pick* and *Place* can be executed.

Another interesting work is the one described in [Beetz et al., 2011]. Here the authors described how they managed that two robots make pancakes cooperatively from a recipe downloaded from the World Wide Web[16]. This recipe is used to generate a plan that is later followed by the robots.

In [de Silva et al., 2013] it is discussed the design and implementation of a geometric task planning for a robot, capable not only of calculating routes but also of reasoning about the objects of the world. The authors make use of the HTN formalism to do so.

1.6. Methodology and rigour

Here we describe the workflow in general terms. This includes the guidelines and monitoring tools which together form the methodology. The goal behind this methodology is being able to face contingencies and define the development and validation process.

1.6.1. Methodology

The project is composed, basically, of a research part, the documentation process and a design and implementation phase. The documentation will take care of the planning and will be updated periodically reflecting the project's progress. The research is done before and during the two other parts. The project itself is divided in several functional components. In the design and implementation of these components we shall apply an iterative and incremental development process. A normal development process contains the following elements:

- Analysis of requirements & planning
- Design
- Implementation
- Testing
- Validation & verification of results

We will iterate periodically during the other phases, starting from a very basic solution that can solve a simplified version of the problem. An example of such simplifications could be removing certain pieces and avoiding initial distribution of the world objects that can make the completion of the task rather difficult for the robots. Then we shall make our way to the goal proposed in section 1.3 improving the system incrementally. To learn more about the project's execution we recommend to read chapter 2 about the temporal planning.

[16]The following video is provided as a demonstration: `https://www.youtube.com/watch?v=gMhxi1CJI4M`

1.6.2. Monitoring tools

Before testing in the real robots we will see how the developed algorithms behaves in a simulation environment (see section 1.4). Thus, the simulator becomes an important tool in the validation of the results. It also allows us to test the algorithm in other scenarios that we cannot reproduce, either because we do not have the necessary resources or because it could seriously compromise the integrity of the involved equipment. However testing other scenaries is secundary and we will focus on our experiment.

On the other hand, the experiment itself is a very powerful demonstration of the requirements' accomplishment on its own. At the end of each development iteration we will perform a test using the physical robot arms. These test will be a testimony of the project's progress and, at the end, of its success. To see more about this we recommend checking chapter 2 about the temporal planning. More specifically, figure 2.1 shows when we are planning to conduct these tests.

Finally we count with the help and feedback of the supervisors who have been introduced in section 1.2.3.

1.7. Limitations and risks

In this section we make clear which aspects can be considered a limitation of our project, so they can be addressed in future work. We also talk about the possibility of some hindrances arising. While the amount of technical obstacles is too large, we propose a general policy to handle them.

1.7.1. Limitations

This project does not converge to a product to be sold. While this is intended, it can be seen as a limitation since the inverted time and resources are not economically compensated, or at least not in short term. On the other hand, we have to understand that out intend here is not to concede robots human-like behaviour. Instead, we are seeking to contribute in building a more adequate platform for tasks that are purely mechanical. An additional limitation is that it does not solve a current need, at least not directly. Our intention, instead, is that our work serves in the future for developing products that help industry and the general public.

1.7.2. Risks

If we were to talk about the risks of the project, one of the most importants is that it has a scientific component. We are not trying to combine widely accepted and mature knowledge together to create a product. Instead, we want to prove a point, and it could happen that we fail at this. We must not forget that, despite the extraordinary advances since its foundation, robotics is continually developing and finding new ways to solve problems more efficiently, and we are near of the frontier between old and new knowledge. However we think that, given the feats that have been achieved in previous works (see 1.5.3), our project is realistic and feasible in the given amount of time. Thus, the risk of failing in achieving our goals is very low. If that happened, it would seriously affect the outcome of the whole project. If we were in this situation, our contingency plan would be gathering all the results and finding the root of the problem so they can be fixed in the future.

On a minor scale, there are other risks that could affect the project in a less serious way. For example, it could happen that the sensors and visualization algorithm cannot identify correctly the pieces' shapes. Should this happen, we would have to look for the root of the problem. Maybe the colour of certain pieces prevents them for being correctly distinguished from the background? Could it be that we cannot find an algorithm that correctly distinguishes two pieces when they are too close? Once the problem has been identified, we would have to think of a workaround. For our two example questions, possible answers could be: change the colour of the background or remove or the pieces that cannot be identified; let always a separation between pieces or use a library of forms to execute a matching algorithm. Depending on the nature of the problem and the chosen workaround, the objective could be more or less affected. We anticipate that there will be several issues during the execution of the project, but we think that the probability of one of these preventing the project from being completed is very low. If an extreme situation that cannot be solved in a reasonable term arises, there is always the option of relaxating the requirements so they can be satisfied more easily. However we will do our best to stick to the plan and accomplish the objectives.

2 Planning

This chapter presents an updated version of how we have decomposed the project into tasks. We talk about the terms of each task. We define the action plan of the project as well. The action plan gives some details about the strategy we have followed in addition to the methodology presented in section 1.6 (*Methodology and rigour*).

2.1. Schedule baseline and work breakdown structure

In this section we show the updated top-down decomposition of the whole project into tasks and subtasks. This is also known as the *work breakdown structure* or WBS. After that, we discuss the project's milestones. Then in the WBS dictionary we talk in more detail about the work packages. In order to see more clearly the chronology and task dependency we recommend checking figure 2.1.

2.1.1. Work breakdown structure

The detailed decomposition is shown in table 2.1. As we can see, we have considered the *Research and learning* phase relevant enough to appear as an additional task. The documentation of the project can be divided into two categories: the PM (*Project Management*) documentation, which is redacted at the begining of the project; and the technical documentation, which belongs to the *Final stage* task. Later it will be shown that the *research and learning* and the *PM documentation* shall be performed simultaneously, since they are rather compatible tasks that can be alternated during the first weeks.

As explained in the previous document in section 1.6 (*Methodology and rigour*), we will work on several functional components, and in each one we will follow an iterative and incremental methodology. We have defined the following modules: **perception pipeline**, in which we tackle the problem of identifying the pieces and the sphere cavities on an image; **simulation assets**, in which we prepare a basic simulation environment for benchmark and visualization of results; **world interface** which is devoted to build the interface between the planner and the real world (i.e. estimation of the state and execution of primitives); and **symbolic planning integration** which consists of integrating a symbolic planner with ROS.

Then there is an entire phase dedicated to experimentation and gathering results and, at the end, there is what we have called the *Final stage* which contains the technical documentation (as it has been said formerly) and the project presentation preparations.

The terms for each task can be seen in sections 2.1.3 and in the Gantt diagram (figure 2.1).

Integration of task and motion planning for robotics	Research & learning	Research of papers
		Learning tools usage
	P.M. documentation	Context & scope
		Planning
		Budget
		First presentation video
		Final document & presentation
		Review of technical competences
	Perception pipeline	Algorithm design & implementation
		Test & validation
	Simulation assets	Simulation 3D models
		Simulation files
	World interface	Algorithm design & implementation
		Test & validation
	Symbolic planning integration in ROS	Algorithm design & implementation
		Test & validation
	Experiments	Gathering of results
		Fixes based on feedback
	Final stage	Technical documentation
		Presentation slides
		Presentation rehearsal

Table 2.1: Project decomposition into different tasks

2.1.2. Milestones

We have identified the three tracking events that take place during the course of the project as the project's milestones. These events are not only useful for checking the progress of the project, but also to propose a set of small goals to be progressively achieved. In other words, they give us an idea of how much work should be completed at different stages of the project in order to finish in the given time. Table 2.2 summarizes the project's milestones.

Date	Milestone	Brief description	Requirements
19/Oct	First presentation	Five minutes presentation of the P.M. aspects of the project, with a turn of fifteen minutes for questions	The P.M. documentation (with the exception of the technical competences review) and slides have to be completed
29/Mar	Mid-term presentation	Review of the project progress with the supervisors	World interface is finished
25/Apr	DEP talk	Degree's End Project presentation before the examining board	All modules completed. The documentation is finished and well formatted.

Table 2.2: Project milestones

2.1.3. WBS dictionary

This section contains a glossary of all the high-level tasks presented in section 2.1.1. We give a brief description of the tasks, the major constituent(s), the predecessor(s), the start and end dates, the internal tasks and the resources it requires.

Each resource is preceded with a mark indicating its category: **(S)** stands for software resource; **(Hw)** stands for hardware resource; and **(H)** stands for human resource. As regards the human resources we have included the role(s) that the developer will perform at each task. The resources and their associated costs are analyzed in greater detail in section 3.1.1.

Task	Research & learning
Major constituent	Preparation
Description	On the one hand we have to gather relevant papers about related works. On the other hand, we have to become familiar with the working environment and tools. This includes learning about the Gazebo simulator and the ROS interface to communicate with the robot.
Predecessor	None
Planned start date	7/Sep
Planned end date	8/Oct
Deliverable(s)	None
Resources	• (Hw) LAPTOP - Laptop with the following specs: AMD A4-5000@1.5GHz CPU, 8GB RAM, Radeon HD8330 GPU • (Hw) PC-LAB - Desktop PC located at the perception and manipulation lab • (S) UBUNTU - Ubuntu 14.04LTS installed on both the PC and the laptop • (S) ROS - full Indigo installation (packet `ros-indigo-desktop-full`) • (S) GAZEBO - Gazebo simulator ver. 2.2 with ROS integration • (H) Software developer
Internal tasks	• Research of papers • Learning tools usage

Table 2.3: Task overview: research & learning

Task	PM documentation
Major constituent	Documentation
Description	We have to redact the project management documentation. This includes the context, scope, planning and budget. In addition we have to prepare the slides for first presentation.
Predecessor	None
Planned start date	14/Sep
Planned end date	25/Oct
Deliverable(s)	PM document (30 pages)
Resources	• (Hw) LAPTOP • (Hw) PC-LAB • (S) LaTeX- Full LaTeX installation (packages `texlive` and `texlive-extra`) • (S) IMPRESS - LibreOffice Impress ver. 4.2.8.2 for creating slide presentations • (S) Gantt-Project - software suite for creating Gantt charts • (H) Project manager
Internal tasks	• Context and scope • Planning • Budget • First presentation video • **Final document and review** • Review of technical competences

Table 2.4: Task overview: PM documentation

Task	Perception pipeline
Major constituent	Development
Description	This stage is devoted to developing the necessary software components for the recognition of the pieces and the sphere cavities in the images registered by the Kinect.
Predecessor	Research & learning
Planned start date	26/Oct
Planned end date	14/Dec
Deliverable(s)	Fully functional perception pipeline. This includes the image filtering and segmentation, and the recognition of the shapes and their rotation angle.
Resources	• (Hw) LAPTOP • (Hw) PC-LAB • (Hw) GAMEKIT - Sphere with cavities presented in section 1.3 • (S) ROS • (S) Gimp • (H) Software engineer • (H) Software developer • (H) Tester
Internal tasks	• Algorithm design & implementation • Test & validation

Table 2.5: Task overview: perception pipeline

Task	Simulation assets
Major constituent	Development
Description	This tasks consists of creating world files and models for simulation and benchmark in Gazebo. We want to achieve basic functionality in Gazebo as well.
Predecessor	Perception pipeline
Planned start date	15/Dec
Planned end date	4/Jan
Deliverable(s)	`.launch` and `.world` files for simulation in Gazebo. Additional models for the pieces and the sphere. Basic functionality in simulation.
Resources	• (Hw) LAPTOP • (S) ROS • (S) GAZEBO • (S) FreeCAD - software 3D CAD suite • (H) Software engineer • (H) Software developer
Internal tasks	• Simulation-related files • 3D models

Table 2.6: Task overview: simulation assets

Task	World interface
Major constituent	Development
Description	This task is devoted to creating an interface between the world and the planner. In other words, we want some means of extracting the relevant information about the world and executing the primitive tasks of the planner (e.g. pick a piece or show the relevant cavity of the sphere).
Predecessor	Simulation assets
Planned start date	5/Jan
Planned end date	4/Mar
Deliverable(s)	A ROS node that offer the relevant services for interacting with the world.
Resources	• (Hw) PC-LAB • (Hw) LAPTOP • (S) ROS • (H) Software engineer • (H) Software developer • (H) Tester
Internal tasks	• Algorithm design & implementation • Test & validation

Table 2.7: Task overview: world interface

Task	Symbolic planning integration in ROS
Major constituent	Development
Description	While there are plenty of geometric planners for ROS, there is not any standarized and maintained symbolic planner. The aim of this task is to adapt Pyhop, a HTN planner, to work in ROS.
Predecessor	World interface
Planned start date	5/Mar
Planned end date	28/Mar
Deliverable(s)	A ROS node that is capable of symbolic and geometric reasoning and is able to build plans for achieving certain objectives, like inserting all the pieces that are on the table in the sphere.
Resources	• (Hw) PC-LAB • (Hw) LAPTOP • (Hw) WAM • (S) ROS • (H) Software engineer • (H) Software developer • (H) Tester
Internal tasks	• Algorithm design & implementation • Test & validation

Table 2.8: Task overview: symbolic planning in ROS

Task	Experimentation & tests
Major constituent	Testing
Description	While there are plenty of geometric planners for ROS, there is not any standarized symbolic planner. The aim of this task is to adapt Pyhop, a HTN planner, to work inside ROS.
Predecessor	Symbolic planning integration in ROS
Planned start date	29/Mar
Planned end date	11/Apr
Deliverable(s)	A ROS node that is capable of symbolic and geometric reasoning and is able to build plans for achieving certain objectives, like inserting all the pieces that are on the table in the sphere.
Resources	• (Hw) PC-LAB • (Hw) LAPTOP • (Hw) WAM • (S) ROS • (H) Software developer • (H) Tester
Internal tasks	• Gathering of results • Fixes based on results. Ideas for future improvement.

Table 2.9: Task overview: experiments

Task	Final stage
Major constituent	Documentation, rehearsal
Description	This stage includes: gathering and formatting the notes taken during the course to redact the technical documentation (and joining this document to the PM document); preparing the slides for the talk before the selection board; and the rehearsal.
Predecessor	World interface
Planned start date	18/Mar
Planned end date	16/Apr
Deliverable(s)	• Final document (PM + Tech) • Final talk slides
Resources	• (Hw) LAPTOP • (Hw) PC-LAB • (S) LaTeX • (S) IMPRESS • (S) Gantt-project • (H) Project manager • (H) Software engineer • (H) Software developer
Internal tasks	• Technical documentation • Presentation slides • Presentation rehearsal

Table 2.10: Task overview: final stage

2.2. Action plan

Aside from the begining of the project when we alternate between redacting the PM documentation and researching, the order in which the remaining tasks are executed follows a rather sequential fashion. This is not strange since there is only one developer and the proposed tasks has a high dependency on previous ones. We will follow the plan showed in figure 2.1 as closely as possible. It is unapprochable to anticipate all kind of contingencies. However, we think that the most probable ones are those which belong to areas that do not fall directly in our main line of work (e.g. problems in the cross-related field of perception or kinematics). This was previously discussed in the sections 1.4 (*Scope*) and 1.7.2 (*Risks*). It may be also said that one of the main purposes of the six-iteration development process is to overcome these possible obstacles, pursuing small sub-goals each time.

The co-director of the project and I shall meet periodically at the perception and manipulation laboratory. Any technical difficulty and doubt can be discussed soon enough with him so the issue does not constitute a serious hindrance. On the other hand, the director and I will discuss each 15 days the progress of the project, and any related theoretical aspect. A continuous contact with the tutor shall be maintained to talk about the academic details of the project, as well.

We have estimated that the total time amount dedicated to the project is 705 hours. Since the project has a duration of approximately 23 weeks, this means that an average of 31 hours is spended each week, and 4.5 hours are spended each day on the project. We think that this is totally feasible.

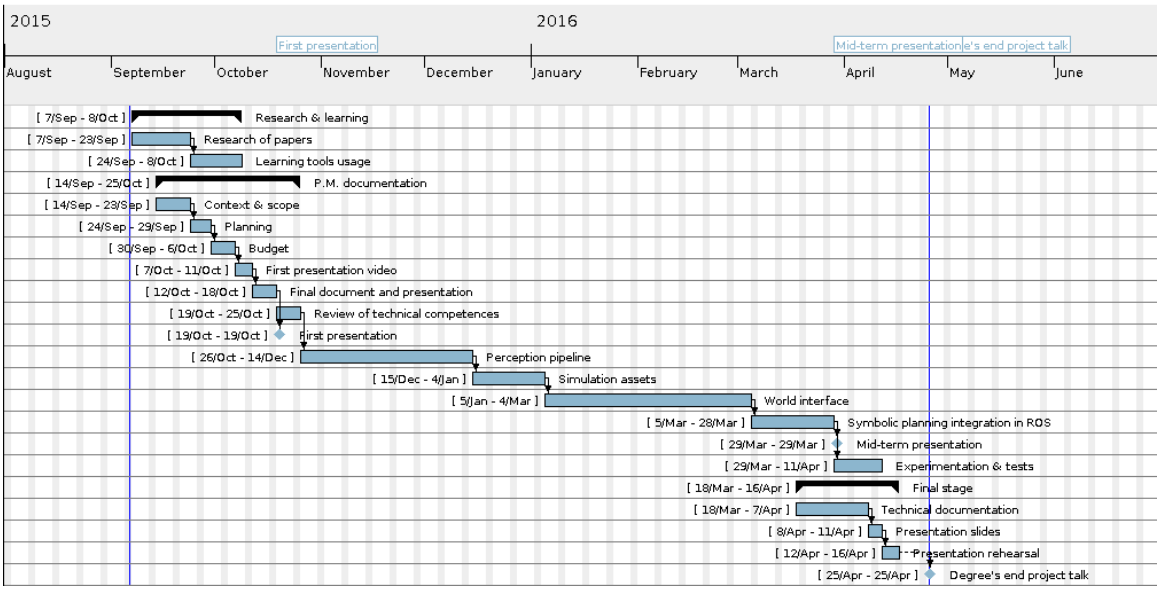

Figure 2.1: Gantt diagram. It shows both the tasks to be performed and the milestones. Since the inner structure for the development iterations from II to VI is the same, only the second one is expanded.

3 Budget and sustainability

This chapter covers the economic, social and environmental analysis of our project. More specifically, we identify the different elements that have an associated cost. These elements can be either hardware, software, human resources or general expenses. Of course, an estimation for these costs is also provided. Next we talk about the possibility of deviations from the initial estimations, and how they shall be handled. Finally, we shall discuss sustainability-related topics from an economic, social and environmental point of view.

It is also important to notice that much of the information within this document is subject to change depending on the evolution of the project. This is true specially for the list of costs.

3.1. Budget

In this section we analyze the costs associated to the software and hardware resources presented in the *Temporal planning* report. We identify the impact of the human resources in the project's budget and talk about the general expenses too. In each case we provide our most educated estimation. At the end of the section we discuss the possibility of significant deviations from the estimations and the actuation protocol in such case.

3.1.1. Cost identification and estimation

In this section, costs are analyzed according to their origin. We also show in table 3.6 the total budget and which part of this budget is linked to each of the tasks defined in the Gantt chart of the previous report.

Hardware resources

Table 3.1 shows the adquisition price and the amortization associated to each hardware element previously presented in the *Work breakdown structure* (section 2.1.1). The total adquisition price and amortization during the project is also shown. We include the gamekit in this section for the sake of completeness, although it can be seen that it may as well be ignored due to its comparatively low price.

Notice that we have set the amortization period of the robot arms longer because of its high purchase price. We expect that such an expensive product remains useful for a significantly greater interval of time.

Resource	Units	Unit price	Amortization period	Price per hour	Hours of use	Amortization
Lab PC	1	€1,000.00	4 years	€0.12	521	€62.52
Laptop	1	€400.00	4 years	€0.05	789	€39.45
WAM	2	€97,500.00	10 years	€4.84	36 (combined)	€174.24
Gamekit	1	€20.00	1 year	€0.01	36	€0.36
Total	-	€196,420.00	-			€276.21

Table 3.1: Costs associated to hardware resources

Software resources

The calculation of the software costs is rather straightforward since all the employed programs are free and open source. Table 3.1 lists the software used in this project along the major update period.

Resource	Unit price	Quantity	Major update period
Ubuntu 14.04LTS	€0.00	1	2 years
ROS Indigo (plus packages)	€0.00	1	2 years
LaTeX(`texlive`)	€0.00	1	2 years
Impress (LibreOffice)	€0.00	1	2 years
FreeCAD (3D CAD)	€0.00	1	2 years

Table 3.2: Costs associated to software resources

Human resources

In the *Temporal planning* report we saw that there is going to be just one developer working on our project. He performs tasks that are tipically conducted by individuals with different specialized roles. Because of this, the wage for our developer has been adjusted according to the amount of time he shall spend impersonating each role.

Role	Salary (per hour)	Number of hours	Total wage
Project manager	€58.96	248	€14,622.08
Software engineer	€29.27	154	€4,510.51
Software developer	€24.83	730	€18,130.67
Tester	€18.62	177	€3,308.77
Total	-	1310	€40,572.23

Table 3.3: Human resources' costs

General expenses

The general expenses are those that are not specific of the project itself but, instead, are inherent to the use of the laboratory facilities and common resources. They are also called operating expenses. In our case the most relevant ones are the electricity bill and the Internet connection.

The price of the Internet connection is not straightforward to calculate, since we are not considering a domestic connection. Instead, the IRI (*Institut de Robòtica i Informàtica Industrial*) is located at the FME (*Facultat de Matemàtiques i Estadística*) and the network resources are shared among all the students, teachers and researchers. To know the exact cost of the Internet connection it would be necessary to check the bills of the FME. Anyway, we have performed the estimation using the same rate as a domestic user.

The electricity consumption is detailed in table 3.4 and the Internet connection cost is presented in table 3.5.

The total expense is €449.36

Resource	Average power	Hours of usage	Price
Laptop	65 W	789 h	€13.78
Lab PC	250 W	178 h	€35.00
WAM	60 W	54 h	€0.58
Total			€49.36

Table 3.4: Electricity consumption and energy cost. A price of €0.2687 per kWh has been assumed.

Monthly price	Project length	Total value
€50.00	8 months	€400.0

Table 3.5: Internet connection cost

Total cost

With all the previous economic costs considered, the cost for the whole project is **€41,297.79**. To this quantity we add a margin of a 15% in order to be in position of handling additional contingencies that introduce a deviation in the cost. Check table 3.6 for a summary of all the costs and the cost decomposition among tasks.

Task	Duration (h)	P. man.	S. eng	S. dev	Tester	Laptop	PC-Lab	WAM	Cost
Research & learning	128	0	0	128	0	128	0	0	€3,184.64
PM documentation	168	168	0	0	0	126	42	0	€9,916.62
Perception pipeline	200	0	30	140	30	100	100	0	€4,929.90
Simulation assets	168	0	0	142.8	25.2	168	0	0	€4,023.35
World interface	210	0	31.5	147	31.5	105	105	18	€5,263.52
Symbolic planning integration in ROS	84	0	12.6	58.8	12.6	42	42	0	€2,070.56
Experimentation	112	0	0	33.6	78.4	0	112	18	€2,394.66
Final stage	240	80	80	80	0	120	120	0	€9,065.20
General expenses									€449.36
Subtotal									€41,297.79
Margin for contingencies (15%)									€6,194.67
Total with contingencies									€47,492.47

Table 3.6: This tables shows the number of hours each resource is active and the cost associated to each task. The last rows show the additional costs and the total cost of the whole project. Only the resources with highest impact on the economic cost have been considered.

3.1.2. Budget control

We have to contemplate the possibility of suffering mid-term unexpected events that introduce a serious deviation from the estimations given at section 3.1.1. Only alterations that involve the hardware, software and human resources are analyzed, since the general expenses are assumed to be fixed.

Changes regarding the hardware resources

Firstly, we shall consider the possibility of needing new hardware equipment. Maybe future versions of this report shall take into account elements that have not been mentioned yet (e.g. maybe a new gripper for the robots or external low budget cameras). However, we

do not think that the additional cost of this new elements will alter significantly the overall budget of the project, specially given the high cost of the human resources and the robots. As for hardware replacement, it is clear that we cannot afford to lose the WAM arm robots, given their huge value, both monetary and as a mean for the proof of concept. If such an event took place, it would prove to be disastrous for the execution of the project. Effectively, the experiment could not be conducted and the project would have to focus on its more theoretical side: designing the planner and test it with simulations. Fortunatelly, this is highly unlikely. It goes without saying that such a sophisticated and expensive system provides a high structural integrity and several safety measures. Moreover we will not use them to perform a dangerous task that could damage the robots in any way. On the other hand, the likelihood of having to substitute the laptop or the desktop computer is somewhat higher. Even so, it is quite low, and if that happens it would not require a large cost (compared to the overall budget). We could definitively afford this event, both from a monetary and a practical point of view since we will use a software repository to store periodically all the code, documents and related files.

Changes regarding the software resources

As for the software, there is not much to say since all the considered programs are free. Future versions of this report may include additional software resources that have not been mentioned until now. However, we shall stick to free software because: it suits our needs; do not involve an additional monetary cost; and carry an importal ethical aspect. This last point is further discussed in section 3.2.2.

Changes regarding the human resources

This is the part were more deviations can be expected. The amount of hours devoted to the project depends on the unexpected events thay can arise during its execution. Effectively, technical difficulties that have not been anticipated at the beginning may require more attention. Our actuation protocol to avoid high deviations in this regard is to stick to the planning presented in figure 2.1 and to adjust the expenses to those presented in table 3.6. In the same table, notice also that we have anticipated an increase of a 15% over the basic budget in order to be flexible before the emergence of contingencies.

3.2. Sustainability

Here we talk about the economic viability and the environmental and social dimensions of our project. We offer first a qualitative analysis on these topics. Table 3.7 gives a quantitative score to each one depending on the balance of the discussion.

3.2.1. Economic sustainability

As it has been said previously in section 1.7.1 (*Limitations*) this project does not entail an immediate economic benefit. It is a research project performed in an academic institution and, as such, it tackles the attainment of results that may be useful for future researchers and products in the field of robotics. This is the kind of project that is sponsored by an university, in this case the UPC; and the government. Along with other works, it will contribute to the amount of knowledge and contributions created in the UPC, thus

improving its notoriety. Hopefully, this will attract more investors who are interested in the evolution of science and technology. The project, however, constitutes only a small part of this scenario. Anyway, first is important to be successful and achieve favorable results.

The estimated budget has to be realistic and low enough for the economic capabilities of the institution that will host the project. In this sense, we think that the cost estimations of the hardware, software, human resources and general expenses are good enough. The margin for contingencies increases our estimation in a 15%, rendering a still reasonable budget that lets room for unexpected events. In balance, the overall budget, along with the contingencies margin, may as well be considered an upper bound of the real cost of the project. If our institution can afford this quantity, it could probably handle well the real cost of the project.

In conclusion, we think that the budget is fairly accurate and realistic. The main drawback is that we do not obtain economic profit in the short term, although this is the case of most research projects in academic contexts. For these reasons, we have rated the economic viability with a 6 (check table 3.7).

3.2.2. Social sustainability

We think that the nature of this project is very altruist from a social point of view because it does not go after an economic benefit. Instead its main aim is to contribute in the advance of technology. More specifically, we want our work to be useful for creating new robots that can help society and science, from assistance to old people to space exploration. Although there is a long road before robots can be seen in some applications, the results of this work shall be available for the scientific community, as much as all the articles that have constituted a source of technical support and inspiration have been available to us (see the *State of the art* section in the *Context and scope* report). However, the average citizen will be mostly unaware of our work in the short-term, since we are not developing a product that could help them somehow.

On the other, we believe that the choice of using exclusively free[1] software suits perfectly our philosophy. This way, our source code can be compiled and executed using the same tools than ours, so the results can be reproduced. In other words, our code will be free software as well. We think that this is the most ethical descision in this case.

To sum up we want our results to be available to researchers in order to contribute to science and technology. We see as a positive factor the exclusive utilization o free software. We do not think we will have any immediate impact in the life of the average individual, neither negative nor positive. This reasons have led us to give a score of 7 (see table 3.7 for all the ratings).

3.2.3. Environmental sustainability

Our project does not have a significant repercussion on the environment (at least not much more than the amount an average individual would use in a comparable time period). Surely, we need energy to run the computers and the robots, but the amount is quite low. The robots consume a bit more than 50W, and they are on only during the tests. The computers will consume much more energy and, in fact, have associated the greatest percentage of energy consumption. When the project ends the energy consumption will

[1] As the free software community likes to put it: *free as in freedom, not as in free beer*

stop. Probably, the most serious environmental issue in our project is the silicium used in the computers and robots' CPU (*Control Process Unit*) and integrated circuits. Anyway, we are not using a massive amount of these elements. As regards the software, it does not carry any environmental degradation, obviously.

If robots become part of our everyday basis in the future, it may be worth talking about the fabrication process. This, however, is a topic that falls outside of our scope and that should be discussed in specialized literature.

All in all, our project does not entail a significant impact on the environment. Because of this, we have given a rating of 9 to the resource analysis section of the sustainability matrix (table 3.7). We have not conceded the maximum mark to this part because we do not pursue to improve nor to protect the environment neither.

Sustainable?	**Economic**	**Social**	**Environmental**
Planning	Economic viability [0,10]	Life quality improvement [0,10]	Resource analysis [0,10]
Score	6	7	9

Table 3.7: Sustainability matrix. Only the rows that concern the planning phase have been included.

Part II

Technical report

4 Theoretical concepts

This chapter provides a description of the terms that will be used through the technical report. It also introduces the mathematical formalism that was relevant in the execution of the project.

4.1. Relevant terms

This section is devoted to introduce some of the most recurrent terms of this report so the reader can conceive an idea of the whole picture. We had to address them at several points of the implementation process. As opposed to section 1.2.1, these concepts are more specific. Even so, it is not our intention to provide an exhaustive background on each topic because they are not directly related with planning.

4.1.1. Rigid geometric transformations

In our experiments, we limit ourselves to rigid bodies. We use geometric transformations to specify how non-deformable objects are located in a 3D scene. Also, in the implementation of the robots' actions we often need to transform from a pose in one frame of reference (namely that of the camera) to another (the robot's frame of reference). While we will make use of an library that takes care of performing such transformations, we need a good understanding of the basics to know how, when and why it is appropriate to do so.

Geometric transformations consist of a translation vector and a certain rotation. The information can be relative to the world frame of reference or to another object. In the most general case, we need six parameters to provide complete information about the relative pose of one frame of reference with respect to another one. Three of these six parameters describe the translation between the origins of the frames of reference and the other three parameters describe their relative orientation.

There are several ways of representing the relative rotation between the frames of reference of two different objects. In our project we deal with both the Tait-Bryan or nautic angles convention (roll, pitch, yaw) and with quaternions. In short, Tait-Bryan angles are three independent rotations while quaternions are hypercomplex numbers that can be used to compress an unit vector and a rotation in the direction of this vector.

In future sections, we will designate ${}_iH^j$ as the transformation from frame i to frame j. Also we can compound transformations. For example: ${}_iH^k = {}_iH^j \bullet {}_jH^k$. Composition of geometric transforms is a key aspect for section 4.1.2.

While it is not the objective of this document to provide the full theory behind geometric transformations, we provide as an example the figure 4.2, where we can see that:

- The translation between the world frame and the `iri_wam_picker_link_footprint` (as we will see later, `iri_wam_picker` is the symbolic name we have given to one of the robots in our experiment) frame is $(0, 0, 0.72)$m. There is no change of orientation so the roll, pitch and yaw are all 0. In quaternions, such a neutral orientation transformation would be represented as $0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k} + 1$.
- The translation between the robot's footprint and the frame `iri_wam_picker_link_base` presents, again, no change of orientation. The translation vector, on the other hand, is $(0.22, 0.14, 0.346)$m.
- The translation vector between the kinect and the robot's base is $(0.901, -0.059, 0.725)$m. The roll, pitch and yaw (in degrees) are respectively 10.98, 79.24 and -172.54. The rotation can be represented as a quaternion as $-0.641\mathbf{i} + 0.032\mathbf{j} + 0.773\mathbf{k} + 0.011$. These values are provided by the robot/camera calibration, and they are important to pick and place objects whose pose is calculated in the frame of reference of the camera.

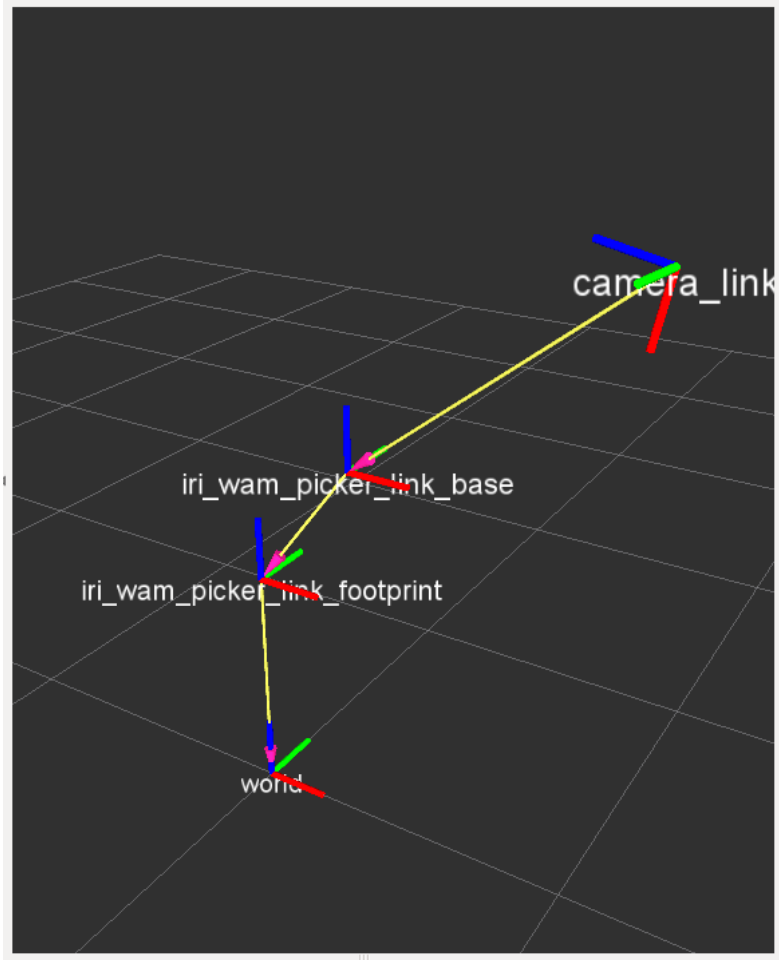

Figure 4.1: Visual representation of some frames of reference.

4.1.2. Forward kinematics

These concepts are somewhat related to those of the previous section.

The problem of **forward kinematics** is to determine the position of a robot's end-effector (e.g. a welder, a paint pistol or a gripper) with respect to the robot's base knowing the state of the joints (angle in case of rotation joints and elongation in case of prismatic joints) and knowing all the constructive parameters of the robot (e.g. the longitude of the links).

Robotic arms can be modeled as a set of interleaved rigid links and joints. There exists a convention known as the Denavit-Hanterberg parameters (D-H for short) that establishes the position of the end-effector based on successive geometric transformations between the robot links. These parameters take into account the offset between links, their length, the relative angle and the twist and can be used to define several geometric transformations between the origin of one link and the next one [1]. These transformations can be composed to obtain the transformation between the robot's base and the end-effector: ${}_{base}H^{effector} = {}_{base}H^{link_1} \bullet ... \bullet {}_{link_n}H^{effector}$. As a visual example, we provide figure 4.2 where we can see the frames of reference of a robot.

The point is: we can calculate efficiently and unequivocally the position of the robot's end-effector. In order to do so we gather the information about the state of the robot from the encoders of the motors (also known as actuators) that control the joints. Also, it is important to know what forward kinematics is in order to understand the problem inverse kinematics, that we will need to solve at several points of the experiments.

4.1.3. Inverse kinematics

The problem of inverse kinematics consists of: given a pose for the end-effector (say, a transformation ${}_{base}H^{effector}$) compute the values of the joints that effectively achieve such transformation. The problem is, therefore, much more difficult than the previous one. To start, there is the possibility that there does not exist a set of joints values that achieve such configuration. A very obvious example is a point located in a region beyond reach. Another difficulty is that there may be more than just one solution. Which one should we choose? A possible strategy is to choose the one that minimizes a certain cost function (e.g. the weighted sum of the arcs performed by each of the joints).

However, the most prominent difficulty is that an analytic solution may be hard or impractical to compute, especially for complex (many joints) robots. It is interesting to note that if we want our robot to be able to reach every point of space at each possible orientation, we need it to move in a six dimensional space even if it lives in a three-dimensional world (one dimension per axis and three additional dimensions to cover all the orientations, as we saw in section 4.1.1). For this very reason there are several approaches to compute the inverse kinematics of a pose, being many of them based on numerical methods. In practice, these methods can solve the inverse kinematics problem very efficiently, even if an analytic solution is hard to obtain. As we will see later, we have to compute the inverse kinematics of the robot at several points in order to be able of grasping and inserting the different pieces.

[1] `https://www.youtube.com/watch?v=rA9tm0gTln8` provides a visualization and a much more detailed explanation of the D-H parameters

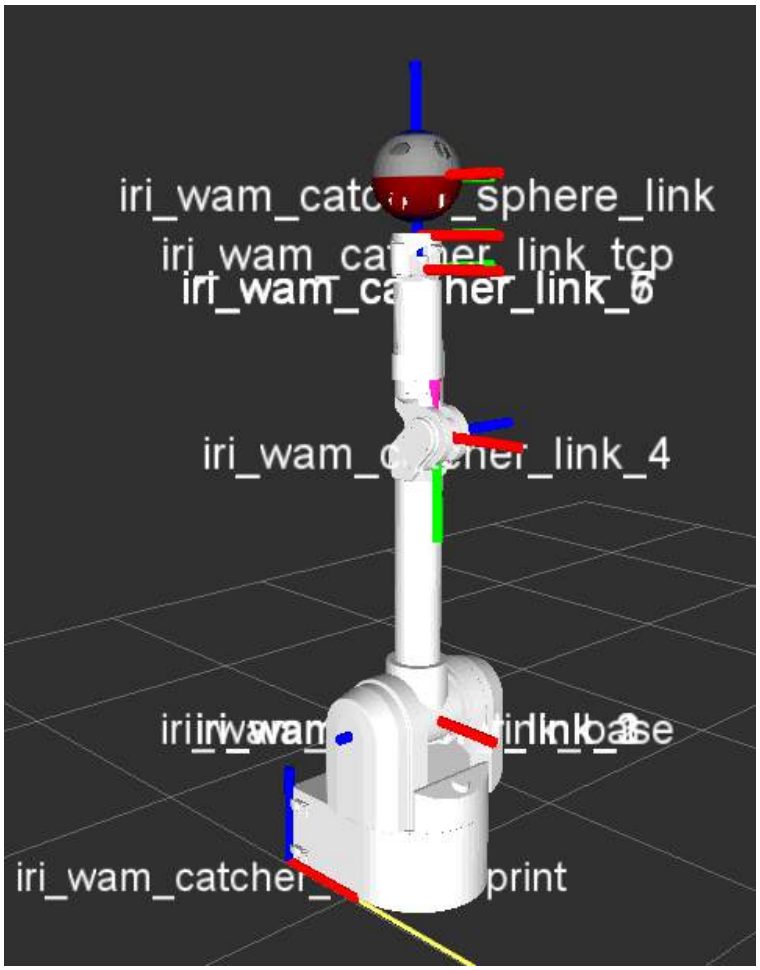

Figure 4.2: Frames of reference of one of the robots. Each link has associated a frame of reference that is defined according to a geometric transformation with respect to the previous link. The geometric transformation depends on the physical properties of the previous link (length of the previous link, offset and twist) as well as on the state of the joint that connects them. Observe how there also is a frame of reference for the end-effector (the toy sphere).

4.2. Mathematic formalism

This section covers the Mathematics involved in the planning process. We present here the relevant formulae, along with a justification of why we use them. Some of them are subject to demonstration. Demonstrations will be presented in the annex for the interested reader.

4.2.1. Estimation of magnitudes based on noisy observations

At several points we have to take information about the real world from a sensor. We use this information to set the details of the problem for the planner and therefore to decide which actions should be taken. However, sensors may introduce a significative amount of noise in the data, so even with an observation we have some uncertainty about the real value of the magnitude. In addition, it could be that the sensor does not provide the magnitude directly, but that we have to apply a processing technique to extract the data.

This processing technique potentially introduces even more noise in the measurement[2].

Because of this we talk about *estimating the magnitude* and not about *obtaining its value*. In this section we propose a model for these observations, a very simple estimator of the real magnitude and the statistical properties of such estimator. This analyisis is important in order to introduce somehow probabilistic effects in the planning process.

The model presented here is largely based on that of [Kaelbling and Lozano-Pérez, 2013] (more specifically, the section *Characterizing belief of a continuous variable*). However, we make more emphasis on how we obtain the estimator, how we update it incrementally after each observation and the statistical effects of each update.

First, let $\bar{X}$ be the value of the magnitude we want estimate, and let X_i be the value of the ith observation. We will act under the assumption that:

$$X_i \sim N(\bar{X}, \sigma_{obs}) \quad \forall i \tag{4.1}$$

In words: each observation follows a Gaussian distribution centered on the real magnitude and has a certain standard deviation σ_{obs}. We also assume that the observations are independent from each other. If we have a set of observations like this:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \tag{4.2}$$

The optimal estimator $\hat{X}_n$ (based on n observations) of the mode of the underlying Gaussian distribution is the mean of the observations:

$$\hat{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{4.3}$$

The estimator is optimal because it is **unbiased** and because it reaches the **Cramér-Rao bound** with equality (proof in the appendix A). The variance of the estimator after n observations is:

$$Var(\hat{X}_n) = \sigma_n^2 = \frac{\sigma_{obs}^2}{n} \tag{4.4}$$

Or, in other words, the standard deviation of the estimation has been reduced by a factor $\sqrt{n}$.

We propose an incremental way of updating the estimator. Let us notice that if the variance of the observations is always the same and we start with a single observation:

$$\begin{array}{c|c} \hat{X}_1 = X_1 & \sigma_1 = \sigma_{obs} \\ \vdots & \vdots \\ \hat{X}_n = \frac{n-1}{n}\hat{X}_{n-1} + \frac{1}{n}X_n & \sigma_n^2 = \left(\frac{n-1}{n}\right)^2 \sigma_{n-1}^2 + \left(\frac{1}{n}\right)^2 \sigma_{obs}^2 \end{array} \tag{4.5}$$

Figure 4.3 illustrates how the uncertainty of the mode estimation decreases with more observations.

[2] As we will see later, our sensor is a Kinect 1 camera that provides an RGB image and the 3D coordinates associated to each pixel in the image; and the processing technique is the segmentation of the image to recognize the pieces and the cavities of the sphere

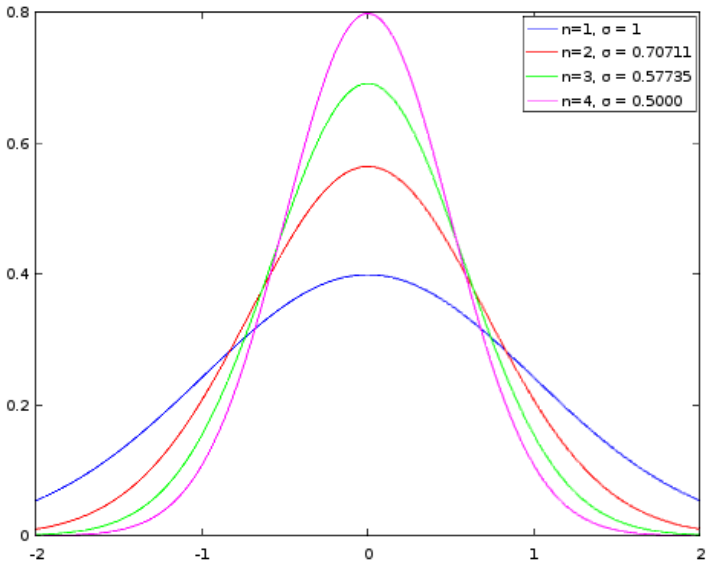

Figure 4.3: PDF of the mode estimation for one to four observations. We have chosen a Gaussian with $\mu = 0$ for the purpose of illustration.

However, what if we want to base the new estimation entirely on the current estimation, the next observation and their variances? How should we weight the next observation and the current estimator? The answer (proof in appendix A) is:

$$\hat{X}_{new} = \frac{\sigma^2_{obs}}{\sigma^2_{obs}+\sigma^2_{current}}\hat{X}_{current} + \frac{\sigma^2_{current}}{\sigma^2_{obs}+\sigma^2_{current}}X_{obs} \;\Big|\; \sigma^2_{new} = \frac{\sigma^2_{obs}\sigma^2_{current}}{\sigma^2_{obs}+\sigma^2_{current}} \tag{4.6}$$

Now, this expression is interesting because it allows us to reason about how the estimator is updated according to its current variance and the variance of the next observation. For instance, if $\sigma^2_{obs} \to \infty$ (and $\sigma^2_{current}$ is bounded), we can easily check that $\hat{X}_n \to \hat{X}_{n-1}$. This would make the new observation negligible. Alternatively, when $\sigma^2_{n-1} \to \infty$ the new estimator will be based almost entirely on the new observation. We do not have to worry about how the variance $\sigma^2_{current}$ was achieved in the first place (even if the variance of observations σ^2_{obs} has changed over time, or if we used another observation technique with different variance). Also the variance of the new estimator is always lower than that of the previous estimator, which is fairly intuitive.

We can calculate the probability of our estimation being in an interval $(\bar{X} - \delta, \bar{X} + \delta)$. This is what in [Kaelbling and Lozano-Pérez, 2013] is called "Probability near mode" (or just PNM). Such probability is:

$$\Pr(|\hat{X}_{current} - \bar{X}| < \delta) = 1 - \epsilon_{current} = \text{erf}\left(\frac{\delta}{\sqrt{2}\sigma_{current}}\right) \tag{4.7}$$

Here, $\epsilon_{current}$ is the complement of the PNM. In successive observations, this probability is updated as follows (proof in appendix A):

$$\epsilon_{new} = 1 - \text{erf}\left(\sqrt{\text{erf}^{-1}\left(1-\epsilon_{current}\right)^2 + \frac{\delta^2}{2\sigma^2_{obs}}}\right) \tag{4.8}$$

The expression 4.7 can be used to calculate how certain we are about the location of the mode. 4.8 is useful to see how the uncertainty is reduced in terms of probability instead of standard deviation. Figure 4.4 illustrates this reduction graphically. We will see that we can **base the cost of the planning actions on these probabilities**, since they help us to **decide if we should keep on observing the scene or if we can already manipulate the environment** with a certain confidence about the success of the action.

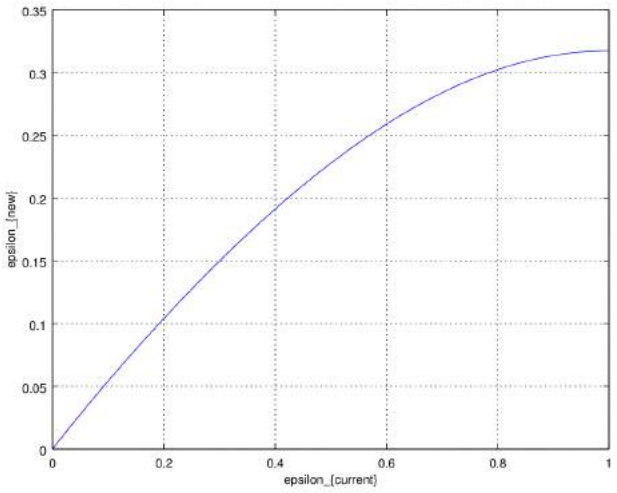

Figure 4.4: Reduction of the uncertainty about the mode in terms of probability. We have adopted $\delta = \sigma_{obs} = 0.005$ for this example. The plot shows in the horizontal axis the current probability of our estimation being farther than δ from the mode. The vertical axis shows which would be the new probability after an additional observation.

Estimation of point location in space

Let us imagine that we want to estimate the position of an object in the space. For the moment let us forget about orientation and focus just on the 3D coordinates of the centroids. We will act under the assumption that we can obtain an observation for the value of the three axis (X, Y and Z) in a certain frame, and that these observations follow a Gaussian distribution centered on the real values. All the theory presented until now about the evolution of the uncertainty with the number of observations apply to each of these variables individually.

However, we would like to find an equivalent for equations 4.7 and 4.8 that works for the three of these variables simultaneously. Say, for example, that we ask ourselves: which is the probability for my current X, Y and Z estimations to be outside a δ-sphere centered on the real position (generalization of the PNM for multivariate distributions)? How would such probability evolve after an observation?

We can group these variables together in a single multivariate Gaussian distribution like the following one:

$$\frac{1}{\sqrt{(2\pi)^3}\sigma_X\sigma_Y\sigma_Z}\exp\left(-\frac{(\hat{X}-\bar{X})^2}{2\sigma_X^2}-\frac{(\hat{Y}-\bar{Y})^2}{2\sigma_Y^2}-\frac{(\hat{Z}-\bar{Z})^2}{2\sigma_Z^2}\right) \tag{4.9}$$

Therefore, the first of the questions is answered integrating the distribution in the volume defined by the sphere of radius δ centered at the mode $(\bar{X},\bar{Y},\bar{Z})$. Considering $\sigma_X = \sigma_Y = \sigma_Z = \sigma_{coords}$ for simplicity (i.e. isotropic Gaussian distribution), the results is (proof in appendix A):

$$\begin{gathered}\Pr(\text{dist}((\bar{X},\bar{Y},\bar{Z}),(\hat{X},\hat{Y},\hat{Z}))>\delta) = \epsilon = 1 - g\left(\frac{\delta}{\sqrt{2}\sigma_{coords}}\right) \\ \text{With } g(x) = \text{erf}\,(x) - \frac{2}{\sqrt{\pi}}x\cdot\exp\left(-x^2\right)\end{gathered} \tag{4.10}$$

This new $g(x)$ is a CDF which describes a sigmoid curve. In figure 4.5 we provide the plot of such function.

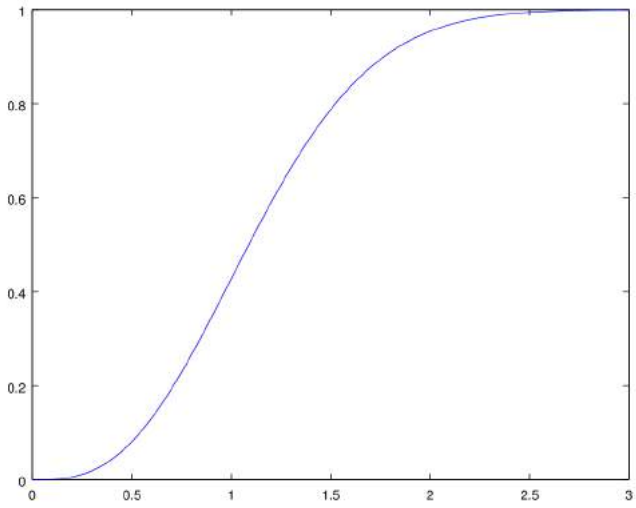

Figure 4.5: Plot of the $g(x)$ sigmoid function. It can be seen how it accomplishes the basic requirements of a cumulative density function: $g(0) = 0$ and $g(x) \to 1$ when $x \to \infty$.

The expression for the evolution of ϵ is very similar to equation 4.8. We just need to change the error function by our $g(x)$:

$$\epsilon_{new} = 1 - \mathrm{g}\left(\sqrt{\mathrm{g}^{-1}\left(1 - \epsilon_{current}\right)^2 + \frac{\delta^2}{2\sigma_{obs}^2}}\right) \tag{4.11}$$

A note about cyclic magnitudes

In our problem we also consider cyclic quantities: the piece's rotation angle. There exist more suitable distribution functions for this kind of magnitudes like the wrapped Gaussian distribution or the von Mises distribution. These, however, are not as easy to handle as the Gaussian distribution. Moreover, when the uncertainty is small, they are very similar to the Gaussian distribution. Therefore, for the sake of simplificity we consider that the observations of cyclic magnitudes follow a Gaussian distribution as well.

However, when updating the estimator it is important to do so in an appropriate manner or the result may be misleading. For example, imagine that at some point we measure an angle and we obtain a value of 5 degrees; after that we perform another measure and obtain an angle of 358 degrees. A naive arithmetic mean would yield a result of 181.5 degrees which is not very realistic. We follow the next strategy in order to update means of cyclic magnitudes:

$$\hat{\theta}_{new} = \begin{cases} \frac{\sigma_{obs}^2}{\sigma_{obs}^2+\sigma_{current}^2}\hat{\theta}_{current} + \frac{\sigma_{current}^2}{\sigma_{obs}^2+\sigma_{current}^2}\theta_{obs} & \text{if } |\hat{\theta}_{current} - \theta_{obs}| \leq \frac{\theta_{max}}{2} \\ \frac{\sigma_{obs}^2}{\sigma_{obs}^2+\sigma_{current}^2}\hat{\theta}_{current} + \frac{\sigma_{current}^2}{\sigma_{obs}^2+\sigma_{current}^2}(\theta_{obs} - \theta_{max}) & \text{if } \hat{\theta}_{obs} - \theta_{current} > \frac{\theta_{max}}{2} \\ \frac{\sigma_{obs}^2}{\sigma_{obs}^2+\sigma_{current}^2}(\hat{\theta}_{current} - \theta_{max}) + \frac{\sigma_{current}^2}{\sigma_{obs}^2+\sigma_{current}^2}\theta_{obs} & \text{if } \hat{\theta}_{current} - \theta_{obs} > \frac{\theta_{max}}{2} \end{cases} \tag{4.12}$$

Here we consider that we wrap our cyclic maginitude in an interval $[0, \theta_{max})$. For angles, $\theta_{max} = 360$. Applying this update mechanism, the estimator would acquire a value of 1.5 degrees for the previous angles (5 and 358), which has much more sense than the 181.5 degrees we obtained before.

4.2.2. Assignment of discrete probabilities with matrix scaling

In [Kaelbling and Lozano-Pérez, 2013] there is a section to discuss how to handle discrete probabilities in the planning process. However, the situation described in the article is limited to assert and modify the location of an object in a finite set of places by means of observation and move actions. They also consider a probability of false positives and false negatives.

However our problem is slightly different. We have:

- $S = \{s_1, s_2, \cdots, s_n\}$, a known set of shapes. We know that these shapes are present somewhere and we want to be able to locate and insert them on the sphere. We also know which is the colour of the piece that has a certain shape. The shapes are all different.
- $D = \{d_1, d_2, \cdots, d_n\}$, a set of blobs that we have detected by means of a computer vision technique[3] (that will be further discussed in the next chapter). Each blob has a shape associated, but due to imperfections in the perception process we cannot tell for sure which one.
- $f(d_i, s_j)$, a similitude function that tells us how similar is the blob d_i to shape s_j. f returns 0 when the blob d_i and the piece with shape d_i do not share the same colour, and it always returns a positive value when they do. We can arrange the values of $f(d_i, s_j)$ in a matrix $B = (b_{ij})_{n\times n}$ where $b_{ij} = f(d_i, s_j)$.

We want to be able to assign a probability p_{ij} to each pair (d_i, s_j). These probabilities should be based on the similitudes $f(d_i, s_j)$: the higher the similitude between a blob and a shape, the more likely is that they are the same.

Based on the previous definitions, the probabilities p_{ij} should satisfy the following:

$$\sum_{i=1}^{n} p_{ij} = 1 \quad \forall j \tag{4.13}$$

$$\sum_{j=1}^{n} p_{ij} = 1 \quad \forall i \tag{4.14}$$

The reason behind the constraint 4.13 is that every shape has one and just one blob associated, while constraint 4.14 comes from the fact that every blob has one and just one associated shape. Both combined mean that we want to establish a bijection between D and S. Now, how can we obtain the set of probabilities? Of course, one could propose several solutions that satisfy the given premises. What we suggest is to scale the matrix of similitudes B pre-multiplying it by a diagonal matrix Γ and post-multiplying it by a diagonal matrix Λ. This is:

$$(p_{ij})_{n\times n} = \Gamma \cdot B \cdot \Lambda = \mathrm{diag}(\gamma_1, \cdots, \gamma_n) \cdot B \cdot \mathrm{diag}(\lambda_1, \cdots, \lambda_n) \tag{4.15}$$

Now, the resulting system of equations is non-linear and it does not have a simple analytic solution for a general value of n. However, we can apply an iterative algorithm to compute the already scaled matrix. This algorithm consists just of alternatively normalizing the rows sum and the columns sum to 1. The details are shown in greater detail in algorithm 1.

[3] If we detected more or less blobs than n it would mean that there are false positives and false negatives in the measure respectively. For this section we assume that the number of detected blobs is correct. If this was not the case in practice, we would need to take another measurement. If the errors were too persistent, it would probably require a fine tuning of the detection algorithm

Algorithm 1 Matrix scaling

procedure ScaleMatrx(M) ▷ Input: square matrix
  $N \leftarrow M$
  $it \leftarrow 0$ ▷ it stores the number of iterations so far
  repeat
    for $i \in N.rows$ **do**
      $N.row(i) \leftarrow N.row(i) \cdot (\sum N.row(i))^{-1}$ ▷ Normalize row sum
    end for
    for $j \in N.cols$ **do**
      $N.col(j) \leftarrow N.col(j) \cdot (\sum N.col(j))^{-1}$ ▷ Normalize column sum
    end for
    $e \leftarrow \max(\max_i(|(\sum N.row(i)) - 1|), \max_j(|(\sum N.row(i)) - 1|))$ ▷ error
    $it \leftarrow it + 1$
  until $it = it_{max} \cup e < \epsilon_{max}$ ▷ Too much iterations or small enough error
end procedure

R. Sinkhorn proves in [Sinkhorn, 1964] that the problem always has solution for matrices with positive elements, and that this algorithm converges to the solution. The proof can be easily extended to block-diagonal matrices where all the blocks are composed of positive elements. The reason is that we can simply focus on each block individually. This last property is important for our particular case, because $f(d_i, s_i)$ returns 0 for pairs (d_i, s_i) that do not share colour. Even in this case, it is possible to arrange the columns and the rows of B so it forms a block diagonal matrix with the desired features. Since modifying the order of the rows and columns does not alter the feasibility of the problem, in practice this means that we can just scale B satisfactorily without rearranging.

We think it is important to highlight that this is not an exclusively ad-hoc technique for our problem. **It can be extrapolated to other contexts** (possibly also in planning) in which is necessary to establish a bijection between elements of two sets based on a certain measure of similitude or likelihood between elements. The matrix scaling technique can be used to obtain probabilities that take into account all the similitudes between pairs.

Example of matrix scaling

Let us provide an easy-to-follow example of matrix scaling. We will also compare it against row normalization.

Imagine the following matrix of similitudes:

$$\begin{bmatrix} 0.95 & 0.65 \\ 0.87 & 0.80 \end{bmatrix} \tag{4.16}$$

Here, following our notation: $f(d_1, s_1) = 0.95$, $f(d_1, s_2) = 0.65$, $f(d_2, s_1) = 0.87$ and $f(d_2, s_2) = 0.80$. Scaling the matrix with the iterative algorithm we obtain the following probabilities:

$$\begin{bmatrix} 0.54 & 0.46 \\ 0.46 & 0.54 \end{bmatrix} \tag{4.17}$$

While normalizing the rows we obtain:

$$\begin{bmatrix} 0.59 & 0.41 \\ 0.52 & 0.48 \end{bmatrix} \tag{4.18}$$

First, let us observe that the matrix 4.18 violates the condition 4.13. Next, if the decision strategy of our planner consisted of matching each blob with the most likely shape, it turns out that for both blobs row normalization has assigned the greatest probability to the same shape.

On the other hand, the matrix scaling technique is more akin to our intuition. For both blobs, the similitude to s_1 is greater than the similitude to s_2. However, since the difference between the similitudes of the first row if greater than that of the second row, it is reasonable to assume that the most likely match is $[(d_1, s_1), (d_2, s_2)]$.

Considerations about uncertainty

We retrieve the concept of entropy from the information theory field in order to characterize how unsure we are about the shape that should be assigned to a given blob. The entropy is generally viewed as a measure of the unpredictability of some process. For discrete random variables, it is defined as follows:

$$\mathrm{H}(X) = -\sum_{o \in \Omega} \Pr(X = o) \cdot \log_2 \Pr(X = o) \tag{4.19}$$

Here, X is a random variable and Ω is the set of all possible outcomes of X. When the base of the logarithm is 2, the entropy is typically measured in *bits*. We will adopt such convention in this document. Figure 4.6 shows the entropy of a random variable that may take three states. The choice of three states comes from the fact that there will be three shapes per per colour at a maximum.

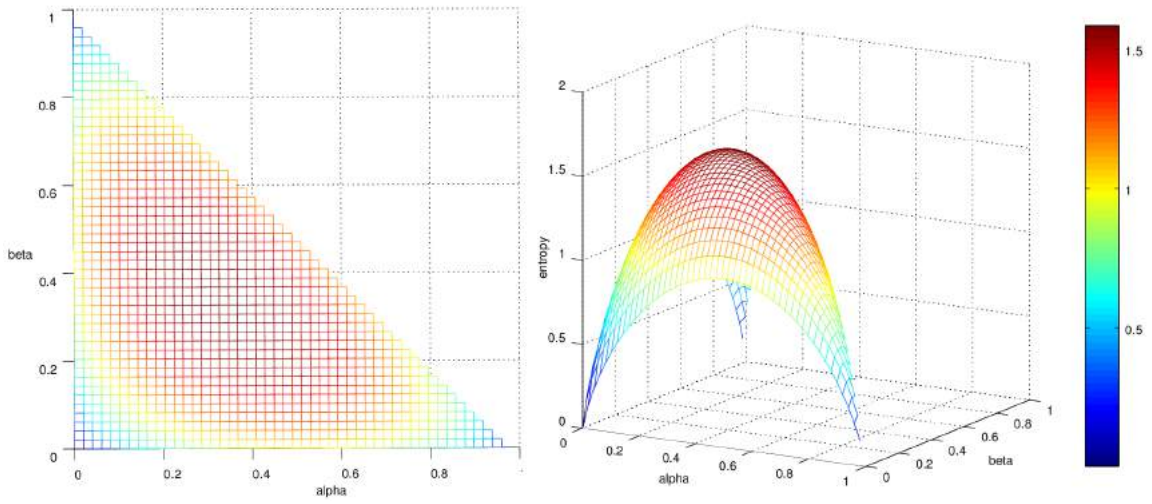

Figure 4.6: 3D visualization of the entropy of three states. Axes α and β represent the probabilities of two of these states. The probability of the third state is therefore $\gamma = 1 - \alpha - \beta$. It is interesting to note that the maximum is located at $\alpha = \gamma = \beta = \frac{1}{3}$, since it is the choice of probabilities that has associated the greatest unpredictability.

Now, we can calculate the uncertainty about which shape corresponds to a certain blob a:

$$\mathrm{H}(p_{aj}) = -\sum_{j=1}^{n} p_{aj} \cdot \log_2 p_{aj} \tag{4.20}$$

The greater the entropy, the more unsure we are about which is the correct match for a. Therefore, this measure is useful to guide the planner into taking more risky actions in low

uncertainty contexts or to be more cautious otherwise. Another definition that will come in handy is that of the normalized entropy:

$$\eta(p_{aj}) = \frac{\mathrm{H}\,(p_{aj})}{\log_2 n_a} \tag{4.21}$$

In this expression, $n_a = |\{s_j \quad / p_{aj} \neq 0\}|$ (i.e. the number of shapes such that $p_{aj} \neq 0$). Therefore $\log_2 n_a$ is the maximum entropy and $\eta(p_{aj}) \in [0, 1]$.

4.3. Planning

In this section we describe the planning paradigm followed in our project. In order to give a wide picture, we put this paradigm in contrast with approaches followed in other contexts. We also talk about how we represent the world state and other relevant elements in the domain.

4.3.1. Approaches for planning for robotic agents

One of the possibilities is to consider a classic planning approach without probabilistic effects. Following this methodology we would call the planner specifying the desired objective and setting the initial state to the the currently observable world state. After each call to the planner we execute the first of the actions (or a fixed number of actions) from the computed path and start over until we arrive to the desired objective. The clear disadvantage of this execution scheme is that it is necessary to replan at each step. If the domain is complex and plans are hard to compute this disadvantage would lead to slow execution.

A similar although more sophisticated approach is that of [Martínez et al., 2014a]. In this work the problems are modeled as MDPs so probabilistic effects come into play. The main topic of this work is to learn manipulation sequences with the purpose of assembling objects. To do so the system plans to find a sequence of actions applying the currently learned experiences. If a plan is not found, the system asks for human interaction in order to learn which should be the next step. Then the first action of the found plan or the action suggested by the human agent is executed. The system monitors the outcome of the action and updates the transition model of the MDP accordingly.

We can find a similar strategy in [Martınez et al., 2015]. It presents a novel algorithm called V-MIN that combines reinforcement learning with teachers demonstrations that roughly follows the same execution scheme.

In [Kaelbling and Lozano-Pérez, 2011] and [Kaelbling and Lozano-Pérez, 2013] the authors introduce a concept called HPN or *Hierarchical planning in the now*. In short, HPN is a technique for planning with short horizons. At first they plan with highly abstract operators. Then, each of the tasks of the abstract plan is further decomposed into increasingly specific tasks until arriving to a plan that only contains simple primitive tasks that the robots know how to execute. The advantage of such scheme is that the planning system plans for small objectives, changing the abstraction level each time. Another important fact is that in the HPN execution model it is not necessary to replan until the state falls outside the previous plan's envelope, i.e. the succession of states that results after executing all the actions sequentially.

An example of abstract operator could be grabbing certain object. The planner can assume that such operator will always succeed when planning in a high abstraction level. Then after the abstract plan in computed, the planner decomposes the grab operator into several additional tasks, like for example making sure that the piece is isolated, pushing near

obstacles if necessary in order to safely pick the desired object, moving to an appropriate configuration and finally picking the object.

It is worth noticing that this technique is somewhat reminiscent of that proposed in [Nau et al., 2015] and in [Ghallab et al., 2014]. Here the authors propose a refinement process in which each of the actions of the computed plan is decomposed into more specific operators.

We apply a similar approach. We think that this scheme is rather compatible with HTN (*Hierarchical Task Network*) planners, which we describe in the following section.

4.3.2. Hierarchical Task Network formalism

We can see HTN planning as a slightly more specialized version of classical planning. Instead of just defining a set of operators and what propositions and numeric variables define a state, we must also define ad-hoc methods in order to guide the planner in a specific domain. Therefore, there exists a compromise between generality and quick and efficient execution based on knowledge or heuristics about the domain.

An HTN planner accepts as input a domain description and a problem instance, much like a classical planner. The domain can be designed by hand, or it can be learned. In the last case the domain can be entirely learnt from the beginning or it can be learnt progressively by means of what it is known as reinforcement learning. [Martınez et al., 2015] and [Martínez et al., 2016] are recent papers that provide insight in how to do so. For the particular case of HTN, [Ilghami et al., 2002] describes *CaMeL*, a supervised algorithm for learning HTN methods.

In our case, however, the domain is simple enough to be modeled by hand. It should contain the following elements: *operators*, *tasks* and *methods*.

Operators

Operators are similar to classical actions (e.g. those in PDDL: *Planning Domain Description Language*). Each operator consists of a precondition that must be true in order to execute the action. Then it has a list with the effects that it has on the state. These effects may be: changing the state of propositions and boolean fluents in the current state, possibly using quantifiers; or binding new values to numerical variables and fluents. In general, operators can have a cost and the planner may try to find a plan that minimizes the total cost.

Tasks

An HTN planner tries to find a plan to accomplish certain task, instead of reaching a goal state. A task is some activity that must be performed. It can be either primitive or compound. A primitive task is associated with an operator and may be executed directly without further decomposition. A compound task contains one or several methods that tell the planner how to decompose the original task into several subtasks. Tasks may receive a list of arguments.

The mechanism of dividing tasks into several subtasks is somewhat reminiscent of how highly abstract operators are decomposed into several specific actions in HPN. The difference is that in HTN the abstraction level of the subtasks is not necessarily lower than that of the original task. For this reason we think that the combination of HTN and HPN is interesting and may lead to promising results.

Methods

Finally we have the methods. As it has been previously said, methods are the mechanism that allow tasks to be divided into several subtasks. A task can contain several methods, and the planner has to choose one of them in order to accomplish the task. They can be used to introduce certain degree of intelligence when taking decisions since they allow to guide the planner, telling it about correct sequences of subtasks that lead to the desired result, instead of having to (potentially) consider the complete set of operators all the time. In a totally ordered HTN planner, the subtasks that the method suggests are completely ordered. This reduces the complexity of the planning process.

In the context of HPN, methods can act as the connecting element between different abstraction levels: each task has a higher abstraction level that the tasks it decomposes into.

Introduction of non-determinism

Another interesting concept is the determinization of domains with probabilistic effects. Instead of approaching the problem directly as a MDP or a POMDP, we can model the uncertainty with operator costs and preconditions that depend on the uncertainty about the current state and the chances of success. These problems are called DSSPP, or Determinized Stochastic Shortest-Path Problem. Later, when we introduce the operators of our domain we will see how we have tackled the non-deterministic nature of our particular case.

Interleaved planning and execution

The execution algorithm followed in [Kaelbling and Lozano-Pérez, 2011] is basically the following: they start checking if the current state is the goal and, if it is, simply returns; if otherwise, it computes a plan using an A* local search that runs backward from the goal; then it iterates through each of the steps of the plan and, if a step is a primitive action it simply executes it; if it is not then it runs the algorithm recursively taking this step as a subgoal. Also, in [Kaelbling and Lozano-Pérez, 2013] the authors introduce an improvement in the algorithm: they consider that actions can fail and that they should start over building a new plan from the current state (like in [Martínez et al., 2014a] and [Martínez et al., 2014b]); and they take into account that there exists the chance that a serendipitous event occurs after executing an action that actually moves the state nearer than expected towards the objective. In such a case, the algorithm would not necessarily need to execute all the actions.

We follow a slightly different approach. We have a planning routine that takes a description of the current state and the desired goal as well. However, our routine follows the HTN paradigm and, therefore, it returns a plan that already contains purely primitive actions and executable actions. We can do this efficiently thanks to the ad-hoc knowledge that methods provide. After we have computed a plan, we execute each action, checking always if the action could be completed. If it could not, then we plan again with an updated description of the world state. We do not consider the possibility of serendipitous events that push the plan further than expected because it does not make much sense in our particular case (we think it is highly unlikely that as a result of executing a PICK action we somehow manage to introduce a piece in the toy sphere). Even so, this feature would prove to be very useful in other contexts so it should not be disregarded whatsoever. The following pieces of pseudo-code should illustrate our algorithm:

The INTERLEAVEDEXECUTION procedure is defined as:

Algorithm 2 Interleaved planning and execution. Top procedure

```
procedure InterleavedExecutionTop
    repeat
        s_now ← ObtainState()                          ▷ Current state
        T ← {InsertAll(s_now.detectedBlobs)}           ▷ Tasks to be performed
        success ← InterleavedExecution(s_now, T)
    until success
end procedure
```

Algorithm 3 Interleaved planning and execution

```
procedure InterleavedExecution(s_now, T)   ▷ Input: current state and set of tasks
    p ← HtnPlan(s_now, T)
    for a ∈ p do                           ▷ Iterate through all the actions of the plan
        success ← Execute(a)
        if not success then
            return False
        end if
    end for
    return True
end procedure
```

4.3.3. Definition of the world state

For the purpose of planning we define some sets of objects, logical assertions and fluents. We will make clear the difference between the ones that are constant from call to call to the planner, the ones that stay constant during a single planning process (although not necessarily between different calls to the planner) and the ones that can actually be altered by the operators.

Parameters that remain constant between calls to the planner

These are the parameters that will never be change even between calls to the planner:

- We will always work with two robots that we have been symbolically identified with the names Picker (the one that grasps and inserts the pieces in the sphere) and Catcher (the one that has the sphere and rotates it in order to show the correct cavity to the picker).
- We will always use the same value of δ (distances) and α (angles) when evaluating the PNM (Probability Near Mode) of a certain estimation (check 4.7 and 4.10 to see what we mean). We can think of these values as tolerances. We will always seek to reduce the uncertainty about the location of the mode under the same value of ϵ (typically 0.05).
- The standard deviation of the observations (both of the blobs' positions which we call σ_{obs}; and of their orientation, ν_{obs}) will not change.
- The parameters b_{min}, ζ and ρ that appear in the Insert, ReceiveFeedback and Pick operators. They modify the planner's behaviour making it more cautious or more hasty (check the description of the operators for more info about this).

Parameters that are constant in a single planning process

The constant variables are:

- $S = \{s_1, s_2, \cdots, s_n\}$, a set with all the shapes that we are currently considering. This may be the set with all the pieces if we are just beginning and we are calling the planner for the first time or a subset if we have already inserted some of the pieces but had to replan at some point.
- $D = \{d_1, d_2, \cdots, d_n\}$, a set with all the blobs we have detected. Again, we can give a subset of the blobs we detected at the beginning if we have correctly identified and inserted some of them.
- $B = (b_{ij})_{n \times n}$, the matrix of similitudes. The planner uses it to calculate the probabilities $(p_{ij})_{n \times n}$ at the beginning of each call. Moreover we can use the bare similitudes when there is not uncertainty about which piece is which (for example, there is a piece per colour) but we want to have some measure of how well positioned is the piece in the robot's hand once it has picked it.

Variable elements

The variable elements are:

- $P = (p_{ij})_{n \times n}$, a matrix that assigns a probability to each pair (d_i, s_j).
- OnTable(d_i), a fluent that indicates whether the ith block is on the table.
- InHand(d_i), a boolean fluent that indicates whether the ith blob is being held by the Picker. If a piece is not on the table nor in the Picker's hand the planner assumes that it is already inserted in the sphere (or otherwise beyond reach).
- RobotPose(r, o), a boolean fluent pose that indicates whether robot r is currently in pose o. $r \in \{$Picker, Catcher$\}$. o is the qualitative identifier of a certain pose. The Catcher's poses include Neutral (a pose where the robot does not disrupt the view of the camera nor obstruct the movements of the Picker) and all the shapes of the cavities indicating which shape[4] is being shown. The Picker's poses include Neutral (same concept as before), Show (showing a piece to the camera), OverTable (after picking a piece from the table) and OverSphere (after inserting a piece in the sphere).
- The current standard deviation of the blobs' positions estimations, $\sigma_{current}$ and the standard deviation of the rotation of the piece being held $\nu_{current}$. Of course, if no piece is being held, then this last variable has no meaning: we think that the rotation estimations taken when the pieces are still on the table are not very reliable, and that we have to examine them closer to the camera in order to obtain a better estimation. When we observe the table, we observe all the pieces at once, so the $\sigma_{current}$ refers to the uncertainty of the position of all the detected pieces.
- UserMatch(d_i, s_j), it indicates that the shape that corresponds to blob d_i is s_j beyond any doubt. UserMatch are either received as a feedback from the user or

[4]The list of shapes is: Triangle, Hexagon, Star8, Trapezium, Trapezoid, Pentagon, Cross, CircularSector, Star5, Star6

guessed by the ReceiveFeedback operator. See the description ReceiveFeedback in section 4.3.4 for more information about this.

The following fluents are also variable, but whether they hold or not (in the case of the boolean ones) or their quantitative value (in the case of the numeric ones) can be inferred from the previous variables. To put it in another way, they are not modified directly. Instead, they reflect changes:

- $\text{ML}(d_i, s_j)$, s_j is the most likely shape for d_i.
- $\text{BV}_{xyz}(d_i, \delta, \epsilon)$, the estimation of the (x, y, z) position of blob d_i is inside a δ-sphere centered on the real value with probability at least $(1 - \epsilon)$.
- $\text{BV}_{\theta}(d_i, \alpha, \epsilon)$, the estimation of the θ rotation of blob d_i is inside an α-interval centered on the real value with probability at least $(1 - \epsilon)$
- We will often use $H(d_i)$ which is, perhaps, an abuse of notation. By it we mean the uncertainty about which is the correct shape for blob d_i and it is calculated as an entropy: $H(d_i) = -\sum_{j=0}^{n} p_{d_i j} \cdot \log_2 p_{d_i j}$. Similarly, we will denote the normalized entropy as $\eta(d_i)$.
- Holding(), gives the blob currently held by the Picker, or $\varnothing$ (None) if there is not any.

4.3.4. Operators

The next step is to define a set of operators or, equivalently, primitive tasks. We show the signature of each operator along with a brief description. The signature of an operator includes:

- **The precondition**: a set of assertions about the world state that must be true so the action can be executed
- **The postcondition or effect**: how the world state will be modified after the operator's execution.
- **The cost**: It tries to measure how hard is to execute the action. It may be based on the uncertainty about the current state and/or the chance of success of the operator. The planner may try to find a plan that minimizes the accumulated cost.

> AssumePose(r, o)
> **Pre: not** RobotPose(r, o)
> **Effect:** RobotPose(r, o)
> **Cost:** 1

This action simply tells one of the robot to change its qualitative position. For example, after picking a piece from the table the robot's pose would be OverTable. Then we can make that the robot shows the piece to the camera from a closer perspective applying operator AssumePose(Picker,Show). There are not probabilistic effects involved in this operator. The choice of the cost is mostly arbitrary. However, in a more advanced version of the operator we could make the cost dependent on the current pose of the robot and the desired one.

ObserveTable()
Pre: RobotPose(r,Neutral) $\forall r$
Effect: $\sigma^2_{current} \leftarrow \frac{\sigma^2_{current}\sigma^2_{obs}}{\sigma^2_{current}+\sigma^2_{obs}}$
Cost: $1 - \log(\epsilon_{xyz})$

The main purpose of the ObserveTable action is to reduce the amount of uncertainty about the location of the pieces that are on the table. The cost is dependent on $\epsilon_{xyz} = 1 - g(\frac{\delta}{\sqrt{2}\sigma_{current}})$. The idea behind this is to penalize the use of the operator in low uncertainty conditions, and to encourage it otherwise. In figure 4.7 we can see the dependence of the cost on the standard deviation $\sigma_{current}$.

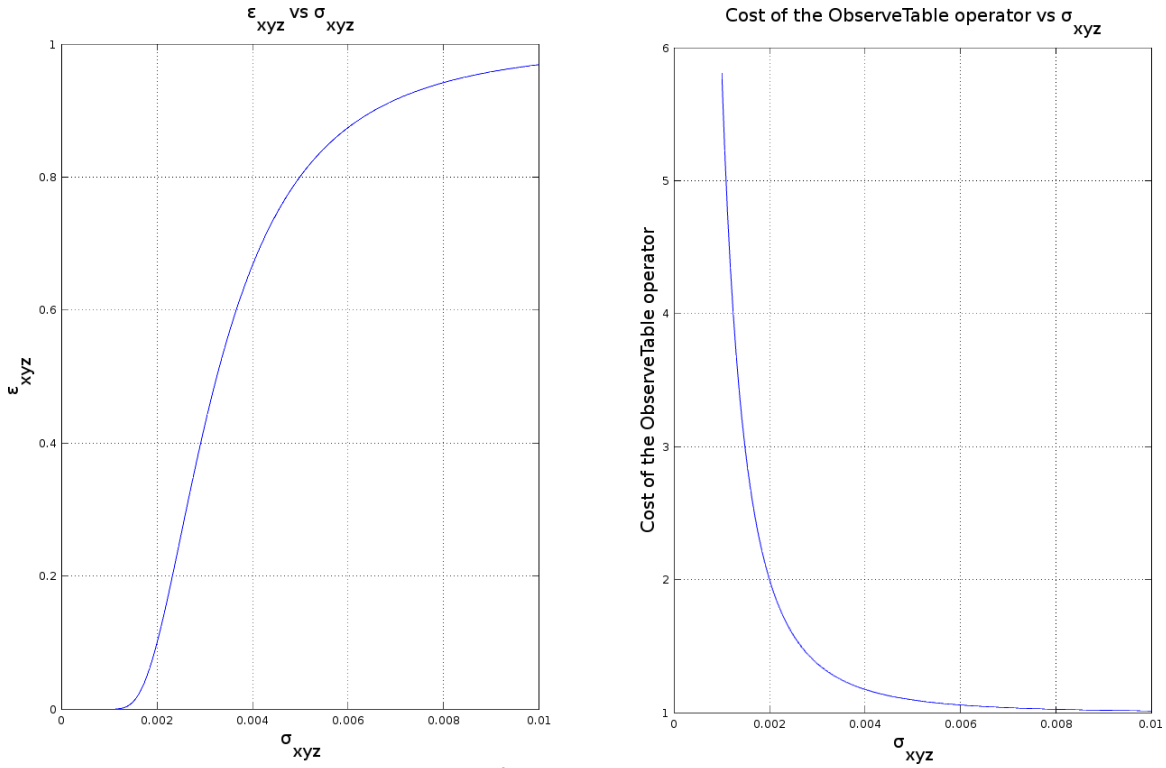

Figure 4.7: At the left, $\epsilon_{xyz} = 1 - g(\frac{\delta}{\sqrt{2}\sigma_{current}})$. At the right, cost of the ObserveTable operator calculated as $1 - \log \epsilon_{xyz}$. In both cases we have taken $\delta = 0.005$ (i.e. 5mm).

Pick(d_i)
Pre: (RobotPose(r,Neutral) $\forall r$) $\cap$ Bv$_{xyz}(d_i, \delta, \epsilon)$ $\cap$ OnTable(d_i)
Effect: not OnTable(d_i) $\cap$ InHand(d_i) $\cap$ RobotPose(Picker,OverTable)
Cost: $1 - \rho \log(1 - \epsilon_{xyz})$

The aim of this operator is to grab one of the detected blobs. Of course, in order for this operator to be applicable, the blob must be on the table. We also require that both robots are in Neutral position so there is no risk of collision. Bv$_{xyz}(d_i, \delta, \epsilon)$ means that we need a confidence of at least $(1 - \epsilon)$ in that the estimator of the blob's position is in a δ-sphere centered on the actual value. The cost penalizes the execution of this operator when there is high uncertainty about the real location of the blob.

ShowBlob(d_i)
Pre: RobotPose(Picker,Show) $\cap$
RobotPose(Catcher,Neutral) $\cap$ Holding() $= d_i$
Effect: $\nu^2_{current} \leftarrow \frac{\nu^2_{current}\nu^2_{obs}}{\nu^2_{current}+\nu^2_{obs}}$
Cost: $1 - \log \epsilon_\theta$

ShowBlob is very similar to ObserveTable. Its main purpose is to improve the estimation of the rotation. We have found that, in order to correctly tell the orientation of a piece, we have to perform a closer look or otherwise the estimation may be very misleading. The cost of the operator depends on $\epsilon_\theta = 1 - \text{erf}\left(\frac{\alpha}{\sqrt{2}\nu_{current}}\right)$, which is the probability of our estimation being outside an α-interval centered on the real value. Again, we seek to penalize the use of this operator when we the estimation is solid.

Insert(d_i, s_j)
Pre: RobotPose(Picker,Neutral) $\cap$ RobotPose(Catcher,s_j) $\cap$ Ml(d_i, s_j)
$\cap$ ($b_{ij} > b_{min} \cup$ UserMatch(d_i, s_j)) $\cap$ Holding() $= d_i \cap$ BV$_\theta(d_i, \alpha, \epsilon)$
Effect: not InHand(d_i) $\cap$ RobotPose(Picker,OverSphere)
$\cap$ $(p_{ij})_{n\times n} \leftarrow$ UpdateProbabilities($(p_{ij})_{n\times n}, d_i, s_j$)
Cost: $1 + \eta(d_i)$

Insert takes care of inserting a blob d_i in the cavity s_j of the sphere. Of course, in order to execute the operator we first need that the Picker is holding d_i and that the Catcher is showing the cavity corresponding to shape s_j. We require that s_i is the most likely shape for d_i and that the similitude of d_i to s_i is greater than a certain threshold b_{min} **or** that the user guarantees that the correct match for d_i is s_i. After the execution of the action the blob is no longer in the robot's hand. The UpdateProbabilities updates the probabilities matrix according to the new match (d_i, s_j). Algorithm 4 shows how this update is performed. The last thing that remains for discussion about this operator is the cost: as we can see it depends on the normalized entropy $\eta(d_i)$. Our intention is to make this operation more expensive under low-uncertainty conditions. The planner can decide whether to insert the sphere on the sphere directly or whether to ask for feedback following the criterion of selecting the cheapest sequence of actions.

Algorithm 4 Update probabilities procedure

procedure UpdateProbabilities($P = (p_{ij})_{n\times n}, d_i, s_j$)
  $Q \leftarrow P$
  $Q.row(i) \leftarrow \mathbf{0}_{1\times n}$
  $Q.column(i) \leftarrow \mathbf{0}_{n\times 1}$
  $Q(i,j) \leftarrow 1$
  return ScaleMatrix(Q)
end procedure

ReceiveFeedback(d_i, s_j)
Choose: $s_j \in \{s_{j'} \quad / p_{ij'} > 0\}$
Pre: RobotPose(Picker,Show) $\cap$ InHand(d_i)
Effect: $(p_{ij})_{n\times n} \leftarrow$ UpdateProbabilities($(p_{ij})_{n\times n}, d_i, s_j$) $\cap$ UserMatch(d_i, s_j).
Cost: $1 + \zeta - \eta(d_i)$

We have included an operator for receiving feedback from an human agent whenever the planner cannot determine with enough confidence which is the piece that the robot

is holding. The operator can also be useful when the piece is being grabbed precariously. It is interesting to notice that it is somewhat reminiscent of the teacher's intevention in [Martínez et al., 2014a].

ReceiveFeedback takes as input a blob and a shape, and adds an UserMatch(d_i, s_j) fluent to the current state that asserts, without any doubt, that d_i's shape is s_j. The shape is given by the user. Since the planner cannot know the answer of the user beforehand it should take a guess before applying the operator, hence the **choose** tag in the signature. When trying to execute the action in the real world, if the feedback from the user differs from what the planner has guessed, then the action fails and a new plan is computed (as shown by the algorithm 2).

The cost depends on a certain constant ζ and on the normalized entropy $\eta(d_i)$. ζ is a parameter that can has to be configured beforehand. The higher it is, the more reluctant will be the planner of taking this action instead of inserting the piece on the sphere directly without asking for feedback. On the other hand, the higher the entropy, the more prone is the planner to use this operator.

At this point it is interesting to perform further analysis on how will the planner decide what should it do when the options are to insert without human interaction and act upon feedback. Here we show the difference between the sequences of actions corresponding to each decision:

- **Insert without feedback:** $\cdots \rightarrow$ ShowBlob(d_i) $\rightarrow \cdots \rightarrow$ ShowBlob(d_i) $\rightarrow$ AssumePose(Picker,Neutral) $\rightarrow$ AssumePose(Catcher,s_j) $\rightarrow$ Insert(d_i, s_j) $\rightarrow \cdots$. The cost of the Insert operator is $1 + \eta(d_i)$.
- **Insert with feedback:** $\cdots \rightarrow$ ShowBlob(d_i) $\rightarrow$ ReceiveFeedback(d_i, s_j) $\rightarrow \cdots \rightarrow$ ShowBlob(d_i) $\rightarrow$ AssumePose(Picker,Neutral) $\rightarrow$ AssumePose(Catcher,s_j) $\rightarrow$ Insert(d_i, s_j) $\rightarrow \cdots$. The cost of ReceiveFeedback is $1 + \zeta - \eta(d_i)$ and Insert has a cost of just 1 because ReceiveFeedback has reduced the entropy to 0.

The number of ShowBlob actions is the same in both sequences since we need to reduce the rotation's uncertainty to the same value in both cases. Therefore, in order for the first decision to be preferred over the second one:

$$(1 + \eta(d_i)) < (1) + (1 + \zeta - \eta(d_i)) \Rightarrow \eta(d_i) < \frac{1 + \zeta}{2} \tag{4.22}$$

And with this we see more clearly the relevance of the parameter ζ. More specifically, it determines a decision threshold for the entropy of the blob. If the entropy is lower than such threshold, the piece is inserted directly. Otherwise, the planner will choose to prompt for human interaction. In addition this expression shows that the reasonable range of values for ζ is $[0, 1]$: if we made ζ lower than 0 the cost of the ReceiveFeedback action could potentially be 0 and making it greater than 1 has no point since $\eta(d_i) \leq 1$. Since with algorithm 2 we do not re-plan until the execution of an action in the real world fails, we can force that Insert and ReceiveFeedback fail whenever $\eta(d_i) \geq \frac{1+\zeta}{2}$ or $\eta(d_i) < \frac{1+\zeta}{2}$, respectively.

5 Implementation

This chapter describes the design decisions and the most relevant implementation details of the project. We also include a review of the simulation techniques and tools that we have used before accessing to the real robot.

The first part of the chapter is an overview of the whole implementation. This includes the list of all the implemented modules and a the relationships between them. Then we dedicate a section to the simulation assets. The rest of the chapter is devoted to present in more detail the functional modules of the project.

5.1. Overview

In this section we present to the reader infrastructure of our project. This infrastructure consists mainly of ROS, Pyhop and the procedures suggested by the Labrobotica philosophy. Then we will give a glance to the major components and constituent blocks that we have implemented.

5.1.1. Introduction to ROS

ROS is the acronym of *Robot Operating System*. This name, however, is misleading since ROS is not an Operating System in the traditional sense of the concept. Instead, ROS is an Open Source platform aimed at programming algorithms and drivers for application in Robotics. The desired functionality is wrapped in what is called a *ROS package* and then it can be shared with the community. The main objective of ROS is to facilitate the exchange and utilization of state-of-the-art algorithms and tools. There exist a high amount of useful packages being shipped for ROS. We will discuss some of these packages later.

Next, we discuss some of the features that make ROS a valuable resource in the field of Robotics.

- **Execution model:** In ROS, the minimum execution unit is the *node*. A node is a concurrent/parallel application or utility that performs certain task or functionality. In order to execute a node, there must be one ROS *master* running. A master handles several nodes and dispatches messages (more about messages below) among them. The master can be running on a remote machine. This execution model provides a natural mechanism to separate the functionality of a complex application into several modules.

- **Topics:** Nodes can communicate between them by means of *messages*. Messages can be published by one or more nodes in what is called a *topic*. Other nodes may subscribe to those topics and receive the messages being published. This allows great flexibility since the subscribers do not have to worry about the implementation details of the publishers as long as they follow the same messaging convention. Applications do not have to be a monolythic pieces of code. Instead, this messaging mechanism allows nodes to be like small functional blocks whose inputs and outputs can be plugged in to other blocks or operate in isolation. Moreover, if the master is running in a remote machine, the messages are sent through the network trasparently so the programmer does not have to worry about sockets. One of the most typical uses of topics is to stream a continuous flow of data (e.g. video from a camera or the joints state of a robot).

- **Services:** Services have certain degree of similitude with regular functions in most programming languages. Nodes can offer services that may be called by other nodes. The specification of a service includes the input arguments and the output received by node that has called the service. Calls to services are blocking, so these are typically used for fast computations (e.g. an inverse kinematics query).

- **Actions:** Actions resemble services in that they offer some kind of callable functionality for being used by other nodes. The difference is that actions are normally used to implement non-blocking functions that may span during a long interval in time. The calling node may preempt the action at any moment, and the node that offered the action can publish a periodic feedback telling the caller about the current state of the action. Because of these characteristics one common use of actions is to implement robot movements.

- **Support for several programming languages:** ROS provides support and an API for several programming languages, including C++, Python, Lisp and Java for Android. Even so, the documentation of ROS and most of its packages is focused mostly on the C++ API and the Python API. The languages used in our project are C++ and Python.

- **Debugging and configuration tools:** ROS comes or can be extended with several debugging and configuration tools that may run from the command line or have graphical interface. One of these tools is `rqt-graph`, a package with contains a Qt application for visualizing the nodes, and publisher/subscriber relationship between them. We also have `rqt-reconfigure` which allows us to dynamically change the parameters of nodes that are already running.

Next we present some example packages in order to give an idea of what kind of functionality can be offered by a ROS package (this is not intended to be an exhaustive list of all the packages we are going to use):

- **tf:** `tf` takes care of publishing the transformations between different frames of reference (for example, between the joints of a robotic arm) at each instant of time. It can also be useful for obtaining the coordinate transformation between two frames that are connected indirectly, or for tracking the transformation between two frames in the past. In addition, the API of `tf` comes with some additional utilities (e.g. performing conversions between angles in RPY and quaternions). Check 4.1.1 to see more about geometrical transformations.

- **rviz:** `rviz` is a multipurpose visualization and interactuation tool. It allows us to view the state of a robot, information from sensors that is published in a topic, etc. In this sense it is a very powerful monitoring tool.
- **moveit:** powerful geometrical planning framework for robots. `moveit` uses OMPL (another ROS package for path planning) for planning trajectories for robots. It also comes with a `rviz` plugin for visualizing plans and calculating new ones. `moveit` works for several robots. It takes an XML description of the robot's joints and links and use them to calculate a plan without self-collision nor collisions against world objects.

The IRI has its own set of ROS packages, developed inside the institute. We make use of some of them. For example (`iri_wam_description`) contains the specification of the robots in terms of joints and links. This package is useful for simulation and for calculating the transformation matrix between the frames of reference of the WAMs' bases and the end-effector. Another useful package is `iri_wam_ik`, which contains a node that offers a service for calculating the inverse kinematics for a given pose (i.e. obtaining the robot joints values for a certain pose in cartesians). To learn more about this topic please refer to section 4.1.3.

5.1.2. LabRobòtica philosophy

LabRobòtica is the group of developers at the IRI. Diagram 5.1 shows the typical development workflow followed at the institute.

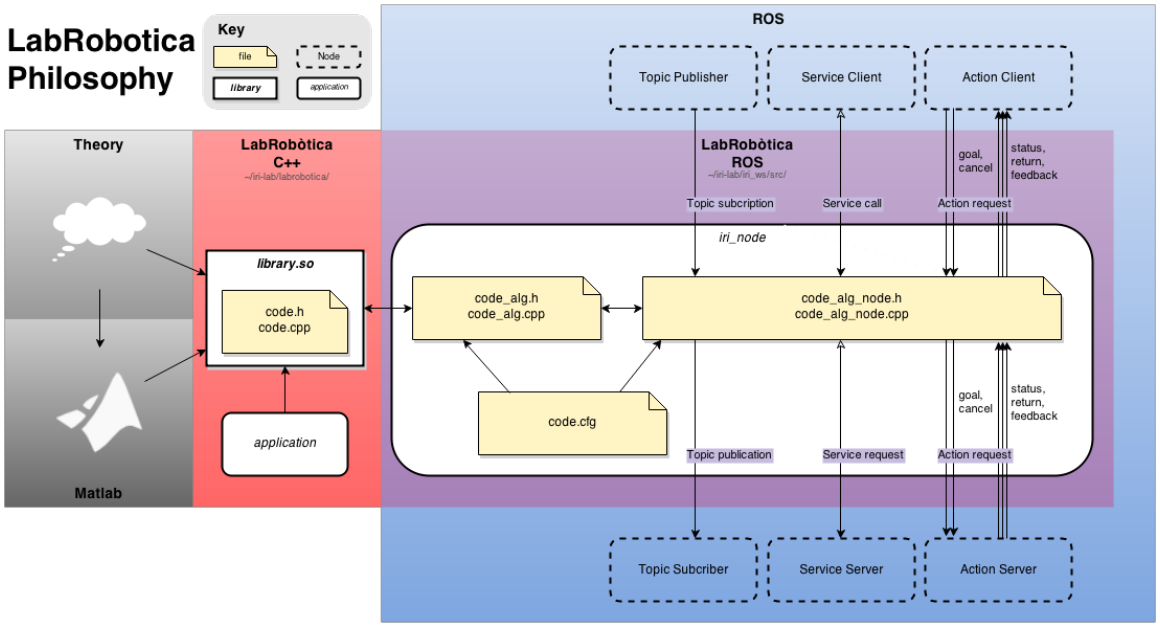

Figure 5.1: This diagram shows the developing workflow suggested at the IRI. Source: `http://wiki.iri.upc.edu/index.php/LabRobotica_Philosophy`

In words, what this diagram shows how to transform an initial idea for a driver or an algorithm from theory to practice. The process start with the proposal of an idea (e.g. a controller for certain robot or an algorithm for inverse kinematics) and the theoretical concepts behind it. If possible, the idea should be illustrated with a Matlab/Octave script in order to obtain preliminary results. After that, the core functionality behind the idea is implemented in a C++ library. This library is typically built with `cmake` and `make`. This library is, in some way, the lowest level component of the new application or driver. Some

projects stop at this point, with a functional library that can be included in other projects. However, the full process contemplates wrapping this library in a ROS node. This node makes use of the API (*Application program interface*) offered by the library and gives useful functionality based on it by mean of the standard ROS mechanism, presented in section 5.1.1.

In our case, thes full process is followed for the perception module. We have implemented as a library a set of extra features based on *OpenCV*. This library (that we have called `cv_extra`) is then used by one or more ROS nodes that make use of the API to segmentate an image and match the obtained contours with the shapes of the pieces.

LabRobòtica maintains a repository with several libraries. In our project we make use of some of them. The most important ones in our case are the controller for the WAM robot and the algorithm for inverse kinematics, since our world interface module makes use of it. For each of these there is a ROS node that acts as a wrapper. For more information on this topic, check `http://wiki.iri.upc.edu/index.php/LabRobotica`.

5.1.3. Planning engine: Pyhop

We can obtain the behaviour described in [Kaelbling and Lozano-Pérez, 2013] with conventional planners like *Fast Forward*, *Fast Downward*, *Gourmand* or *Prost*. We have decided to use a HTN planner for the reasons listed in section 4.3.1. A well-know family of HTN planners is SHOP (*Simple Hierarchical Ordered Planner*), by D. Nau:

- **SHOP**, in Lisp. More details in [Nau et al., 1999].
- **SHOP2**, in Lisp. Explained in [Nau et al., 2003].
- **JSHOP**, programmed in Java. Uses a planner compilation technique to create domain-specific planners from the domain description. This technique is explained at [Ilghami and Nau, 2003]
- **Pyhop**, a very simple planner written in Python. It can be obtained from `https://bitbucket.org/dananau/pyhop`

For our project, we have chosen Pyhop because: it allows inmediate integration with other nodes in ROS; it does not make use of a parsing mechanism since the domains and the problems are specified directly in Python; additional functionality and capabilities can be easily integrated in this planner (i.e. the code of the planner is easily modifiable and we can experiment with features that are not typically implemented in conventional planners). Another characteristic is that in Pyhop the states are represented by means of conventional Python classes and we can choose any representation method we like: simple variable bindings, dictionaries, lists, matrices, etc. However, we want to make clear that Pyhop is very simple and far from the sophistication of the latests planners like PROST, Gourmand, Fast Downward, or even from the other planners of its family (JSHOP and SHOP2).

Also it is important to mention that the original version of Pyhop does not support metric minimization. Since we would like to minimize the accumulated cost of the operators, we have modified Pyhop to support this feature using a branch & bound with a limit in the recursion depth. Anyway, this modification is transparent for the users in the sense that they do not have to use the feature nor worry about modifying their previous domains. The following example works both in both the modified version and in the original one.

Introduction to Pyhop

This subsection is devoted to explore some of the particularities of Pyhop. To do so we have ellaborated an example. Let us imagine the following problem: sorting a list with only one operator for swapping two elements. We will show the specification of the domain and an example problem step by step. This example should be easy-to-follow even if the reader has not prior knowledge of Pyhop of does not have experience with the Python programming language.

```
from pyhop import *

# Operators' definition

def swap(state,x,y):
  """ Exchanges positions of x and y. Precond: x and y must be different
  elements. """
  if x != y:
    tmp = state.position[x]
    state.position[x] = state.position[y]
    state.position[y] = tmp
    return state
  else: return False # Operators must return False if they do not succeed.

declare_operators(swap) # This call declares a primitive task for each of
                        # the specified operators
```

Code snippet 5.1: Declaration of operators in Pyhop

This code snippet is pretty straighforward. In the first line we import all the methods and classes from Pyhop. Then we define a single operator that interchanges the positions of two elements. Operators in Pyhop always receive a state as the first argument, followed by the remaining parameters. Proposition and fluents are represented as variable bindings in the state object. In this case, the state only contains a dictionary that stores, for each element, its position. When the operator can be executed successfully, it returns the new modified state. If this is not the case because the requirements (precondition) are not met, it just returns `False`.

```
# Methods' definition

def sort_list(state,goal):
  """ Method for top-level task. It receives as an argument a goal state.
  It decomposes the task into several smaller subtasks that consist of putting
  each element in its final position. """
  return [('put_in',element,goal.position[element])
      for element in state.position.keys()]

# The following call defines a task sort_list with a single method.
declare_methods('sort_list',sort_list)

def change_position(state,x,endpos):
  """ Puts element x in position endpos. """
  # Find element that is on position endpos
  for y,position in state.position.items():
    if position == endpos:
      return [('swap',x,y)]
  return False # If we reach this point the method cannot succeed.

def let_alone(state,x,endpos):
  """ Let element x alone because it is in its desired position. """
  if state.position[x] == endpos:
```

```
    return [] # No more task decomposition (success without further action)
  else: return False # The element is not in its final position

# Task put_in with two methods
declare_methods('put_in',let_alone,change_position)
```

Code snippet 5.2: Declaration of methods in Pyhop

Contrarily to operators, methods do not return a state. Instead they return a list of subtasks (including the arguments that each subtask receives). Previously we have said that HTN planners make plan for accomplishing tasks instead of reaching a goal state. If we want to reach a particular world state, we can define a task that receives as an argument such goal. This is what we do in the `sort_list` method that belongs to the homonymous task. This completes the specification of the domain. In Pyhop the states do not have to be rigourously defined since they consists of a simple object with arbitrary attribute bindings. However, we can choose to write a stricter state definition by means of a class that inherits from Pyhop's `State` (or make a completely new class from scratch since Python supports the *duck typing* style). We have not done so for this simple example.

```
# Define the start state.
start = State('start')
start.position = {'a':2,'b':3,'c':1,'d':4}

# ... and the goal state
goal = State('goal')
goal.position = {'a':1,'b':2,'c':3,'d':4}

# Define tasks to be accomplished
tasks = [('sort_list',goal)]
```

Code snippet 5.3: Definition of a problem instance

This shows how we define a problem in Pyhop. As it can be seen, we create a starting state and a set of tasks (in this case, only one) that have to be accomplished.

```
pyhop(start,tasks,verbose=3)
```

Code snippet 5.4: Running Pyhop

This last snippet shows how we run Pyhop for this particular problem instance. This yields the following actions sequence: `swap(a,c)`, `swap(c,b)`. It can be checked that this sequence produces the correct result.

In order to use the new metric minimization feature, one would need to add a keyword argument (`minimize_metric=<attribute name>`) to the call to the planner. For example:

```
# With this call the planner will try to find a plan that minimizes a metric.
# Of course, for this to work we would need that the state contains a certain
# attribute called cost.
pyhop(start2,tasks,verbose=2,minimize_metric='cost')
```

Code snippet 5.5: Running Pyhop with metric minimization

5.1.4. Implemented modules

Figure 5.2 shows a simplified overview of the modules and nodes we have implemented for our application. Further sections will ellaborate more on each module. For the moment we offer a brief description about each one:

- **The perception pipeline** takes care of all the perception related problems. We have identified three main challenges: detecting and identifying the pieces that are lying on the table; detecting the cavities and providing an estimation of the rotation angle; and detecting the piece that is being shown by the PICKER to the camera and estimating its rotation angle. For the first two challenges we have to provide some mechanism for detecting the blobs in the images provided by the Kinect, calculating the similitudes between them and the known shapes and obtaining the 3D coordinates of the centroid. For the last challenge we have to detect and obtain the similitudes as well. We do not calculate the 3D coordinates though, the reasons being: in order to perform a closer look with satisfactory results we need the piece to be very near to the camera, so near that the depth sensor is not able to compute the coordinates; we already have the piece in the end-effector, so we currently know where it is located (even so it would be useful to compute its 3D coordinates as an additional testimony of whether or not the piece is being grasped correctly). Section 5.3 provides more details about this module.

- **The world interface** is devoted to all the actions that involve moving the robots. We have established a separation between the perception mechanisms (described before) and the actuation mechanisms. The name may be somewhat misleading since taking information from the world is also a way of interfacing with it. Anyway, semantics aside, the purpose of this module is to give instructions to both the CATCHER and the PICKER. The CATCHER actions consist of adopting a NEUTRAL position and exposing a certain cavity both for being visible for the camera and so the PICKER can insert the relevant piece into it. The PICKER actions include adopting the NEUTRAL and SHOW position and picking or placing a piece. Section 5.4 ellaborates more on this.

- **The interleaved planning-execution** part is written in Python and it consists of the implementation of the algorithm 2, at section 4.3.2. It takes care of interfacing with all the services and actions offered by the previously described module, and of handling all the ROS-related mechanisms and procedures. It also have access to our modified Pyhop version, that have called ROSHOP.

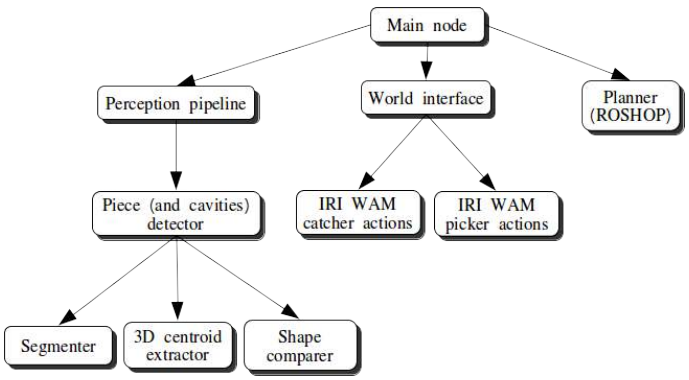

Figure 5.2: Block diagram of the whole application. The edges indicate the dependence relationship between them. The resulting graphs is a tree in which the nodes with greater height offers services and actions to their parents.

5.1.5. Source code

The ROS related source code of the project is maintained in a public Git repository[1]. The reader can access it and clone the repository if he desires. Please be aware that these packages have dependences on several of the IRI's LabRobòtica projects. The instructions to install these packages can be found at the IRI's wiki[2].

For the perception algorithms we have implemented an additional C++ library called `cv_extra`. This library is maintained in a different repository, [3], and it should be installed as well.

It is also worth mentioning that the code is fully documented. We have used Doxygen-like comments and, in case of the `cv_extra` library, the documentation is already compiled in HTML.

5.2. Simulation assets

We think that the work behind the simulation of the scene is relevant enough to deserve some attention. In our case we have found simulation to be useful for testing the perception and actuation parts in contexts when we do not have access to the laboratory equipment and also when testing experimental changes.

We have used Gazebo, an application aimed at simulating robotics scenes. Gazebo accepts world descriptions in a format called SDF (*Simulation Description Format*) that follows the XML convention. In addition, when combined with ROS, URDF (*Unified Robot Description Format*) robot models can also be loaded into the simulation. We have taken the WAM robot description from the IRI (`iri_wam_description` package), and introduced minor modifications in order to represent the particularities of our PICKER and CATCHER. The modifications are located in the packages `iri_wam_picker_description` and `iri_wam_catcher_description`, respectively. There is also some 3D modeling work behind the preparation of the simulated scene, namely the creation of a toy sphere mesh and the reproduction of all the pieces. We have tried to model these elements with the maximum amount of realism possible, respecting the dimensions of the real objects. Figure 5.3 shows the result.

The package `iri_task_motion_integration_gazebo` contains all the simulation-related content. This includes an scene with the two robotic arms located in (approximately) same position where they are in the real life experiment. Figure 5.4 shows how this scene looks like. The scene is functional in that we can programatically control the robots with other nodes. However, the PICKER does not have a functional gripper so the experiment cannot be fully reproduced. Even so, has been useful to test other matters:

- The workspace of the robot, this is, the volume of space where the inverse kinematics module can provide solutions with reliability.
- The perception techniques, since the Kinect camera in the simulation is functional.
- The basics of robot control. We have employed a recurrent benchmark script that uses the perception pipeline to obtain the location of the blobs and that instructs the picker to position its end-effector (where the gripper should be) over them.

[1] `https://bitbucket.org/sprkrd/integration-of-task-and-motion-planning-for-robotics`
[2] `http://wiki.iri.upc.edu/index.php/LabRobotica`
[3] `https://bitbucket.org/sprkrd/cv_extra`

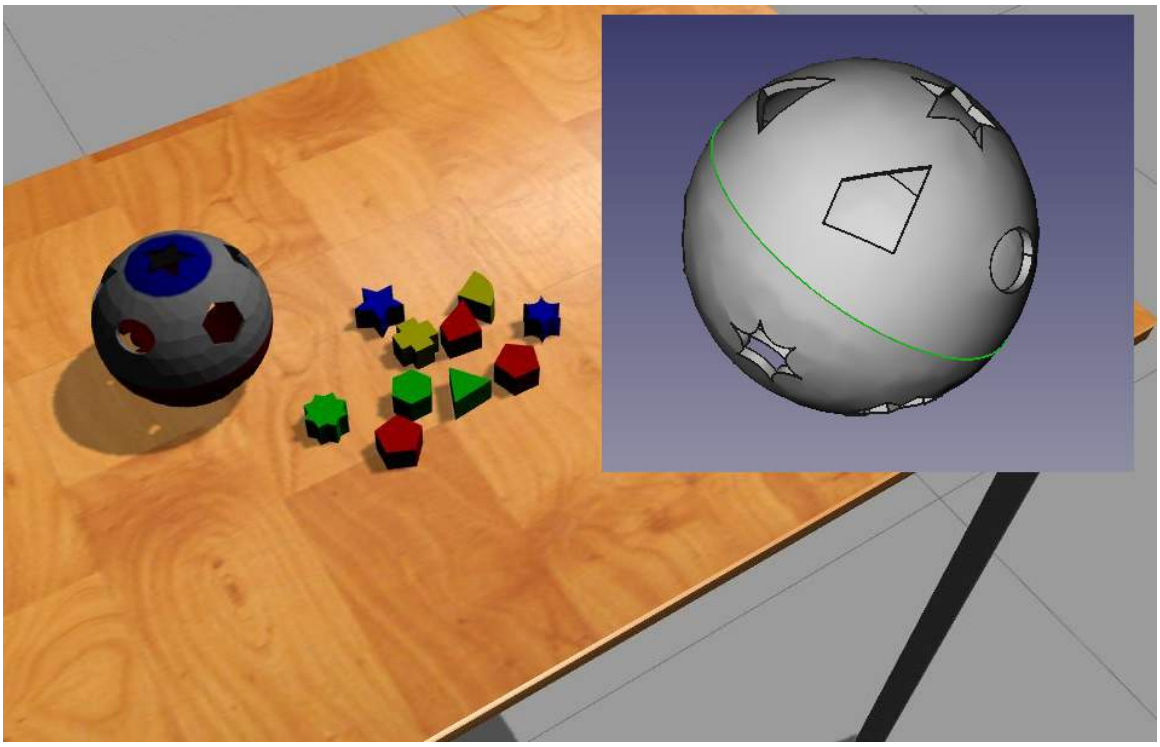

Figure 5.3: At the top corner we have the model of the sphere, modeled in FreeCad. The rest of the picture shows the already textured sphere and the pieces in a Gazebo simulation.

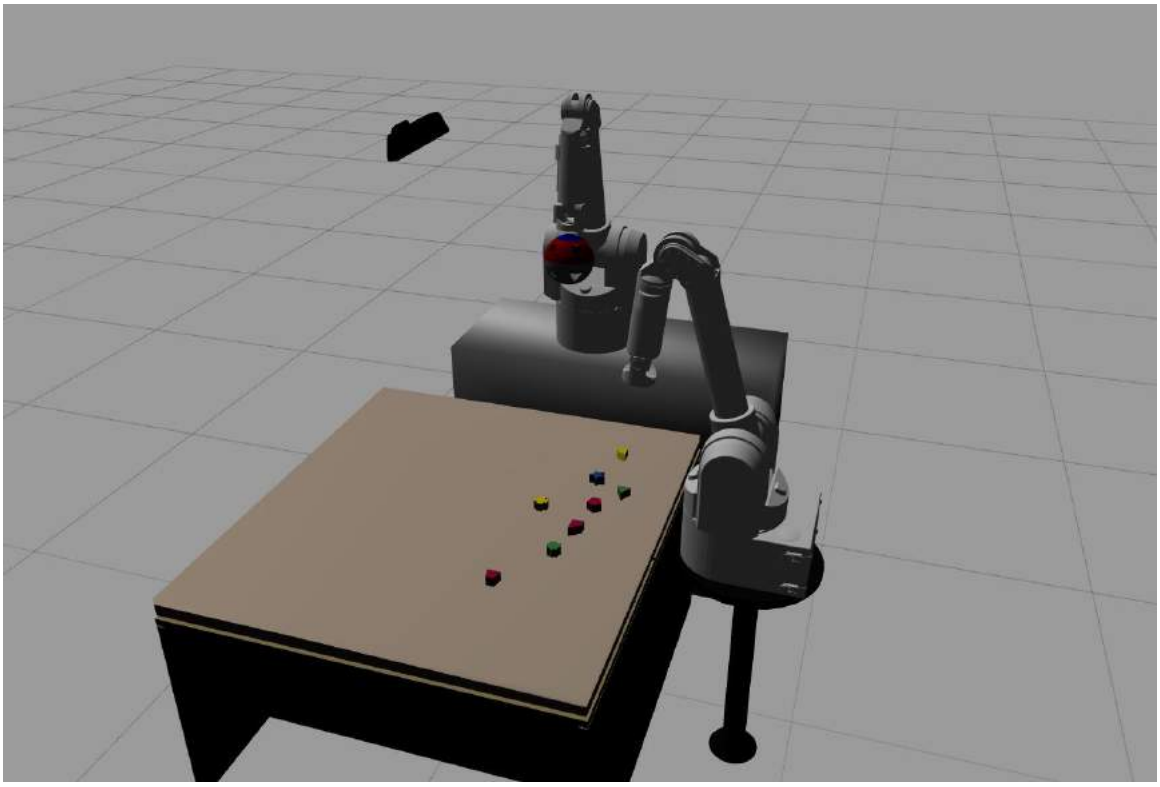

Figure 5.4: Simulated scene in Gazebo. We can see the main elements of the experiment: the Kinect camera at the ceiling (although in the simulation it is actually floating), the two WAM robots (one of them with the sphere attached as an end-effector) and the pieces distributed over the table.

5.3. Perception pipeline

This section is dedicated to the functional blocks that form the perception pipeline. The final purpose of this module is to detect and identify the pieces that are distributed over the table and the cavities of the sphere. We are interested in estimating not only which shape is associated to each piece, but also their rotation. We have identified four stages: filtering, image segmentation, 3D point extraction and shape comparison. In each of these we make a heavy use of OpenCV functionalities. We have also implemented a C++ library (called `cv_extra`) that offers useful methods and classes for each of the perception blocks.

5.3.1. Filtering

We receive the images from a Kinect camera. These images are somewhat noisy. In order to obtain better results in the later phases, we thought it is important to provide a denoising mechanism. We have considered several possibilities:

- **Temporal averaging**: we can average the latest images continuously. If we consider the noise at each pixel to have zero mean and being uncorrelated with the noise at other pixels, averaging similar images that have been taken recently reduces the variance of the noise. Under the assumption that the camera is fixed and the scene objects are not moving, this technique has the advantage of not blurring the edges of the scene. However, the utilization of a temporal averager has the problem of producing an afterimage or trail effect. This is the result of the transitory response of the filter. We have to decide how many images we will average together.

- **Normalized box blur**: one of the most simplest filter. It simply averages the color of each pixel with the color of the surrounding ones (in a square neighborhood). The box filter is a low-pass filter (i.e. it preserves constant and slow gradients). Tipically, noise in images has associated high frequency components so the box filter may probe being effective against it. As an advantage, it is a separable filter and it is very fast to compute. However it may blur edges, and the frequency response of the filter presents side lobes (which means that high frequencies are not completely eliminated). This filter receives as a parameter the neighborhood size.

- **Gaussian filter**: It is very similar to the box filter, but it has a Gaussian kernel instead of a square one. It shares the efficiency advantage, and it fixes the inconvenient of the side lobes. However, it stills blurs edges. As a parameter, it needs the standard deviation of the Gaussian kernel in X and in Y (or a single standard deviation if the filter is isotropic). The higher the standard deviation, the more blurry is the filtered image.

- **Median filter**: This filter substitutes the color at each pixel with the median of the surroinding ones. It is more respectful with edges. However, it rounds the corners of the scene objects and it does not behave well in areas with highly saturated colors.

- **Bilateral filter**: One of the most sophistiated filter. It is non-linear. It performs an average on each pixel's neighborhood based not only on the colour but also on the euclidean distance. In practice this means that the filter does a great job both removing noise and preserving edges. However, it is very slow compared to the other options.

Figure 5.5 shows a comparison of all these filtering methods. We have implemented a ROS node that combines a temporal averager filter with an additional filter (one of the other filters mentioned in this list). Each filter is parametrizable in runtime. We have implemented the temporal averager as a class that contains a queue of images and the sum of all the images in the queue. Whenever a new image enters the queue, we remove the oldest image (if the queue is full) and update the sum accordingly. For the other filters we have used the OpenCV's implementation.

We have found that a combination that provides very good results is the temporal averager with few images (5 or 10) and a isotropic Gaussian filter with low deviation ($\sigma \in [0.5, 1.5]$).

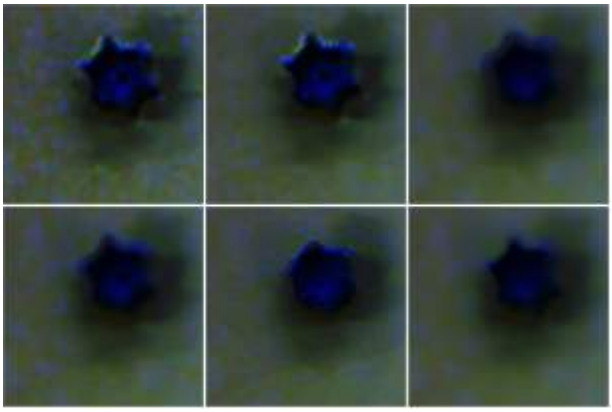

Figure 5.5: Comparison of the considered filters. The images has been equalized so the reader can distinguish better the noisy pixels. From left to right and from bottom to top the images represent: the original image without filtering; the image filtered with a temporal averager of 10 images (we can see the edges of the piece clearly); the image filtered with a 5×5 box filter (the edges of the pieces are more blurred); the image filtered with an isotropic Gaussian filter with $\sigma = 3$ (also blurry); the image filtered with a median filter with a kernel size of 3; and the image filtered with a bilateral filter ($\sigma_{color} = 70$ and $\sigma_{distance} = 5$) which gives great denoising power without affecting the edges too much.

5.3.2. Segmentation

Segmentating an image consists of labeling each pixel depending on the category they belong. In our case, we want to distinguish between the pixels that represent a piece, a cavity or the background. The input to the segmentation stage is the image filtered by the former node.

We have considered several segmentation techniques: K-means, watershed and simple selection based on color. Moreover we have tested with several color spaces to see which yields the best results. At the end we have decided to use two different algorithms for detecting the pieces and for detecting the holes in the sphere. These algorithms are offered as services by the `iri_piece_segmenter` node.

It is interesting to notice that once the image is segmentated we are interested in the connected components (i.e. groups of pixels with the same label). During the document these components is what we have been calling *blob*. Once we have identified a blob, we can compute very easily its contour and its centroid.

Algorithm for the detection of pieces

The algorithm is an hybrid of two of the techniques presented before. First, we convert the image to the HSV color space, since it is more appropiate for evaluating the color of a piece independently of the lighting conditions. Then, for each one of the main colors (yellow, blue, red and green) we select the pixels that fall inside a certain tolerance region. We perform a morphological erosion with the selected pixels in order to remove spurious pixels and to define the *sure foreground* and *sure background* masks. We use these masks as the starting point or seed of the watershed algorithm[4] , that takes care of making a finer segmentation. Among other advantages, the watershed stage does a good job minimizing the number of connected components, avoiding the problem of pieces that are "split asunder" by a simple color selection segmenter. The typical solution to this problem tends to be a succession of morphological transformations in order to connect separate components (morphological closening) and remove spurious pixels (morphological opening), but this is actually harmful for the proper recognition of the shape. Check figure 5.6 to see an example of what the results look like.

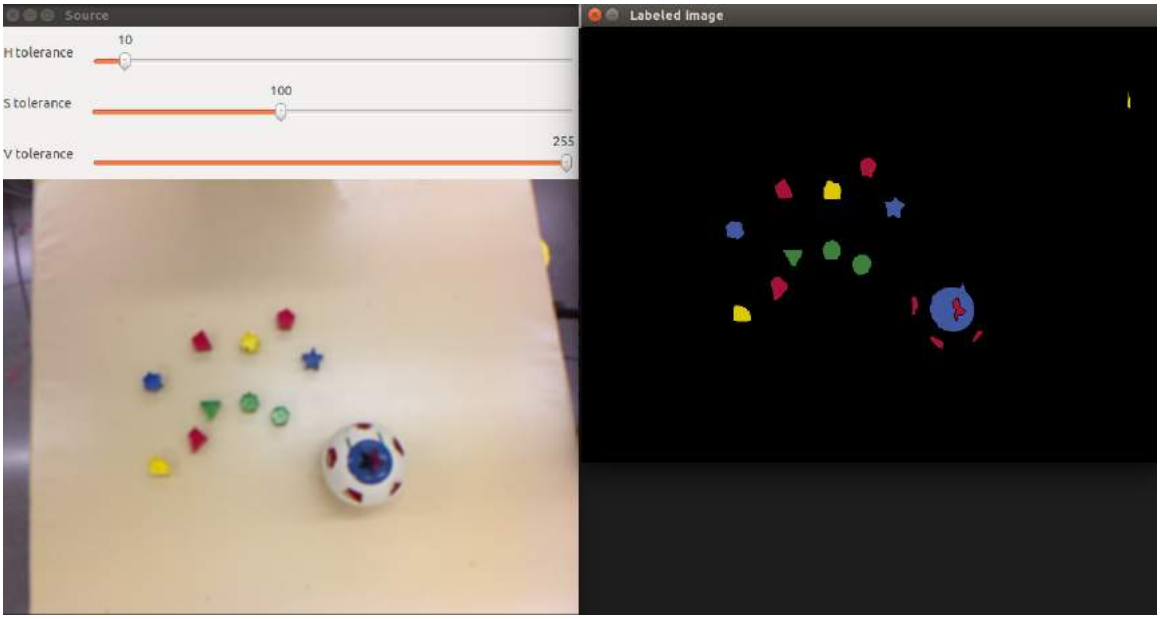

Figure 5.6: Example application that shows the results of the segmentation algorithm for pieces

It is worth mentioning that, in the segmenter node, we have added some extra features. For example, there exists the option of cropping the borders of the image, so the segmenter does not detect anything outside the region of interest. The second one is that the segmenter discards any blob whose area is not greater than a threshold (configurable by the user in runtime). With this we want to remove spurious blobs that have resisted until the end of the algorithm. The node also allows to modify in runtime the HSV range of each colour (red, blue, yellow and green) for the colour selection phase. Algorithm 5 presents a more formal description of the whole segmentation process.

[4]The watershed algorithm is a region growing segmentation technique inspired by how drainage basins are filled with water during precipitations. To learn more about the watershed algorithm and about its usage in colour images we recommend [Meyer, 1992].

Algorithm 5 Piece segmentation procedure

```
procedure SegmentateOnTable(M)                          ▷ Input: already cropped image
    L ← SimpleColorLabeling(M)  ▷ Labeled image with simple colour thresholding
    sureFG ← Erode(L,Circle(r_erode))    ▷ Erosion with a circular structural element
    sureBG ← NonLabeled(Dilate(L,Circle(r_dilate)))
    markers ← ObtainMarkers(sureFG, sureBG)                ▷ Combine BG and FG
    L' ← Watershed(markers)
    detected ← ∅
    for blob ∈ConnectedComponents(L') do
        if blob.label ≠ Background ∩ Area(blob) ≥ A_min then
            detected.Add(blob)
        end if
    end for
    return detected
end procedure
```

Algorithm for the detection of cavities and for individual pieces

The sphere cavities are a more difficult matter, since they cannot be identified by single uniform colors. For this reason, we let the robot that holds the sphere to position it more or less at the same position of the camera's cone of view. Then we use the watershed algorithm, situating the seed at the center of where the cavity should be. Since the cavities share similar lighting conditions, this provides a satisfactory result. Figure 5.7 shows an example.

Figure 5.7: Segmentation of a cavity. Note how the seed is roughly located at the position of the cavity and the watershed algorithm takes care of segmentating the shape in more detail.

The technique for detecting an individual piece held by the Picker and being show to the camera is almost the same. We mainly change the location of the seed and compute the mean of the selected pixels in order to see which is the colour of the piece (the colour is

important since it allows us to compare the detected blob exclusively with the shapes that share colour). Figure 5.8 shows an example.

Figure 5.8: Segmentation of a piece that is being shown to the camera

5.3.3. Obtaining the 3D centroids of pieces and cavities

In order to obtain the images from the Kinect we use a ROS wrapper for the FreeNect[5] driver. The driver also provides a 3D point cloud with each point in a one-to-one correspondence with the pixels of the image (provided we have enabled the *depth registration* option of the driver). Therefore, for the pieces that are distributed over the table we simply check which are the 3D coordinates that are associated to the centroid of the detected blobs.

The situation is quite different for the cavities since there are not 3D coordinates associated to their centroids or the coordinates correspond to a point that is inside the sphere. What we do then is to average the coordinates of all the points that are in a neighborhood of where our cavity should be. We discard all the points that exceed a certain distance from the camera, hereby averaging only the points that are on the surface of the sphere.

Currently, we do not obtain the 3D coordinates of the piece that the PICKER holds when it shows it to the camera.

5.3.4. Obtaining the similitude between shapes

Finally we describe the technique we employ to compare shapes and detected blobs. The purpose is to extract a magnitude that tells how similar they are. We would also like to know the rotation of the piece so we can grab and insert it in an appropriate manner. We have tested some options like feature matching via SURF and SIFT. However, this does not yield satisfactory results since the detected features are not consistent enough. Finally we

[5] `https://openkinect.org/wiki/Main_Page`

have opted for a less sophisticated but very consistent algorithm.

We have gathered a set of patterns that reproduce the shape of the base with greatest area of our pieces. These templates are shown in figure 5.9. We compare each of these patterns with each of the detected pieces. To do so, we scale them so they have roughly the same area. Then we keep rotating the template and overlapping it to the shape. We count the number of pixels that differ (the error). The rotation angle that has the least quantity of errors associated is the optimum angle for that piece and shape. The most likely shape for a particular piece (without considering yet other pieces and the scale matrix technique described in section 4.2.2) is the one that has the smallest error in the optimum angle. The optimum angle indicates the rotation of the piece. We perform two optimizations in order to make this process quicker:

- We compare each piece with only the relevant shapes. For example, among the yellow pieces there are a cross and a circular sector. Therefore, a detected yellow piece should be compared only with the cross and with the circular sector.
- Several pieces have simmetries that make unnecessary to cover angles between 0 and 360 degrees. For example, for the cross it has sense to only consider angles between 0 and 90 degrees.

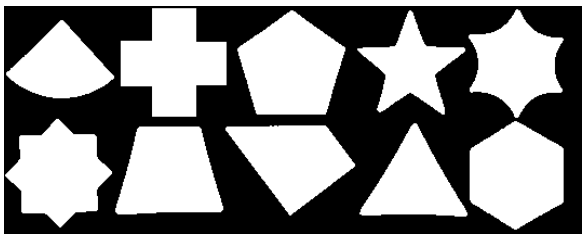

Figure 5.9: Here we show the shapes of the pieces we have at our disposal. From left to right and from top to bottom, we have each one as CircularSector, Cross, Pentagon, Star5, Star6, Star8, Trapezium, Trapezoid, Triangle, Hexagon

.

Figure 5.10 shows an example application that tries to find the most likely shape and the optimum angle for some piece detected via segmentation.

Figure 5.10: Example application that shows the results of the matching algorithm. The center image is the detected shape. At the left we have the most probable template shape and, at the right, how this template looks when rotated at the optimum angle.

Our similitude value is calculated as follows:

$$sim_{ij} \leftarrow 1 - \frac{\text{errors}(i,j)}{\pi r.^2} \tag{5.1}$$

r is radius of the circumference that contains both the template and the detected shape once they are scaled. For this reason, the maximum error is the area of the circumference

and $\pi r.^2$ is a normalization quantity. Therefore, $sim_{ij} \in [0, 1]$, and its value satisfies the requirements of the similitude function described in section 4.2.2.

However, in practice we have seen that this measure is somewhat "benevolent" with shapes that, to our eyes, are clearly different and should receive a much lower similitude value. Figure 5.8, although presented in a different context, is a great example of this. When we took that image we obtained the following similitudes:

- With the PENTAGON: 0.9159
- With the TRAPEZIUM: 0.8202
- With the TRAPEZOID: 0.6816

We would expect a much higher difference between the similitudes to the PENTAGON and the TRAPEZIUM. If we assigned an orientative probability to each shape in a way that it is proportional to the similitude (again, we are not considering other pieces nor matrix scaling), we would obtain $\{0.3788, 0.3392, 0.28192\}$, respectively. For this very reason, we define an alternative measure of the similitude:

$$sim'_{ij} \leftarrow sim^p_{ij}, \quad p \geq 1 \tag{5.2}$$

Raising the similitude to a quantity greater than 1 accentuates the differences between high similitudes. In more precise terms, it compresses the low similitudes and expands the high similitudes (a similar idea is used in image processing for image equalization and in gamma correction). For $p = 4$, we would obtain the following alternative similitudes:

- With the PENTAGON: 0.7037
- With the TRAPEZIUM: 0.4526
- With the TRAPEZOID: 0.2158

And the orientative probabilities would be $\{0.5129, 0.3298, 0.1573\}$, respectively. This seems much more reasonable. An even higher p could, arguably, lead to more intuitive probabilities.

Algorithm 6 presents the whole procedure in a more formal way.

5.4. World interface

This section is devoted to discuss how we have implemented the actions of the robotic arms. We have established a difference between the actions executed by the PICKER and the actions executed by the CATCHER.

IRI WAM Catcher

The CATCHER has the sphere attached as an end-effector. Its mission is to show the relevant cavity to the camera and make the insertion operation easier for the PICKER. This has lead us to implement and action called ASSUMEPOSE. It simply receives the qualitative name of the desired pose. The CATCHER moves and tries to situate the end-effector accordingly. The pose name can be either NEUTRAL or one of the names of the cavities they intend to make available for the PICKER. For example, invoking ASSUMEPOSE(NEUTRAL) would instruct the CATCHER to adopt a pose in which it is not an obstacle for the other robot or for obtaining a clear view of the table from the camera's perspective. Alternatively,

Algorithm 6 Shape comparison procedure

procedure COMPARE($c_1, c_2, \Delta\theta, \theta_{max}$) ▷ Input: contours, θ resolution and maximum θ
  $c'_1 \leftarrow$ SCALE(c_1) ▷ Scale first contour. This is the template.
  $c'_2 \leftarrow$ SCALE(c_2) ▷ Scale second contour so AREA(c'_1) $\approx$ AREA(c'_2)
  $e \leftarrow \infty$ ▷ Minimum error until now
  $\theta_{opt} \leftarrow$ NONE ▷ Optimum angle
  for $\theta \leftarrow 0, \theta < \theta_{max}, \theta \leftarrow \theta + \Delta\theta$ **do**
    $M_1 \leftarrow$ EMPTYBINARYCANVAS()
    $M_2 \leftarrow$ EMPTYBINARYCANVAS()
    DRAWFILLEDROTATEDCONTOUR(c'_1, M_1, θ)
    DRAWFILLEDCONTOUR(c'_2, M_2)
    $M_{different} \leftarrow M_1$ **xor** M_2 ▷ Sets to 1 the different pixels, and to 0 the equal ones
    if COUNTNONZERO($M_{different}$) $< e$ **then**
      $e \leftarrow$ COUNTNONZERO($M_{different}$)
      $\theta_{opt} \leftarrow \theta$
    end if
  end for
  $sim \leftarrow 1 - \frac{e}{\pi \cdot r^2}$ ▷ r is a parameter of the algorithm (related to canvas size)
  (Optional) $sim \leftarrow sim^p$ ▷ This step can be ommited or delayed
  return sim, θ_{opt}
end procedure

ASSUMEPOSE(TRIANGLE) would tell the CATCHER to move the toy sphere to a location where the TRIANGLE cavity is facing upwards and the camera has a clear view of it.

We have made sure that, whenever we order the robot to show a particular cavity, it centers the sphere roughly in the same points, and that it only changes its orientation. This is a requirement of the cavity detection strategy described in section 5.3.2.

In order to guarantee the required precision and avoid slight changes that would harm the cavity detection, adequate pre-computed values of the robot joints has been stored in a dictionary for each pose. We have checked that all the cavities can be detected correctly. When the action is invoked, the node only has to lookup the joints that are associated to the requested pose in the dictionary and move the robot so it assumes the corresponding joint values.

The action has been implemented in a node called `iri_wam_catcher_actions`, and it requires the `joints_move` service from the IRI WAM controller to work. As a configuration option, our node accepts the maximum velocity at which we want the robot to move. It reads the precomputed joints values from a file when executed. It goes without saying that this file can be update if necessary.

IRI WAM Picker

The PICKER supports an ASSUMEPOSE as well, although with a more limited set of possibilities: it can be used to adopt NEUTRAL position (same concept than before) or SHOW position, intended to perform a closer look of a piece that has already been picked by the robot.

The other important actions is PICKORPLACEPIECE, which has multiple purposes. It receives the following information as a goal:

- **centroid**, a 3D point (more specifically, a `geometry_msgs/Point3DStamped` message. It is the location where the gripper should pick or place a certain piece. Care should

be taken when specifying the `frame_id` of the header (this is the ID of the frame of reference of the points in which the point coordinates are measured). If the frame of reference is that of the Kinect, it is important that the transformation between the Kinect and the robot's frame of reference is defined (see more about geometric transformations in section 4.1.1).

- **pick**, a boolean variable that tells if the robot should pick or leave a piece.
- **erratic**, a boolean variable that tells the robot to perform additional erratic movements after having reached the grasp/ungrasp position. The idea behind this feature is to increase the probability of inserting the pieces into the sphere. The precision required to achieve this feat is quite high and we think that some erratic movement can be beneficial.

This action depends on the inverse kinematics module from the IRI to work since it needs to know how to reach the grasping/ungrasping point. Not just that: the actions also computes a pre(un)grasp point that is located at a fixed distance above the grasp/ungrasp point. This pre(un)grasp point is an intermediate point whose purpose is to force that the robot moves the gripper to the grasping/ungrasping position in a straight line from above (approximately).

This package is similar in many aspects to the `iri_wam_generic_pickorplace` package from the IRI, and indeed we would have used this one if it was not because we also wanted a feature to move erratically.

We can see an example of how the two robots perform cooperatively at figure 5.11.

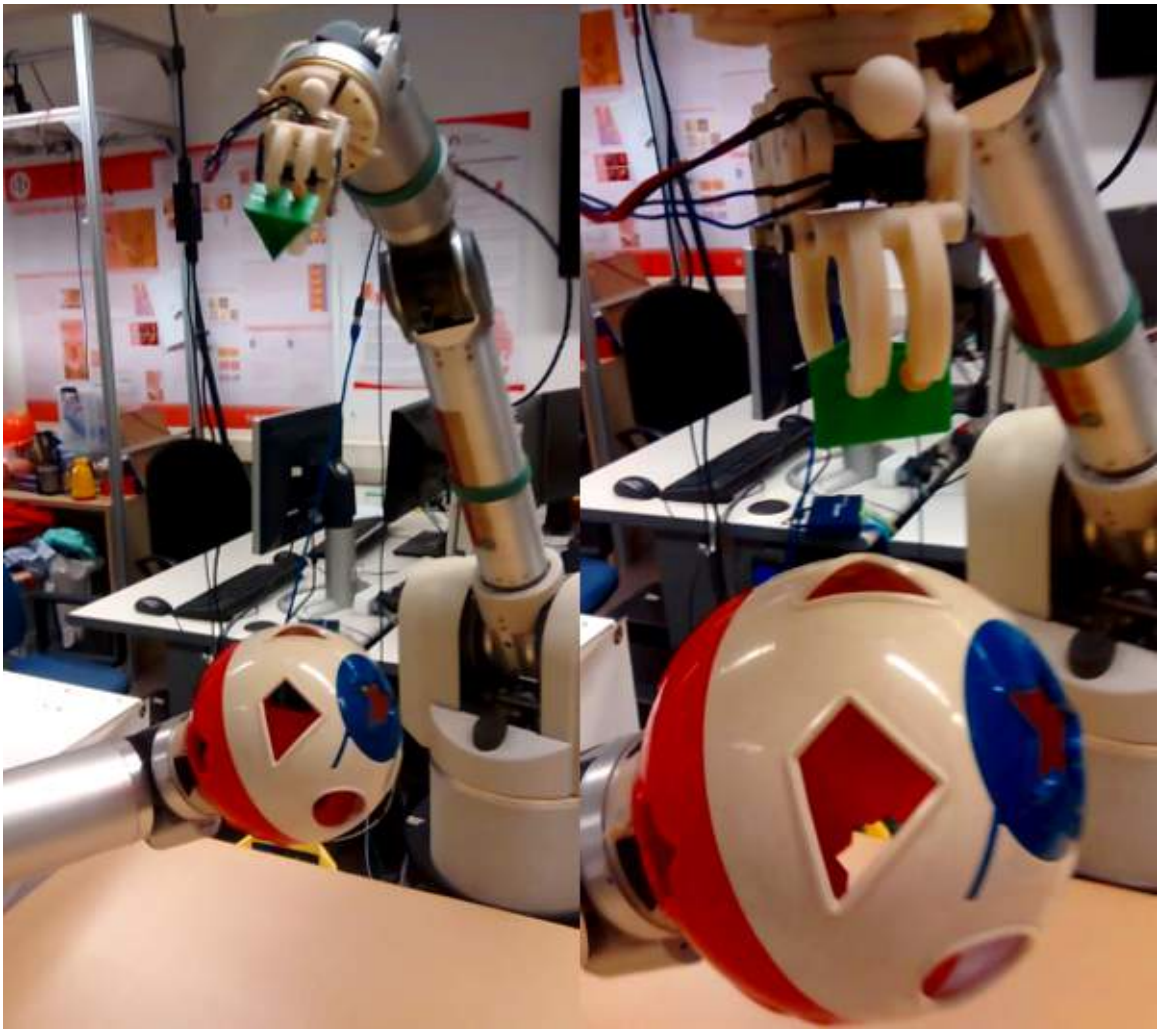

Figure 5.11: Insertion of a piece in the sphere. Here we can see the CATCHER exposing the triangle cavity and the PICKER approaching the sphere in order to insert the piece. To do that we have invoked the PICKORPLACE action with **pick** set to *false*.

6

Experiments and conclusions

In this chapter we cover the experimentation process and present our conclusions about the whole project.

6.1. Description of the experimentation process

We wanted to test several ideas. The first is how well performs the entropy when dealing with contingencies. In our case, these contingencies could be present in the form of pieces that have been poorly picked (or not picked at all). We also wanted to see how well we could control how risky were the decisions taken by the planner based on the tuning of the operator's costs and the minimum required similitude for inserting a piece without feedback. The effectiveness of the matrix scaling technique is also under our scope. For this this reason we propose the following experiments:

- **Test with a single piece**. We shall choose one of the pieces and see how the whole system works. We want to check how many observations are performed before the robots act and if this number of observations reduce the uncertainty about the piece location and rotation satisfactorily. This experiment should also expose how well the actuation and perception parts of our work perform.
- **Test with a piece from each color**. Theoretically the whole system should work in a very similar way. There is not uncertainty about the identity of each blob since the colour already gives them away. Even so we want to test this in practice, and this is the logical step before continuing with more complex experiments.
- **Test with several pieces from the same colour**. The aim of this experiment is to put into practice the matrix scaling technique. It should also provide insight on how well we can control the tendency of the planner to ask for feedback with different operator's costs and the suitability of the entropy for this matter.
- **Testing with mixed colours and several pieces per colour**. The most complex version of the problem. This experiment addresses all the previous elements simultaneously.

6.2. Results

While we did not have the opportunity of performing all the experiments in their full extend, we still could take some notes about the behaviour of the planner under at least two different circumstances.

Test with a single piece

When testing the whole system with a single piece we can tell the following:

- How many observations will the system perform before acting. Naturally, the number depends on the parameters ρ, on the tolerance δ and on the maximum allowable uncertainty about the estimation ϵ. When These two last values fixed the minimum number of observations while ρ actually had influence on how many extra observations would be performed after reaching the minimum.
- Modifying the minimum similitude b_{min} to 0 or 1 we were able to force the system to prompt the user for feedback or to always take the risky decision of inserting the piece without feedback. Setting b_{min} to an intermediate value would lead to a compromise. Certain times the PICKER would grab a piece precariously. After this, showing it to the camera would yield a low similitude value. Even if it is the only piece of the problem, the b_{min} threshold is not reached and the planner prompts the user for feedback anyway. Alternatively, when the grasp is good, the similitude is high and the system does not ask for feedback. We think that picture 6.1 is very illustrative in this matter.
- Other than that we were able to test how the system performed in terms of perception and actuation. The weakest part is the insertion of the piece in the sphere since it requires great precision. The piece alignment is good. The PICKER tries to insert it with the correct orientation. However, small displacements made that the piece did not fall completely inside the sphere. A small push from the nearest person was required.

6.2.1. Test with just one piece per colour

The test with only one piece per colour was pretty much like executing several tests with a single piece. The segmenter did not have any problem identyifing and labeling the blobs correctly. This test allowed us to see that the system was able to handle several pieces correctly.

6.3. Final regards

We wanted to test a system that performs both symbolic and geometrical planning. The geometrical planning part is not evidenced as the planning of paths or trajectories that avoid obstacles since the volume in which the robots have to move is mostly free of obstacles and we can avoid collisions between them with a careful selection of methods and task ordering.

Instead the geometrical constraints are reflected in the need of conducting observation actions. Not only that, the planner is aware that these observations are noisy and that it has to use many of them in order to estimate the desired magnitude correctly. While the

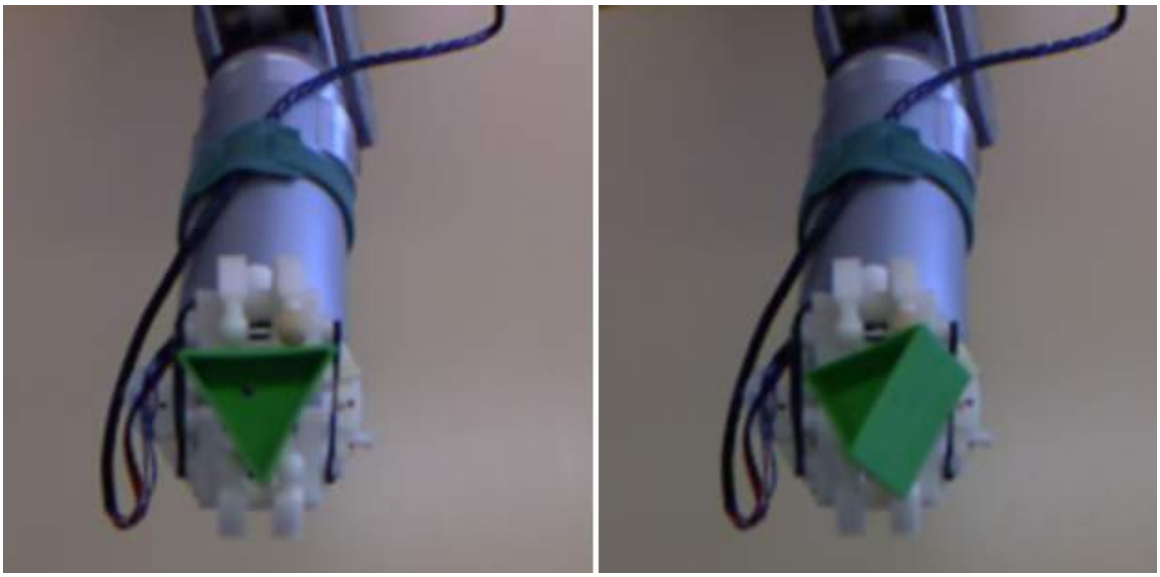

Figure 6.1: Comparison between a good grip (left) and a bad grip (right). The similitudes between the detected blobs in both cases and the triangle shape were, respectively, 0.78 and 0.45. Therefore, a value of $b_{min} = 0.5$ would successfully distinguish each situation and ask for feedback accordingly.

operator does not deal directly with the numeric values of the coordinates and the rotations (it does not need them for anything) it knows that these variables exist and that enough observations are required in order to calculate a good estimation. The locations of the pieces are required in order to grasp them, and their rotations are needed so we can insert them in the sphere successfully. This contrasts with a symbolic domain that always assumes that the world state is known and does not care about geometrical restrictions whatsoever (e.g. the well-known "blocks-world" problem).

Also we have introduced several ideas that we think can be interesting for future work in this line:

- The use of matrix scaling for stablishing bijections between sets of equal cardinalities based on a similitude function between elements.
- The use of the entropy of discrete states to establish a decision criterion based on uncertainty. In our case we have used this to decide whether to ask for feedback nor act without supervision.

Of course, we also recognize the limitations and possible improvements of our work, namely:

- The implementation is highly experimental. In the future we would like to take advantage of the latest advancements in planning engines.
- Although they are not in the main scope of our project, the perception and actiation techniques can, of course, be improved. The PICKER acts in an open loop, deciding where to go based just on the calibration between the robot and the camera.

As an additional idea for future work, and from a more technology-oriented point of view, we would like to implement an interface between one of the latest planners (e.g. PROST or Gourmand) and ROS. We think that a standarized API for symbolic planning in ROS would indeed be beneficial for encouraging more research and experimentation in the area of planning. Currently there exists a package with a similar idea in the ROS catalogue

of packages[1]. However it is slightly outdated and for an old version of ROS. Even so, it provides very interesting features like a monitoring GUI (*Graphical User Interface*).

All in all, we hope that some, if not all, of our experiences and idea may be useful for future work on this or related topics.

[1] `http://wiki.ros.org/symbolic_planning`

Appendices

A

Proof of mathematical expresions

A.1. Optimality of the mode estimator based on the sample mean

The estimator based on the sample mean is clearly unbiased (i.e. $\mathrm{E}(\hat{X}_n) = \bar{X} \quad \forall n$). We demonstrate here that the estimator accomplishes the Cramér-Rao bound with equality.

First let us remind the inequality:

$$\mathrm{var}(\hat{X}) \geq \frac{1}{\mathrm{I}(\hat{X})}, \quad \mathrm{I}(\hat{X}) = \mathrm{E}\left[\left(\frac{\partial \ell(\mathbf{X};\bar{X})}{\partial \bar{X}}\right)^2\right]$$

In this expression, $\ell(\mathbf{X};\bar{X})$ is the natural logarithm of the likelihood function, that measures how likely is a certain set of observations for the given mode $\bar{X}$. In our case, the likelihood function is the product of the Gaussian PDFs (since the observations are independent) particularized to each one of the observations. This means that:

$$\ell(\mathbf{X};\bar{X}) = \log\left(\frac{1}{\sqrt{(2\pi\sigma_x)^n}}\right) + \sum_{i=1}^{n} \frac{(X_i - \bar{X})^2}{2\sigma_x^2}$$

where n is the number of observations. Then:

$$\frac{\partial \ell(\mathbf{X};\bar{X})}{\partial \bar{X}} = -\sum_{i=1}^{n} \frac{X_i - \bar{X}}{\sigma_x^2}$$

$$\left(\frac{\partial \ell(\mathbf{X};\bar{X})}{\partial \bar{X}}\right)^2 = -\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{X_i - \bar{X}}{\sigma_x^2}\frac{X_j - \bar{X}}{\sigma_x^2} = -\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{X_iX_j - X_i\bar{X} - X_j\bar{X} + \bar{X}^2}{\sigma_x^4}$$

The last step is to compute the expected value of the former expression:

$$\mathrm{E}\left[\left(\frac{\partial \ell(\mathbf{X};\bar{X})}{\partial \bar{X}}\right)^2\right] = -\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\mathrm{E}(X_iX_j) - \mathrm{E}(X_i)\bar{X} - \mathrm{E}(X_j)\bar{X} + \bar{X}^2}{\sigma_x^4} =$$

$$\frac{(n^2\bar{X}^2 + n\sigma_x^2) - n^2\bar{X}^2 - n^2\bar{X}^2 + n^2\bar{X}^2}{\sigma_x^4} = \frac{n}{\sigma_x^2}$$

Therefore, $\mathrm{var}(\hat{X}^2) \geq \frac{\sigma_x^2}{n}$. We have already seen in equation 4.4 that this lower bound is the variance of the sample mean estimator. Therefore, the Cramér-Rao bound is reached with equality and we are done.

A.2. Proof of expression 4.6

In equation 4.6 we saw that the optimum way of updating an estimator that follows a Gaussian distribution with another Gaussian observation is:

$$\hat{X}_{new} = \frac{\sigma^2_{obs}}{\sigma^2_{obs} + \sigma^2_{current}} \hat{X}_{current} + \frac{\sigma^2_{current}}{\sigma^2_{obs} + \sigma^2_{current}} X_{obs}$$

This is obtained as the result of a simple optimization problem. We suposse a generic weight of α for the old estimator, and a weight of $1 - \alpha$ for the second (it must be this way or the new estimator would be biased):

$$\hat{X}_{new} = \alpha \hat{X}_{current} + (1 - \alpha) X_{obs}$$

We want the value of α that minimizes the variance of the new estimator:

$$\sigma^2_{new} = \alpha^2 \sigma^2_{current} + (1 - \alpha)^2 \sigma^2_{obs}$$

We calculate for which α the derivative is 0:

$$\frac{\mathrm{d}\sigma^2_{new}}{\mathrm{d}\alpha} = 2\alpha\sigma^2_{current} + 2(1 - \alpha)\sigma^2_{obs} = 0$$

This is a simple linear equation. We solve it for α to obtain:

$$\alpha = \frac{\sigma^2_{obs}}{\sigma^2_{obs} + \sigma^2_{current}}, \quad 1 - \alpha = \frac{\sigma^2_{current}}{\sigma^2_{obs} + \sigma^2_{current}}$$

and we are done.

A.3. Proof of expression 4.8

This result comes as a direct consequence of **Proposition 4** from [Kaelbling and Lozano-Pérez, 2013]. The authors were interested in planning backwards from the goal, so they have come to the following result (adapting the authors' notation to ours):

$$\epsilon_{current} = 1 - \mathrm{erf}\left(\sqrt{\mathrm{erf}^{-1}\left(1 - \epsilon_{new}\right)^2 - \frac{\delta^2}{2\sigma^2_{obs}}}\right)$$

We can obtain equation 4.8 just rearranging the terms of the former expression and isolating ϵ_{new}.

A.4. Proof of expression 4.10

First we obtain the probability of the estimation being inside a δ-sphere centered on the mode. This can be obtained integrating the multivariate Gaussian PDF in spherical coordinates:

$$\int_0^{\pi} \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\phi \int_0^{\delta} \mathrm{d}r \frac{1}{\sqrt{(2\pi)^3}\sigma^3_{xyz}} \exp\left(-\frac{r^2}{2\sigma^2_{xyz}}\right) r^2 \sin\theta$$

We can solve in a single step the two outer integrals and obtain:

$$\frac{4\pi}{\sqrt{(2\pi)^3}\sigma^3_{xyz}} \int_0^{\delta} \mathrm{d}r \, \exp\left(-\frac{r^2}{2\sigma^2_{xyz}}\right) \cdot r^2$$

At this point we can apply integration by parts, with $\mathrm{d}v = r \cdot \exp\left(-\frac{r^2}{2\sigma_{xyz}^2}\right)$ and $u = r$. After rearranging terms we would get the following:

$$\frac{2}{\sqrt{\pi}} \int_0^{\frac{\delta}{\sqrt{2}\sigma_{xyz}}} \exp\left(-\tau^2\right) \mathrm{d}\tau - \frac{2}{\sqrt{\pi}} \frac{\delta}{\sqrt{2}\sigma_{xyz}} \exp\left(\frac{-\delta^2}{2\sigma_{xyz}^2}\right)$$

In the previous expression:

$$\frac{2}{\sqrt{\pi}} \int_0^{\frac{\delta}{\sqrt{2}\sigma_{xyz}}} \exp\left(-\tau^2\right) \mathrm{d}\tau = \mathrm{erf}\left(\frac{\delta}{\sqrt{2}\sigma_{xyz}}\right)$$

Therefore we have that:

$$\Pr(\mathrm{dist}((\bar{X}, \bar{Y}, \bar{Z}), (\hat{X}, \hat{Y}, \hat{Z})) < \delta) = \mathrm{erf}\left(x\right) - \frac{2}{\sqrt{\pi}} x \cdot \exp\left(-x^2\right)\Big|_{x=\frac{\delta}{\sqrt{2}\sigma_{xyz}}} = g(\frac{\delta}{\sqrt{2}\sigma_{xyz}})$$

And:

$$\Pr(\mathrm{dist}((\bar{X}, \bar{Y}, \bar{Z}), (\hat{X}, \hat{Y}, \hat{Z})) < \delta) = 1 - g(\frac{\delta}{\sqrt{2}\sigma_{xyz}})$$

At this point, we are done.

References

[Beetz et al., 2011] Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mosenlechner, L., Pangercic, D., Ruhr, T., and Tenorth, M. (2011). Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536. IEEE.

[Bohlin and Kavraki, 2000] Bohlin, R. and Kavraki, L. E. (2000). Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 521–528. IEEE.

[Bonet and Geffner, 2003] Bonet, B. and Geffner, H. (2003). Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS*, volume 3, pages 12–21.

[de Silva et al., 2013] de Silva, L., Pandey, A. K., Gharbi, M., and Alami, R. (2013). Towards combining htn planning and geometric task planning. *arXiv preprint arXiv:1307.1482*.

[Erol et al., 1996] Erol, K., Hendler, J., and Nau, D. S. (1996). Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93.

[Erol et al., 1995] Erol, K., Nau, D. S., and Subrahmanian, V. S. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1):75–88.

[Fikes and Nilsson, 1972] Fikes, R. E. and Nilsson, N. J. (1972). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208.

[Ghallab et al., 2014] Ghallab, M., Nau, D., and Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17.

[Hockstein et al., 2007] Hockstein, N., Gourin, C., Faust, R., and Terris, D. (2007). A history of robots: from science fiction to surgical robotics. *Journal of robotic surgery*, 1(2):113–118.

[Ilghami and Nau, 2003] Ilghami, O. and Nau, D. S. (2003). A general approach to synthesize problem-specific planners. Technical report, DTIC Document.

[Ilghami et al., 2002] Ilghami, O., Nau, D. S., Munoz-Avila, H., and Aha, D. W. (2002). Camel: Learning method preconditions for htn planning. In *AIPS*, pages 131–142.

[Kaelbling and Lozano-Pérez, 2011] Kaelbling, L. P. and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1470–1477. IEEE.

[Kaelbling and Lozano-Pérez, 2013] Kaelbling, L. P. and Lozano-Pérez, T. (2013). Integrated task and motion planning in belief space. *The International Journal of Robotics Research*.

[Keller and Eyerich, 2012] Keller, T. and Eyerich, P. (2012). PROST: Probabilistic Planning Based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 119–127. AAAI Press.

[Kolobov et al., 2012] Kolobov, A., Mausam, and Weld, D. (2012). LRTDP vs. UCT for online probabilistic planning. In *AAAI Conference on Artificial Intelligence*.

[LaValle, 1998] LaValle, S. M. (1998). Rapidly-exploring random trees a new tool for path planning.

[LaValle, 2006] LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

[Lozano-Pérez and Kaelbling, 2014] Lozano-Pérez, T. and Kaelbling, L. P. (2014). A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3684–3691. IEEE.

[Martínez et al., 2014a] Martínez, D., Alenyà, G., Jimenez, P., Torras, C., Rossmann, J., Wantia, N., Aksoy, E. E., Haller, S., and Piater, J. (2014a). Active learning of manipulation sequences. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5671–5678. IEEE.

[Martínez et al., 2014b] Martínez, D., Alenya, G., and Torras, C. (2014b). Finding safe policies in model-based active learning.

[Martınez et al., 2015] Martınez, D., Alenya, G., and Torras, C. (2015). V-min: Efficient reinforcement learning through demonstrations and relaxed reward demands.

[Martínez et al., 2016] Martínez, D., Alenya, G., and Torras, C. (2016). Relational reinforcement learning with guided demonstrations. *Artificial Intelligence*.

[Meyer, 1992] Meyer, F. (1992). Color image segmentation. In *Image Processing and its Applications, 1992., International Conference on*, pages 303–306. IET.

[Nau et al., 1999] Nau, D., Cao, Y., Lotem, A., and Munoz-Avila, H. (1999). Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 968–973. Morgan Kaufmann Publishers Inc.

[Nau et al., 2003] Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An HTN planning system. *J. Artif. Intell. Res.(JAIR)*, 20:379–404.

[Nau et al., 2015] Nau, D. S., Ghallab, M., and Traverso, P. (2015). Blended planning and acting: Preliminary approach, research challenges. In *AAAI*, pages 4047–4051.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

[Sanner, 2010] Sanner, S. (2010). Relational dynamic influence diagram language (RDDL): Language description. `http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`.

[Silver and Veness, 2010] Silver, D. and Veness, J. (2010). Monte-carlo planning in large POMDPs. In *Advances in neural information processing systems*, pages 2164–2172.

[Sinkhorn, 1964] Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879.

[Somani et al., 2013] Somani, A., Ye, N., Hsu, D., and Lee, W. S. (2013). Despot: Online POMPD planning with regularization. In *Advances In Neural Information Processing Systems*, pages 1772–1780.

[Stone, 2004] Stone, W. L. (2004). *Robotics and automation handbook*, chapter 1. CRC press.

[Şucan et al., 2012] Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82. `http://ompl.kavrakilab.org`.