Universidad
Carlos III de Madrid
www.uc3m.es

Departamento de Ingeniería de
Sistemas y Automática

PhD Thesis

Robot Imagination System

Author:
Juan Carlos
González Víctores

Doctoral Advisors:
Dr. Carlos Balaguer
Dr. Alberto Jardón

Leganés, July 2014

Universidad Carlos III de Madrid

Departamento de Ingeniería de Sistemas y Automática

PhD Thesis

El tribunal aprueba la tesis doctoral titulada "Robot Imagination System", realizado por Juan Carlos González Víctores y dirigida por Dr. Carlos Balaguer Bernaldo de Quirós y Dr. Alberto Jardón Huete.

Firma del Tribunal Calificador:

Presidente: Thrishantha Nanayakkara _____

Vocal: Vicente Matellán Olivera _____

Secretario: Santiago Garrido Bullón _____

Calificación: _____

Leganés, July 2014

- To Lorena, family and friends. I love you.

Check out the Summer 2011 album recorded with Small Hotel Room for more credits. :)
...and of course, thank You!

Acknowledgements

This work was supported by MINECO ministry of Spanish Government within grant DPI2010-21047-C02-01 ARCADIA, and within S2009/DPI-1559 ROBOCITY2030 phase II grant by the Comunidad de Madrid and EU structural funds.

Santiago Morante Cendrero provided the Enhanced Prediction Algorithm that has been introduced in the Inference chapter, and also the Continuous Goal-Directed Actions paradigm that is used within the Execution chapter. He additionally wrote many Python snippets that were later integrated within the software architecture, and did a great deal of writing in the related publications. Hope to see your thesis soon!

Juan Miguel García Haro made TEO the humanoid robot available and operative, taking time he could have spent with his own thesis. Hope to see your thesis too!

Carlos Balaguer and Alberto Jardón have been great doctoral advisors, and have given me the opportunity of being part of the Robotics Lab research group. Thank you so much.

Abstract

This thesis presents the Robot Imagination System (RIS). This system provides a convenient mechanism for a robot to learn a user's descriptive vocabulary, and how it relates to the world for action. With RIS, a user can describe unfamiliar objects to a robot, and the robot will understand the description as long as it is a combination of words that have been previously used to describe other objects.

One of the core uses of the RIS functionality is object recognition. Allowing requests with word combinations that have never been presented before together is well beyond the scope of many of the most relevant state of the art object recognition systems. RIS is not limited to object recognition. Through the use of evolutionary algorithms, the system endows the robot with the capability of generating a mental model (imagination) of a requested unfamiliar object. This capability allows the robot to work with this newly generated model within its simulations, or to expose the model to a user by projecting it on a screen or drawing the mental model as feedback so the user can provide a more detailed description if required.

A new paradigm for robot action based on consequences on the environment has been integrated within the RIS architecture. Changes in the environment are continuously tracked, and actions are considered complete when the performed effects are closest to the desired effects, in a closed perception loop. Experimental validations have been performed in real environments using the humanoid robot Teo, bringing the Robot Imagination System closer to everyday household environments in the near future.

Resumen

Esta tesis presenta el Sistema de Imaginación para Robots (RIS, por sus siglas en inglés). Este sistema proporciona un mecanismo conveniente para que un robot aprenda el vocabulario que un usuario utiliza para descripciones, y cómo esto se relaciona con el mundo. Con RIS, un usuario puede describir un objeto desconocido, y el robot entenderá la descripción mientras ésta sea una combinación de palabras que hayan sido utilizadas previamente para describir otros objetos.

Uno de los usos principales de la funcionalidad de RIS es el reconocimiento de objetos. Permitir consultas con combinaciones de palabras que nunca han sido presentadas juntas con anterioridad sobrepasa el alcance de una gran porción de los sistemas relevantes de reconocimiento de objetos del estado del arte. RIS no está limitado al reconocimiento de objetos. A través de algoritmos evolutivos, el sistema proporciona a un robot la capacidad de generar un modelo mental (imaginación) a partir de la consulta de un objeto desconocido. Esta capacidad permite que el robot trabaje con este modelo nuevamente generado en sus simulaciones, o exponer el model a un usuario proyéctandolo en una pantalla o dibujando el modelo mental para cerrar un lazo donde el usuario puede proporcionar una descripción más detallada si esto se requiere.

Un nuevo paradigma de la acción en robots, basado en las consectuencias en el entorno, ha sido introducido en la arquitectura RIS. Los cambios en el entorno se monitorizan continuamente, y las acciones se consideran completas cuando los efectos realizados maximizan su parecido a los efectos deseados, en

un lazo cerrado de percepción. Se han realizado validaciones experimentales en entornos reales utilizando el robot humanoids Teo, acercando el Sistema de Imaginación para Robots a entornos domésticos cotidianos para un futuro cercano.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern technologies are progressively being incorporated into our everyday lives. We find ourselves immersed in environments that are flooded with advanced embedded electronics and the latest software upgrades. Technology is everywhere: home entertainment systems, computers, mobile phones, tablets, netbooks, ebooks. The scope of growing domestic technologies is also growing beyond electronics and software: domotic actuators and small robots have been making their move into our home environment. Iocchi et al. pointed out that there are already several millions of robots used for basic household chores (e.g. vacuum cleaning) [7].

1.1 Origin of this Thesis

One question that arises is if this growth has been matched by comfort of use across this wide range of devices. Inexperienced people and even infants are able to interact with touch screens and buttons, navigating through tabs, menus, and icons. However, as complexity of scenarios and user specifications increase, the complexity of command interfaces also increases. For instance, one could devise a 42-button controller for the movement of a 21 degree-of-freedom humanoid robot, but this happens to be impractical to perform real world tasks. A work of Ariki et al. is focused on mapping high-dimensional

movements to low-dimensional controllers [8]. However, a drawback within Ariki's approach is that the mapping is tedious and task-specific. The author of this thesis has also worked on enabling robot task creation on devices with which users may be previously acquainted, such as mobile phones, tablets, or video-game controllers [9][10][11]. In these works, users recorded words or sentences, and separately programmed sequences of robotic movements as tasks. Then, the users linked the recorded words or sentences to trigger the execution of these pre-programmed tasks. These publications, while novel and useful to a certain extent, also depict a number of severe shortcomings of classical robot programming when it comes to programming tasks that will be performed in domestic environments.

A. There are no perceptual links or semantic relationships between the physical characteristics of objects and the words used to describe them.

B. There is a lack of inference capabilities for interpreting commands that include combinations of words that have never been taught together, but have been taught separately.

C. There are no bindings between actions and their effects. If there are changes in the environment, the robot's program must be modified in order to achieve the same effect.

Certain modern approaches overcome some of these limitations of classical robot programming. For instance, supervised learning, a subset of machine learning algorithms, allows training a system with data that has been "labeled" by a user [12]. In the context of computer vision in a domestic environment, a user can train a system that incorporates supervised learning techniques by providing it with images of the objects to work with and their names (hence "labeled" data). While such a classifier provides a basic implementation of (A), the knowledge reuse capability of (B) requires superior inference capabilities.

Similarly, techniques such as robot imitation, also known as robot learning by demonstration (LbD) or programming by demonstration (PbD) [13], aim to overcome the need for teleoperation and manual hard-coding of every robot behavior [14]. These techniques simplify programming robot actions. However, in general they do not solve the issue of (C). Tasks are usually encoded as trajectories in the robot motor joint space, where effects on the environment are not stored or asserted.

1.2 Main Objectives

With the previously presented list of shortcomings of classical robot programming and modern techniques in mind, this thesis aims to overcome these general problems by providing specific solutions.

A. To develop a framework that allows linking physical characteristics of objects and the words used to describe them.

B. To enable inference mechanisms that allow a robot to work with combinations of words used to describe objects, even if these words have never been previously taught together.

C. To allow a robot to act according to the effect desired on an object, instead of relying on pre-programmed trajectories alone.

D. Development, integration, experimentation and validation will be considered essential in determining success.

In a broader sense, this thesis also attempts to be a review on the author's most relevant and recent works in the field of robotics, providing published and previously unreleased material. This work as a whole attempts to be an objective yet critical retrospective, analyzing errors from the past, paving the way for the future.

1.3 Target Scenario

Imagine the following scenario: A woman has bought a robot companion, and has activated a system that complies with the objectives of this thesis. She spends an initial period of time teaching the robot her own vocabulary, centered on objects of her home environment, with sentences such as "this is a red pen" and "this is a blue eraser". This initial phase, which could be similar to teaching a child, is depicted in the first two frames of Figure 1.1.

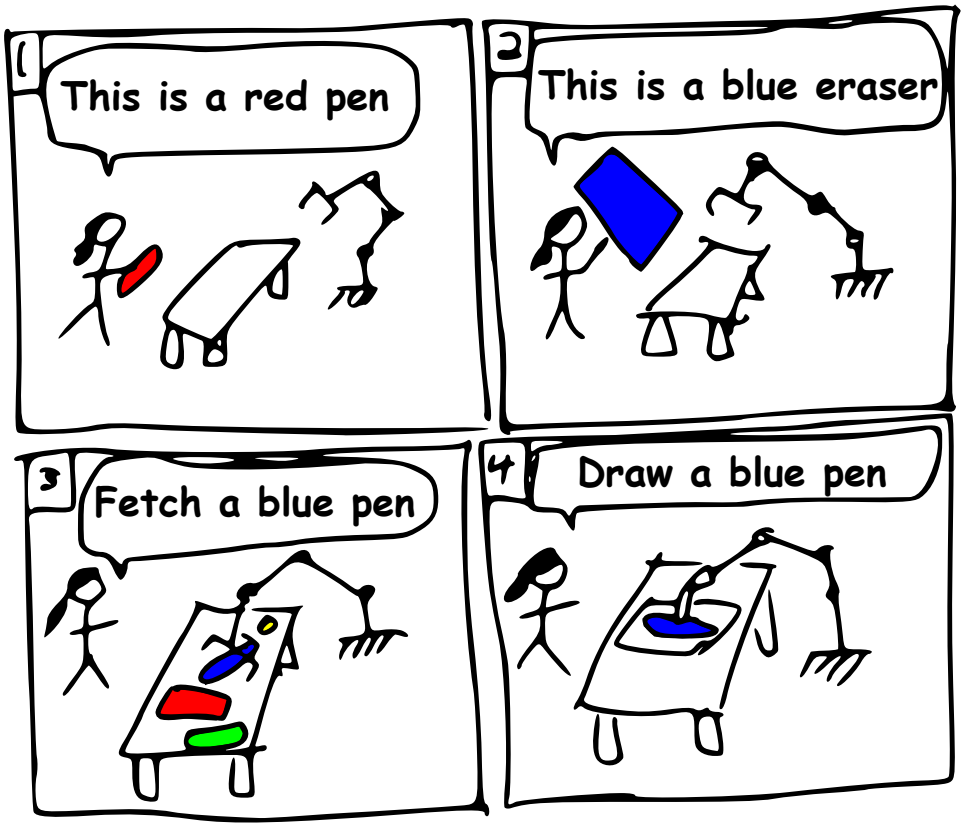

Figure 1.1: Sketch of the Robot Imagination System target scenario.

Once the robot has acquired enough vocabulary, she may benefit from the system's inference capabilities. For instance, she could ask the robot to fetch her "blue pen", and the robot might be able to find it without ever having seen that or any other blue pen in the past, as seen in the third frame of Figure 1.1. Note that she never has to indicate any kind of category with sentences such as "red and blue are colors" or "pens and erasers are objects".

As an additional feature, the robot can be asked to draw a sketch of the object for which it is looking, as depicted in the last frame of Figure 1.1. It may also display it on a screen if available.

1.4 Expected Novelties

While a broader view of all the related works will be seen in chapter 2 of this thesis, a number of expected novelties can already be listed. The final developments will be tested on a real humanoid robotic platform, Teo [15], which can be seen performing a task in Figure 1.2.

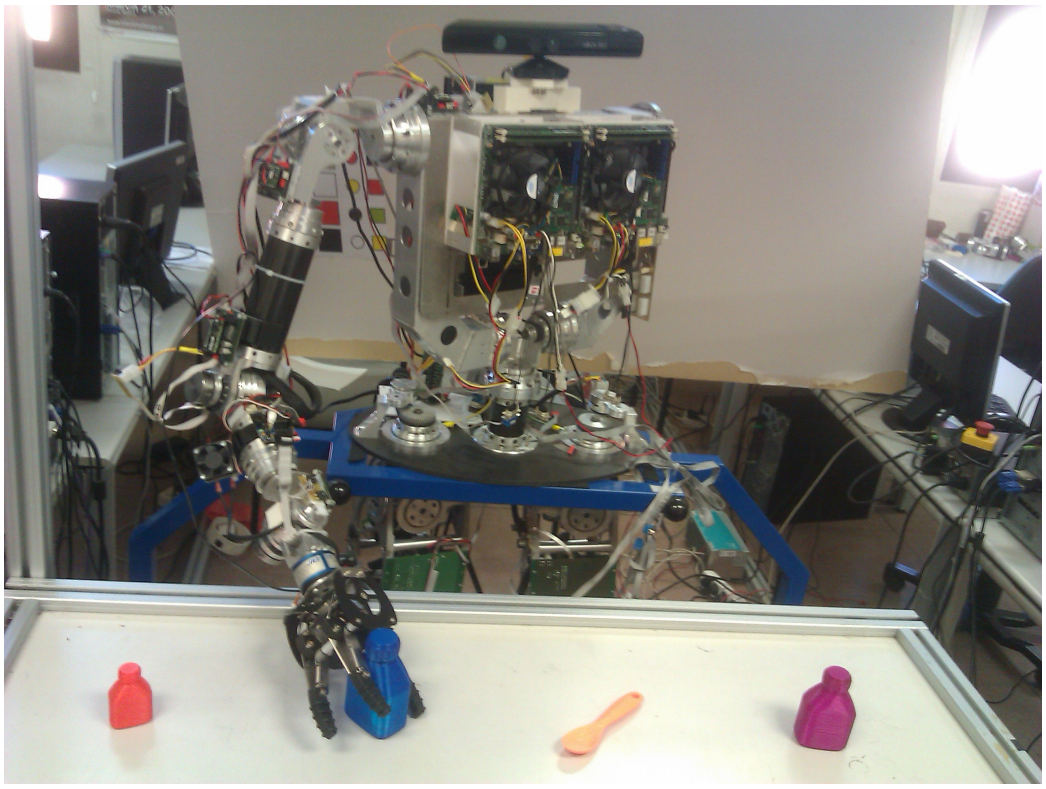

Figure 1.2: Teo is a full-sized humanoid robot that has been developed by the Robotics Lab research group of Universidad Carlos III de Madrid.

1. Imagination. While other works present images generated from hard-coded rules [16], or small gray-scale images (that do not represent shapes or objects) generated from similar images [5] as computational

creativity, this thesis aims at endowing robots with the capability of generating human-understandable mental images that have never been seen before, given human language descriptions.

2. Grounded language. The priority from the robot's perspective is to obtain knowledge that is "hooked" to reality. The aim is to learn the descriptive vocabulary of a user, so human language descriptions may be given in existing or even invented languages.

3. Spatial language. While other works hard-code forms such as "my left" or "your left" to set cameras on actors in a virtual environment [17], this thesis presents mechanisms to distinguish and understand egocentric and allocentric language directly from spoken interaction.

4. Execution. In contrast to the popular approach of robot imitation [14], which relies on encoding robot motor joint parameters alone, task execution in this thesis will be goal-oriented. Specifically, tasks will be encoded based on the continuous variation of the effects of each robot action on the environment.

5. Integration. All of these ingredients are combined to result in the design and implementation of the final presented system.

1.5 Document Structure

Though this work can be read in a sequential fashion, the chapters are mainly self-contained. Therefore, the interested reader may also decide to concentrate on single parts.

Chapter 2 introduces the background concepts and related works that are interesting to understand the topics of this thesis. This chapter is recommended for all readers, especially for those without a background in symbol grounding, mental imagery, computational creativity or goal-directed actions.

Chapter 3 presents the proposed architecture, providing an overall view of its three main blocks, and its internal components and connections.

Chapter 4 is dedicated to the proposed architecture's perception block and its components.

Chapter 5 explains the core algorithms of the proposed architecture's inference block and certain implementation specifics.

Chapter 6 deals with the strategies of the proposed architecture's execution block.

Chapter 7 presents the experiments, validations and final demostrations.

Chapter 8 depicts the conclusions and future lines of research.

Chapter 2

Background

The growing trends of robotics in domestic environments have been described in chapter 1. Three different shortcomings of classical robot programming were identified, and two different modern techniques for overcoming these shortcomings were introduced. This chapter aims to introduce a more extensive review on the scientific literature that is relevant to the theme topics of this thesis. We begin by briefly introducing artificial cognitive systems in section 2.1. We then introduce the symbol grounding problem in section 2.2 in response to shortcoming (A) identified in chapter 1. Shortcoming (B) identified in chapter 1 inspires the literature on mental models and mental imagery of section 2.3. Finally, shortcoming (C) will lead to the literature review on robot task execution of section 2.4.

2.1 Artificial Cognitive Systems

Artificial cognitive systems in general subscribe to one of two different paradigms of cognition [18]: the "cognitivist" approach based on symbolic information processing representational systems, and the "emergent" systems approach, embracing connectionist systems, dynamical systems, and enactive systems, all based to a lesser or greater extent on principles of self-organization.

According to F.J. Varela [19], the origin of the pure cognitivist vision comes from cybernetics (1943–1953), with the idea of an intelligence only based on logic. The catalog of cognitivist tools includes, among others, the ones from automated planning and knowledge and logic-based tools such as expert systems, case-based reasoning, or ontologies[1]. Alternative mechanisms also include co-occurrence statistics of word counts among documents, such as those used by Latent Semantic Analysis (LSA), capable of "learning words at the rate of school-aged children" [24], or the Hyperspace Analogue to Language (HAL), capable of "inferring semantic, grammatical, and abstract distinctions" [25]. Examples of originally cognitivist architectures include ACT-R [26] and Soar [27].

ACT-R is composed by several specialized modules. Each module processes a different kind of information: a vision module, a manual module for controlling hands, a declarative module for retrieving information, a goal module which changes its internal state when a task is accomplished. There is also a coordinator module, which manages all of the other modules. Reasoning is a cyclic process where patterns of information are identified, and new rules are produced. The perceptuo-motor system of ACT-R does not encode direct sensor information, but assumes that the vison module has translated the visual data into objects (symbols), centering only on attention and recognition.

Soar's behavior is a combination of rules produced by the system, in the form of IF-THEN states. To solve a problem, a search in the problem space (the set of states) is performed to cyclically move closer to a solution. Every cycle is composed by two phases: elaboration (knowledge recollection) and decision (choosing next action to be taken).

[1]An ontology is a knowledge representation system where the main elements are a categorization structure and a set of rules among the elements belonging to these categories, which allow to exploit parent-child relationships [20]. Popular ontology languages include CycL for its use in Cyc [21], and the semantic web's OWL [22] which is also currently used in the robotic cloud computing project RoboEarth [23].

2.2 The Symbol Grounding Problem

"A. There are no perceptual links or semantic relationships between the physical characteristics of objects and the words used to describe them."

This issue was first identified by S. Harnad in 1987 [28], and then formally defined by Harnad in 1990 as the "symbol grounding problem" [29]. The discussion in which he was involed was about the scope and limits of purely symbolic models of the mind, and about the proper role of connectionism in cognitive modeling. As an extension to Searle's Chinese room argument [30], Harnad believes it is impossible to learn a language (e.g. Chinese) with just a dictionary of that language (e.g. Chinese/Chinese). The trip through the dictionary would amount to a merry-go-round, passing endlessly from one meaningless symbol or symbol-string (the definiens) to another (the definiendum), never coming to a halt on what anything meant. Deb Roy refered to this as "circular definitions", and depicts an English version of this issue in Figure 2.1 [1].

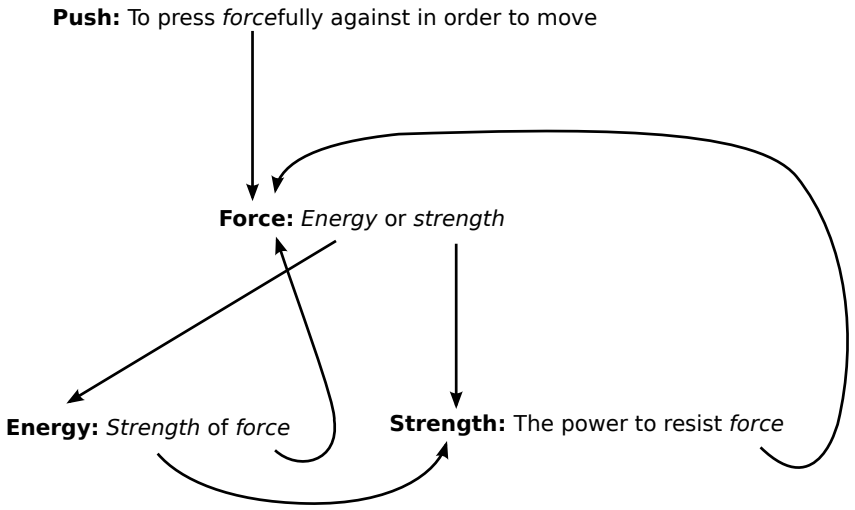

Figure 2.1: A network of definitions extracted from Webster's Dictionary containing circularities. To make use of such symbolic networks, non-linguistic knowledge is essential to ground basic terms of linguistic definitions [1].

The notion of linking words with physical objects, actions and abstract concepts is called "symbol grounding". It provides words with the "hooks" to

reality that enable understanding language, and even other words that can be involved in circular definitions. Symbol grounding has also been commonly referred to as language grounding [31][32], semantic grounding [33][34], the anchoring problem [35][36], or bridging the semantic gap [37][38]. Decades of research have provided different views ranging from psychology [39] to information retrieval for the semantic web [40] and, more recently, cognitive systems in robotics [41]. Note that symbol grounding is not an issue for all problem statements related with artificial intelligence. For instance, the Turing Test [42] can be passed by a purely symbolic artificial cognitive system if it is given the correct set of rules. However, in computer vision and robotics, linking words to perceptual input is essential for tasks as complexity increases and language is involved.

A number of visually grounded systems have been developed throughout the years. They have mainly focused on generating descriptions of synthetic and real scenarios, based on previously grounded information. One of the first works in linking grounded information to language was VIsual TRAnslator (VITRA) [43]. Dynamical situations are provided via video to VITRA, which in turn analyses and performs automatic generation of natural language descriptions for the movements it recognizes. Another approach [44] uses simple user-robot interaction and language games to conceptualize an object, though no further language grounding or inference possibilities are studied. DESCRIBER [2], see Figure 2.2, adds learning techniques to assign ranges of values of features to words. To achieve this task, the system is trained by manually transcribing human-made descriptions of computer-generated coloured rectangles. Then, every word is considered a potential label, and they are filtered to use only relevant ones. The system assigns a subset of features to each word. Then, an algorithm compares feature distributions between descriptions formed with these words. Finally, the system finds the subset of features for which the distributions are maximally divergent when the word is present and when it is not, and assigns these features

to the word. The results achieved by this system allow generating semantic descriptions of objects selected by a user on a screen, including spatial relationships with respect to other objects. The system's tests account for 10 non-overlaping rectangles. The features used are: red, green, blue, hight to width ratio, area, X and Y position of upper left corner, and ratio of maximum dimension to minimum dimension.

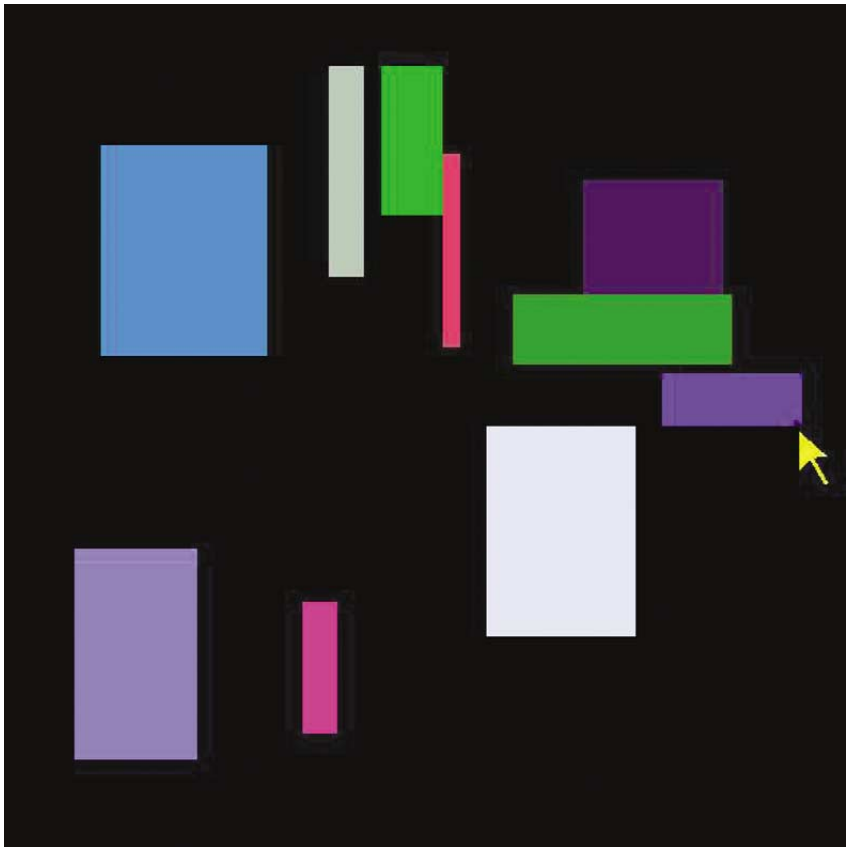

Figure 2.2: Given this task, DESCRIBER [2] could generate the description "the horizontal purple rectangle below the horizontal green rectangle".

DESCRIBER's internal mechanisms were later adapted to Newt, which provided real-world recognition capabilities based on end-user spoken descriptions [45]. These and similar contemporary visually grounded systems can be found throughout literature [46]. In the present, traditionally cognitivist architectures such as Soar are starting to develop language grounding extensions [47][48].

2.3 Mental Models and Mental Imagery

"B. There is a lack of inference capabilities for interpreting commands that include combinations of words that have never been taught together, but have been taught separately."

In the present, human inference is far from being well understood. On the other side, a general purpose computational inference motor that could emulate human inference does not exist either. Instead, a great variety of "narrow" artificial intelligence algorithms exist, many of which are used to provide inference within specific domains. The specific inference capability described in (B) has been rarely studied within robotic literature. However, literature from different fields focused on mental models, mental imagery, and creativity actually shed some light on this issue.

The term "mental model" was first proposed by K. Craik in 1943, to describe a "*small-scale model* of external reality" carried by an organism, additionally presenting some biological advantages [49]. P.N. Johnson-Laird more formally described a mental model as the stabilized representation of reality an agent perceives [50]. This representation is usually contrasted to the noisy and unstable nature of sensory systems. The notion of a mental model as an agent's stabilized representation of reality is further established by the "object persistence" concept studied in philosophy and psychology, which refers to the awareness of an object when it is not visible [51]. In reasoning in artificial intelligence, object persistence has been implemented through "epistemic fluents" (dynamic properties that express an agent's belief rather than actual sensation) [52]. Epistemic fluents have also inspired object persistence in video-game research [53]. In the field robotics, object persistence has been called "object permanence" by K.-Y. Hsiao et al., who developed a computational implementation of through a simulator [54]. In Hsiao's work, visually-detected objects are instanced as their virtual equivalents, and stored for a certain period of time even when they are not sensed.

In cognitive science lingo, the acquisition of a mental model through sensation is considered a bottom-up process. Sensory signals are received from the "bottom", and go "up" into mental constructs[2], where higher-level cognitive processes such as inference may occur. Following this taxonomy, the term "mental imagery" refers to a top-down process, where high-level mechanisms such as inference result in experience that resembles perceptual experience, but which occurs in the absence of the appropriate stimuli for the relevant perception [56]. In humans, a recent clinical psychology review situates generated mental images within a topographically organised area of the brain known as the visual buffer [57], which subscribes to the description given by S.M. Kosslyn in 1980 [58]. R.H. Logie's alternative model postulates that mental imagery occurs in a dedicated area called the visuo-spatial sketchpad, which is itself composed by a visual cache for shape and color and an inner scribe which stores spatial information and is related to movement [59]. Logie's model is based on A.D. Baddeley and G.J. Hitch's working memory model [60], see Figure 2.3.

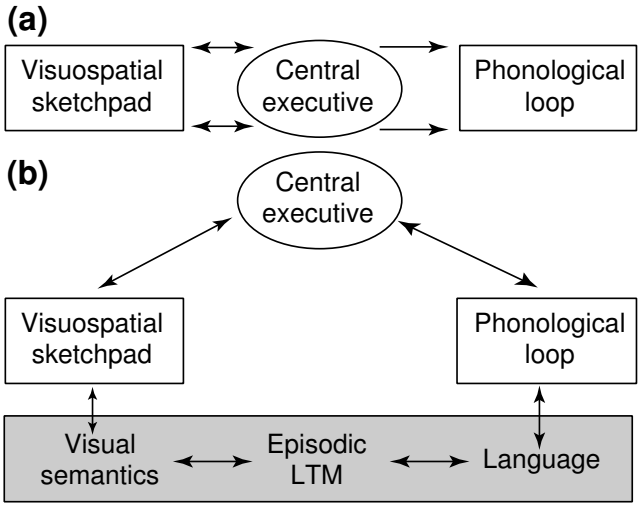

Figure 2.3: (a) The initial three-component model of working memory proposed by Baddeley and Hitch. (b) A further development of the working memory model [3].

[2]Note that alternative interpretations of perception, such as J.J. Gibson's "ecological approach" [55], may criticize the assumption of pre-existing mental constructs.

As a top-down process, mental imagery is capable of generating mental images that have not been previously perceived. The ability to come up with ideas or artefacts that are new, surprising and valuable is precisely the definition of "creativity" given by M. Boden [4]. Novelty is measured relative to the agent's previous knowledge (this is called psychological creativity, or simply P-creativity by Boden) rather than relative to the complete history of science and innovation (historical creativity, or simply H-creativity). Boden further groups creativity in the following three categories.

- Combinational creativity. This first type of creativity involves making unfamiliar combinations of familiar ideas. Mental imagery that generates mental models from commands that include combinations of words that have never been taught together, but have been taught separately, would be a complex manifestation of combinational creativity.

- Exploratory creativity. This second type of creativity can come from adding a new trick to a repertoire within a specific domain or conceptual space (in a real sense it's something that fits in an established model, but without previous awareness of its existence). Boden states that this is the type of creativity that is most performed by researchers and artists, who explore within the rules established by a state of art, and produce novel contributions that generate only a marginal degree of surprise.

- Transformational creativity. In contrast to exploratory creativity in which a specific domain or conceptual space is explored, transformation involves knowing the rules, but moving outside of them. Examples are Picasso's Cubism (where a disruptive surprise comes from introducing a diversity of perspectives within the same picture) or Kekulé's account on insight about benzene molecules (imagining molecules as snakes chasing their tails rather than the classic view of open chains of atoms).

Theory and implementation of a working computational creativity is, however, sparse throughout literature. An ungrounded example (and therefore, of relatively small value for robotics) of combinational creativity can be found in the JAPE system, that generates jokes of a general type familiar to every eight-year-old [61]. JAPE can generate puns such as "What's the difference between money and a bottom? One you spare and bank, the other you bare and spank." using a dictionary and a hard-coded model based on the pun "What do you get when you cross a sheep and a kangaroo? A woolly jumper.". A very different example of computational creativity (not necessarily combinational creativity) is AARON, which "automatically generates novel line drawings of different styles" [16]. AARON's image generation is based on large sets of IF–THEN rules, that must be manually hard-coded for each style and drawing topic, but result in surprising images such as those of Figure 2.4.

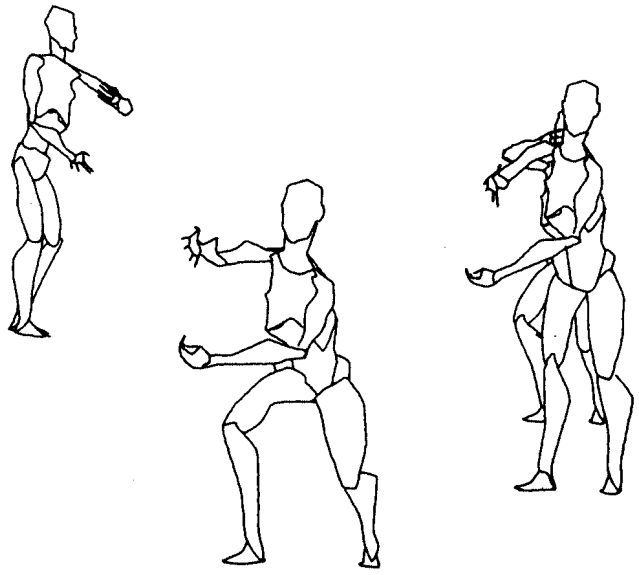

Figure 2.4: AARON generated these antropomorphic images from IF–THEN rules and planning algorithms in 1989 [4].

A prominent example of combinational creativity (equivalent, at least in intention, to a mental imagery process to create unfamiliar images based on familiar images) is the "AHA! experience" developed by P. Thagard et al.

in 2011 [5]. Thagard's connectivist approach for combination is based on artificial neural networks to achieve convolution. Figure 2.5 depicts result of the convolution of two familiar images (far left images) to generate a novel image (far right image). Furthermore, Thagard additionally evaluated the surprise generated within a population of human judges.

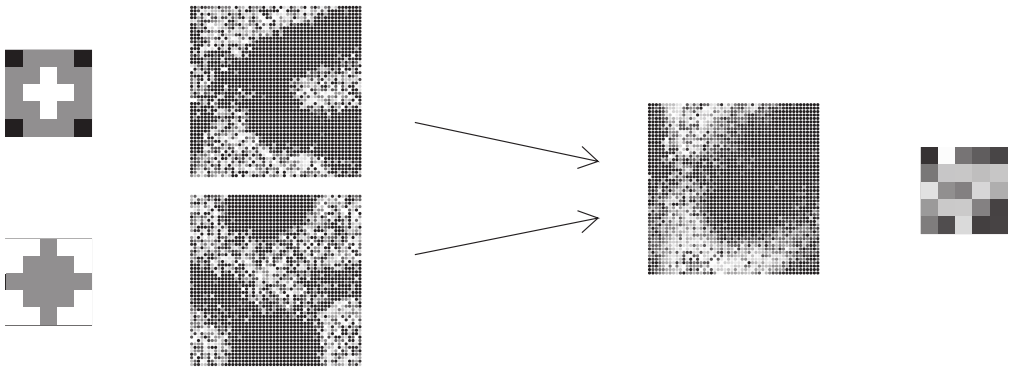

Figure 2.5: Thagard's (2011) "AHA! experience" connectivist approach combines images through convolution achieved with artificial neural networks [5].

Moving away from the field of computational creativity, a number of systems that can somehow perform mental imagery can also be observed. For instance, a work from the field of multi-modal inference has provided tools to discover unknown properties of an object from limited views of it [62]. Although it is highly focused on multi-modal categorization, it provides direct cross-feature mappings that allow it to infer certain missing features, such as auditory information when only visual information is available.

In existing robotics computational models, however, mental imagery is usually limited to the capability of "recalling" previous sensations. However, the work of N. Mavridis and D. Roy in 2006 on Grounded Situated Models is of great interest [63]. Inspired by Probabilistic Occupancy Maps (a video-game research model of object persistence [64]), Mavridis incorporates the possibility to "imagine" the position of an object on a 2-dimensional plane. This "imagination" command in turn increment's the robot agent's belief on the object occupying a given square cell.

2.4 Robot Task Execution

"C. There are no bindings between actions and their effects. If there are changes in the environment, the robot's program must be modified in order to achieve the same effect."

In classical robot programming, users program sequences of robotic movements. A common method is to move the robot's end-effector to each point of interest and record it, and then program a sequence of instructions which additionally include velocity, precision and type of movement. The robot's movement is teleoperated using buttons, or sometimes a joystick (e.g. a 3 degrees of freedom joystick that can switch behavior through menus). The coordinates of points of interest can also be directly hard-coded, which requires a full knowledge of the environment and absolute certainty that there will be no external perturbations.

Techniques such as robot imitation, also known as learning by demonstration (LfD) or programming by demonstration (PbD) [13], aim to overcome the need for teleoperation and manual hard-coding of every robot behavior [14]. These techniques simplify programming robot actions. The way these methods generalize an action is by recording the kinematics of a demonstrator when performing the action, and then applying different machine learning algorithms. The demonstrator can either be the guided robot itself, a human with sensors attached, or video sequences with human movements.

In [65], a human demonstrator performs a task several times (e.g. hitting a ball) using a robotic arm. Positions, orientations and velocities of the arm are recorded, and the number of representative states of the action are estimated with Hidden Markov Models (HMM). HMM are used to handle spatio-temporal variabilities of trajectories across several demonstrations. Finally, and in order for the robot to execute the trajectory, Gaussian Mixture Regression (GMR) is used to create a regression function using previous HMM states. This reconstructed trajectory is the one the robot reproduces

to imitate the human movement. Another common technique used, along with HMM [66][67], is Gaussian Mixture Models (GMM) as in [68][69].

Recognizing an action through external measurements is called direct action recognition. In [70], they perform a neuro-fuzzy classification of optical flow features between consecutive frames of human movement in video sequences. Neuro-fuzzy is a combination of fuzzy logic with neural networks, using the classified output of a fuzzy system as an input to the neural network. In [71], they track and filter human hand and feet trajectories through Principal Component Analysis (PCA). First, they record trajectories of key points from a video. Then, they split them into sub-units called basic motions. Next, they extract some features of the basic motions, and project these feature vectors into a reduced space generated by PCA, resulting in the formation of clusters of similar actions. For recognition purposes, they record an action, transform it with the same process explained, project its vector onto the reduced space, and finally, associate it with the closest cluster.

As shown, the focus in these types of research is on learning the kinematics of actions. By using only kinematics, actions are limited to be executed exactly as taught. Any disturbance, e.g. an obstacle along the trajectory or a displacement of the target object, would make task completion impossible. They do not solve, in general, the issue of adapting a robot's behaviour depending on changes in the environment. Tasks are usually encoded as trajectories in the robot motor joint space, so expected effects on the environment are not stored. An approach used by some authors is using Dynamic Movement Primitives (DMP) [72][73]. DMP encode tasks as superposed motor primitives that are control policies to reach a certain Cartesian space goal. As control policies, motor primitives are robust to perturbations on a robotic arm, and limit cycle movements may also be superposed, achieving movements such as drumming or tennis swings [73]. However, the actual effects in the environment are not asserted, as goals are encoded as Cartesian space targets and trajectories are encoded in the robot motor joint space.

Until now we have seen that there is a lack of codification of action effects in classical robot programming and several examples of modern approaches (namely robot imitation and dynamic motor primitives), and only the kinematic aspects of a robot's movements are considered. Roughly speaking, we can say that sequences of movements are blindly performed by robots during task execution. Robots can, effectively, repeat the same movements, but the consequences remain a mystery for them [74]. However, humans can understand the effects an action can have on an object. What is more, scientific literature indicates that the human brain encodes actions as end-goals. A psychological experiment shows how when children imitate others grasping a person's ear, they tend to imitate the action goal (which ear to grasp) rather than the kinematic aspects of the action (which hand is used to grasp) [75].

Neuroscience has discovered evidence supporting goal encoding of actions, especially with the study of neurons endowed with mirror properties ("mirror neurons") [76]. Mirror neurons fire when an action is performed, and also upon observation of the same action performed by another subject. When macaque monkeys were trained to grasp food with a tool, their mirror neurons fired when they performed the action, and also fired when they observed the same action performed by an experimenter, even when using a different tool [77]. This too demonstrates the relevance of the goal as opposed to motor trajectories. The F5 cortical area of macaque monkeys has been found to contain mirror neurons. Research has found evidence of an analogous area of mirror neurons in human inferior frontal gyrus [78].

When talking about goal-directed actions in robotics, a goal encoding is found in [79] where, despite they learn the kinematic trajectory to perform an action, they also encode some goals to be achieved. They recreate a version of the previously cited psychological experiment with children [75] with colored dots on a table, which are touched by a human with both arms in alternation. When the dots stay on the table, children tend to imitate the goal (which dot to touch), and not the arm used to do it. In the recreated

experiment, the demonstrator repeats the same task and, while observing the demonstration, the robot tries to extract a set of constraints for the task. Later, the robot computes the trajectory that best satisfies the constraints and generates a motion.

In a recent co-authored work [80], two different types of goal-directed actions were identified.

- Goal only: When the information used is only the difference between the initial and the final state of the element. A different co-authored work studied this type of goal-directed actions using several different machine learning algorithms [81].

- Continuous tracking: When the whole process of change is taken into account. This may be achieved with a continuous trajectory in a high-dimensional space.

Notions of continuous tracking can be found in [82], which uses a combination of object spatial and demonstrator-hand movement tracking. They build a system with a set of primitive actions (inverse models). When the human demonstrator performs an action, they continuously track the object and the hand spatially through time. At the same time, they run all inverse models during action stages to find the best performance of each model in each stage. Finally, they construct a high-level inverse model composed by those selected primitives, being able to imitate the action goal with similar spatial movements. Note that the parameters to be imitated are not robot motor joint positions, the only target is the hand position.

In the previously mentioned recent co-authored work [80], inspired by neuronal behavior, we presented the Continuous Goal-Directed Actions[3] (CGDA) infrastructure and defined CGDA as actions in which the parameters analyzed are the ones belonging to the object (or more generally, elements)

[3]The term *Continuous Goal-Directed Actions* was coined by the first author Santiago Morante of this contribution to the ICRA 2013 major international robotics conference.

affected by actions continuously in time. By using CGDA, an action can be encoded in a complementary way, by learning, not only by how it is performed (kinematic parameters), but also the effects of the action. This "double learning" may allow the robot to complete the action, even when the scenario changes or when elements block its usual path of execution, by generating alternative motions which accomplish the encoded goals. Generalizing, recognizing, and executing CGDA is however complex and in ongoing research.

Chapter 3

Robot Imagination System

This thesis presents a novel system, called "Robot Imagination System" (RIS), that aims at fulfilling the proposed main objectives. Figure 3.1 depicts its basic architecture. The RIS's components are grouped in three blocks: A. Perception, B. Inference, C. Execution.

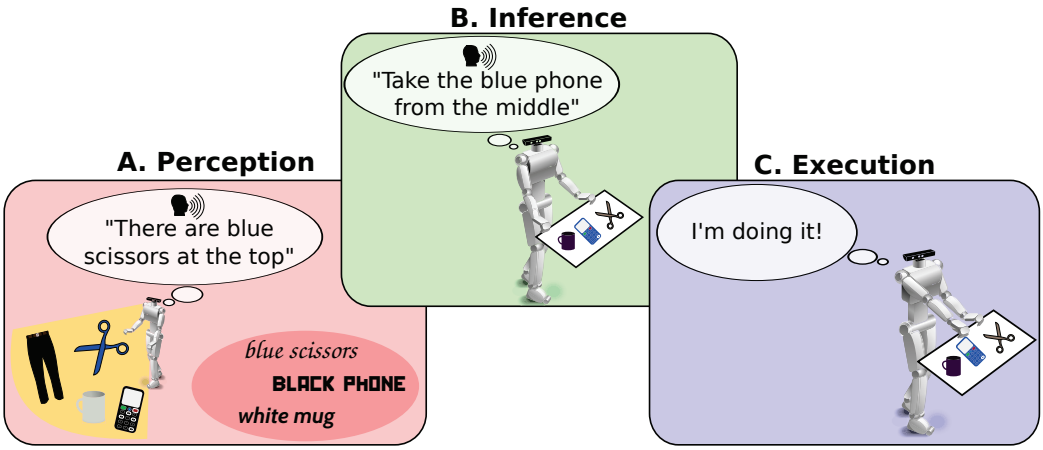

Figure 3.1: The Robot Imagination System's three blocks of components.

The term *robot imagination* has been incorporated in the system's name to depict its strong focus on creative inference. The term was actually first coined by the author in the recent IROS 2013 major international robotics conference [83] to denote a robot's capacity to "imagine" or generate mental models through mental imagery given combinations of words that have never been taught together, but have been taught separately.

3.1 The RIS as a Computational Model

The proposed architecture can also be seen as a computational model for emulating certain human abilities, such as imagination. Two important premises should be taken into account regarding this interpretation.

- If we classify computational models as either scientific or technological models (similar to how Thill et al. classify affordance models [84]), the RIS falls into the second category. It aims to allow more efficient intelligent machines and robots. It is inspired by experimental evidence, but is not strongly constrained by it.

- As such, despite theories of direct perception [85] and enaction [86] that tend to claim that the border between perception, inference and execution is diffuse or inexistent, the architecture accommodates to a Classical British empiricist model [87], where boundaries between components are clearly defined. This serves for organizational purposes and also promotes the reuse of existing components within the architecture.

3.2 Description of Blocks

The RIS's components are grouped in three blocks. Each of these blocks is focused on overcoming one of the shortcomings of classical robot programming presented in section 1.1: A. Perception, B. Inference, C. Execution. The following is a brief description of each one of these blocks that compose the system.

A. **Perception:** The input block of the system. This block includes visual and auditory systems, along with a language grounding component.

B. **Inference:** The core imagination block of the system. Recognition capabilities will also be enabled through this component.

C. **Execution:** The output block of the system. This block includes visual and auditory output, along with the core execution component.

3.3 Component Breakdown

The RIS blocks may be depicted as groups of individual interconnected components. Figure 3.2 presents this breakdown into components, and the interconnections between these components. The components that are depicted in light red belong to the perception block, and are the following: Computer Vision, Grounding Core, and Speech Recognition. The component that is depicted in light green belongs to the inference block, and is the Imagination Core, which provides imagination and recognition capabilities. The components that are depicted in light blue belong to the execution block, and are the following: Visual Output, Execution Core, and Speech Output.

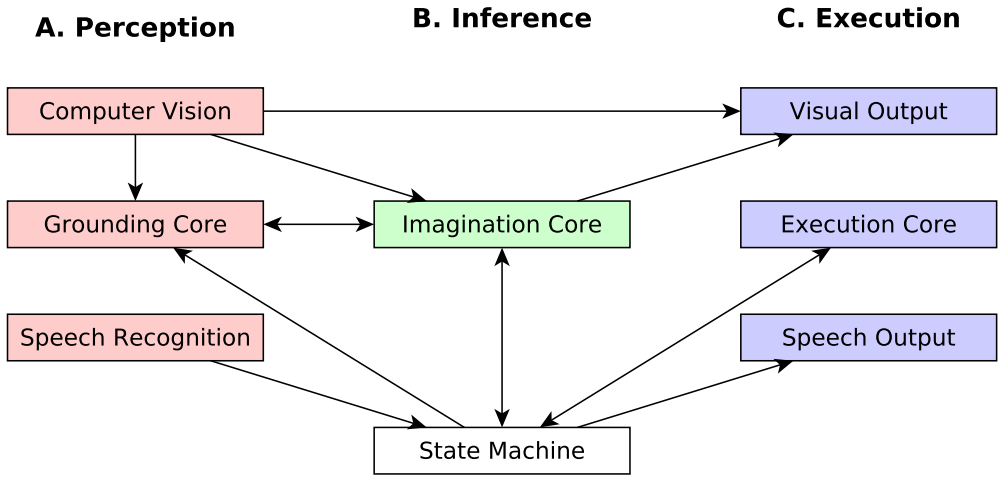

Figure 3.2: The RIS component breakdown and interconnections.

An independent component, the State Machine (located at the bottom of Figure 3.2 with a white background), manages the system as a whole and also enables spoken interaction with the end-user. While a number of mechanisms could perform this management task, a Finite-State Machine

(FSM) [88] is fit for this purpose. Figure 3.3 depicts an FSM state diagram that implements a basic functional system management.

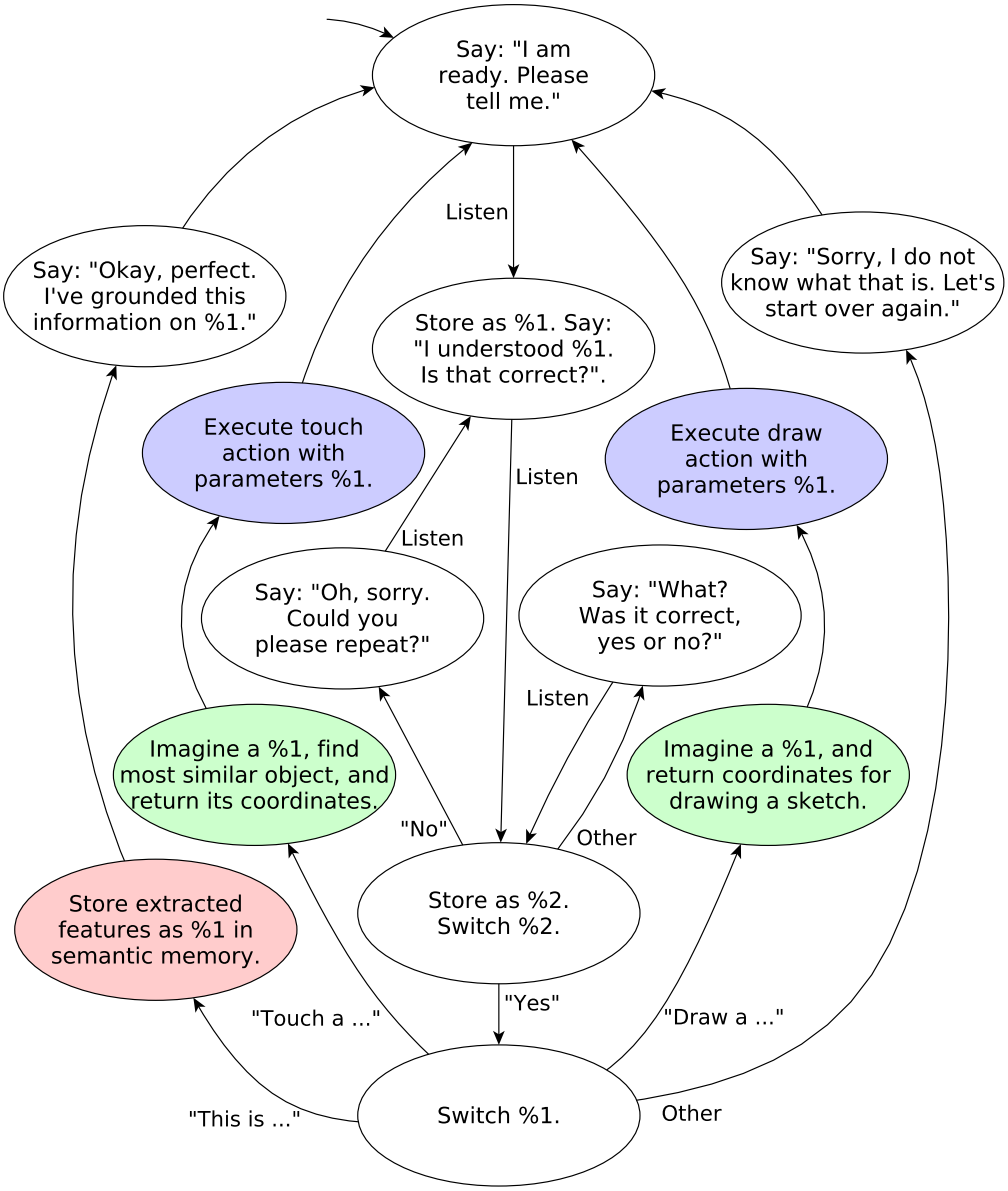

Figure 3.3: Finite-State Machine example state diagram.

With this example FSM state diagram implementation, the components of the RIS collaborate according to the following scheme.

- The Computer Vision component receives a continuous stream of images from a 2D or 3D camera. It segments objects from each frame,

and then extracts a fixed set of features (height, area, color, etc.) from each object. It sends an output stream of this extracted data upon arrival and processing.

- The Speech Recognition component waits for user input. Once words have been heard, it repeats these words to the user and requests user confirmation.

- If the confirmed words match the keywords "this is", the State Machine forwards the rest of the words to the Grounding Core with a "push" command. The Grounding Core is synchronized with the Computer Vision component, and merges these words with the instantaneous Computer Vision output.

- The Imagination Core always has access to the Grounding Core to perform inference, and may also access the Computer Vision component to discover the Cartesian coordinates of a described object in a certain instant.

- If the confirmed words match the keywords "touch a", the State Machine forwards the rest of the words to the Inference Core, and queries it for the coordinates of the object to touch. The State Machine then forwards these coordinates to the Execution Core with a "touch" command.

- If the confirmed words match the keywords "imagine a", the State Machine forwards the rest of the words to the Inference Core, and queries it for the coordinates to draw a sketch of the object it has imagined. The State Machine then forwards these coordinates to the Execution Core with a "draw" command.

- If the confirmed words do not match any known keywords, the State Machine returns to its initial state.

3.4 Chapter Summary

In this chapter we have described the Robot Imagination System (RIS), a novel system proposed for overcoming shortcomings of classical robot programming and also modern techniques. We have seen its interpretation as a computational model under specific premises, and presented high-level descriptions of component behaviors and interconnections. Additionally, we have seen how the system as a whole is much more focused on language and the objects of action than on pure robot motor joint positions or end-effector coordinates as seen in classical schemes. This agrees with the proposed main objectives of this thesis.

In the next three chapters, we intend to give full insight on theoretical, algorithmic and implementation issues of the described components of each block, guiding the reader to understand the integrated robotic system.

Chapter 4

Perception Block

Human perception is the complex process that begins with an external ("distal", or far) stimulus, and ends with a final "percept" or mental representation [89]. The receptors of external stimuli may be rod or cone cells of a retina (for visual stimuli), or the inner ear's components (for auditory stimuli). Different theories of psychology [90] and neuroscience [91] debate upon the nature of the final "percept" within human perception. Despite this debate, Figure 4.1 depicts a generally accepted model of human perception.

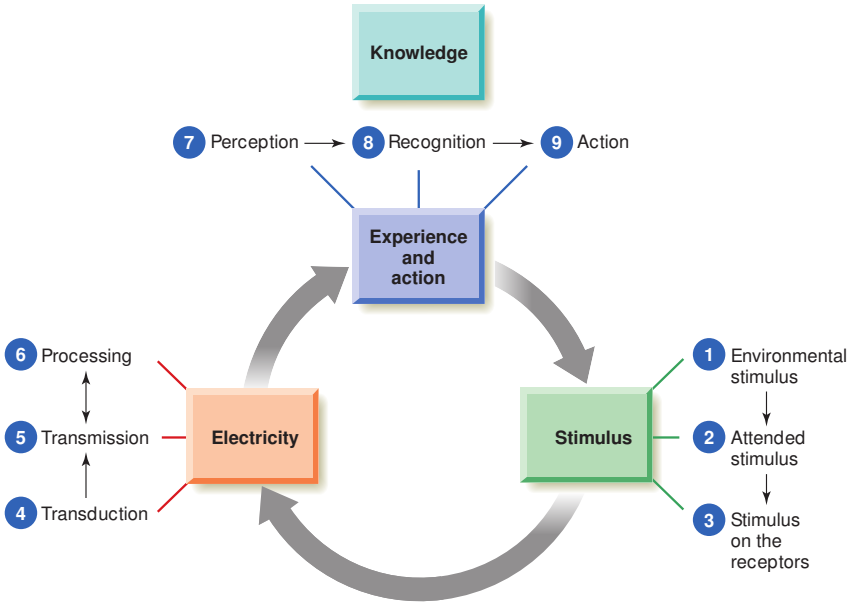

Figure 4.1: The steps in the perception process are arranged in a circle to emphasize that the process is dynamic and continually changing [6].

As illustrated in Figure 4.1, we can speak about Perception as a complete, dynamic and continually changing cycle that includes recognition and action, or also as a subprocess within that same cycle. An interesting appearance is that of "knowledge" as a subproduct of Perception as a whole. In RIS, sensation (which corresponds to Electricity in the previous diagram, and is composed by transduction, transmission and processing) currently includes vision and audition. Section 4.1 of this chapter will be dedicated to the system's Computer Vision component. Existing robotic components will be used for Speech Recognition. Knowledge plays a fundamental role in RIS, and is implemented within the Grounding Core component, which will be seen in section 4.2 of this chapter.

4.1 Computer Vision

Computer vision started out in the early 1970s, and was viewed as the visual perception component of an intelligent robotic system that would be relatively simple to achieve [92]. In fact, early assumptions supposed that the computer vision problem could be solved within the scope of months [93]. However, decades in, artificial vision is a highly active field of research. A great portion of academic institutions and industrial research centers have dedicated computer vision groups.

Our interest is in segmenting objects from each camera frame, and then extracting features from these objects. These extracted features can then be stored by the Grounding Core, and also used by the Imagination Core. Due to the scale and complexity of computer vision, only minor contributions to its state of the art as a field have been performed in this thesis. A review of the techniques that have been used to perform object segmentation in 2D and 3D camera images will be given in subsections 4.1.1 and 4.1.2 respectively. Concepts and mathematical expressions corresponding to the extraction of features from segmented objects will be seen in subsection 4.1.3.

4.1.1 Object Segmentation in 2D Camera Images

In line with conventional approaches, our first objective is to distinguish between foreground and background, because objects are potentially part of the foreground. Our intention is to create a binary mask, where all foreground pixels are activated and background pixels are null, to ultimately detect object contours. Either of the following different methods may be applied given an RGB color image.

1. Directly in the RGB color space. If the color of the objects to segment or background is known, we can operate directly in the RGB color space, avoiding the computational implications of edge detection or converting images to other color spaces. In [94], the author of this thesis proposed a simple method where a layer of low relevance (e.g. G or B to find red) is subtracted from the layer of highest relevance to find a color (e.g. R to find red). This subtraction reduces the effects of light variation. The image may then be transformed into a binary mask by applying a band pass filter to obtain the objects, or a band reject filter to remove the background.

2. Transformation to the HSV color space. While transformation of RGB images to the HSV color space may be computationally expensive for slow machines, this is less of an issue for modern desktop hardware. If computationally viable, and the color of the objects or background to segment is known, the following method may be applied. Knowing the color of the objects to segment, the image can be transformed to the HSV color space, and a band pass filter can be applied to the hue (H) layer. If instead the color of the background to segment is known, a band reject filter can be applied to the hue (H) layer.

3. Edge detectors. Working with objects and background of unknown and diverse colors requires more sophisticated solutions, such as the use of

edge detectors for arbitrary edge profiles. A well-known edge detector that fits this description is the Canny edge detector [95]. The masks obtained with edge detectors contain "hollow" objects (not all of the internal pixels of the objects are activated), but these masks are equally useful to determine the contours of the objects.

The binary masks that result from the previous methods are usually imperfect for obtaining object contours. A process of morphological closing (an erosion followed by a dilation) is used to remove noise. Since morphological operations act on a pixel-level, the number of pixels is set as a percentage of the original image's width and height. The curves joining all the continuously activated pixels (along the boundary) are the detected object contours. Figure 4.2 depicts binary images generated using method (1) layer B minus layer G, that was used to detect a blue marker to teach spatial language. The contours of the object are highlighted in the figure (pink), and rotated bounding boxes surround the detected objects (dark blue).

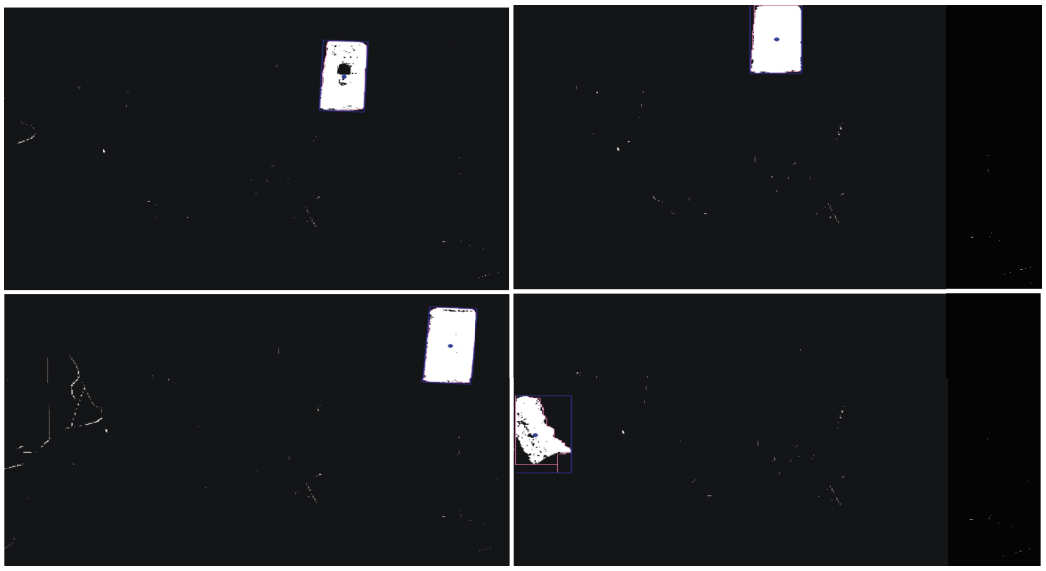

Figure 4.2: Object segmentation performed on a stream of 2D camera images.

4.1.2 Object Segmentation in 3D Camera Images

Stereopsis, or the impression of depth that is originated from binocular vision, is a high-level cognitive process in human beings. The issue of stereoscopic vision has been computationally treated through the complex study of disparity among paired images. The development of low-cost electronic sensors that directly measure depth has created a renewed interest in 3D image processing. These sensors use technologies such as structured light and time-of-flight measurements.

Following the same premises as for 2D object segmentation, our first objective is to distinguish between foreground and background, because objects are potentially part of the foreground. 3D images that arrive from low-cost electronic depth sensors tend to be extremely noisy. An early visual processing stage must remove bogus data, which computationally correspond to 3D pixels with "NaN" or "not-a-number" values.

These 3D images, or "point clouds" (a name that has become popular upon the arrival of these low-cost sensors) are still noisy and computationally difficult to handle. To filter and reduce the computational load of a point cloud, a technique called "voxelization" is used. A voxel or volume-pixel is a small-sized 3D cube in space. The voxelization technique creates a 3D voxel grid over the input point cloud data. Then, in each voxel, all the points present are approximated (and thus, downsampled) with their centroid. This approach is slightly slower than approximating them with the center of the voxel, but it represents the underlying surface more accurately[1].

An extremely popular assumption in 3D computer vision is to consider that the background is integrally composed by planes (floor, tables, shelves). The Random Sample Consensus (RANSAC) paradigm allows fitting point clouds to geometrical models [96]. The common practice followed is to use the

[1]This description is based on the description of a computer algorithm that performs voxelization, available at http://docs.pointclouds.org/trunk/a01645.html#details last accessed May 29, 2014.

RANSAC paradigm to discover planes among the point cloud. The 3D pixels that correspond to these planes are believed to be part of the background, and ultimately removed.

The resulting point cloud will be composed by disjointed clusters of 3D pixels in space. These 3D pixel disjointed clusters may be gathered by using a process called Euclidean Clustering, that consists in grouping given an Euclidean distance threshold. This Euclidean distance threshold can be given as a percentage of the known resolution of the electronic depth sensor. For aesthetic detail and efficient collision computation, a triangle mesh may be obtained based on projections of the local neighborhoods using a greedy surface triangulation algorithm [97], or a Poisson surface reconstruction algorithm [98]. Figure 4.3 depicts resulting object triangle meshes (blue), where the object of the center of the robot's gaze is highlighted (red).

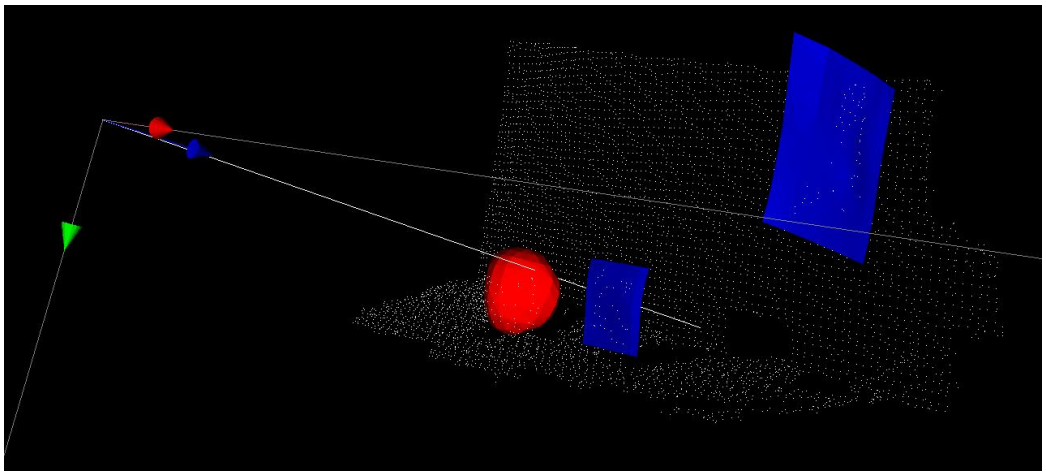

Figure 4.3: Object segmentation performed on a stream of 3D camera images.

4.1.3 Feature Extraction

The following features are extracted and streamed from each flat contour (2D) or triangle mesh (3D), and may be used at convenience. These definitions will correspond to the flat contour case for simplicity, but analogous definitions exist for the 3D case.

- Centroid. The centroid is obtained from the image moments of the contour. The centroid coordinates are computed using the expressions: $C_x = \frac{M_{10}}{M_{00}}$ and $C_y = \frac{M_{01}}{M_{00}}$. These expressions result in pixel values. Even though the RIS can work directly in the pixel space, this point can be expressed in metric values given the depth C_z (e.g. obtained by direct depth sensor measure corresponding to this pixel) and the camera's intrinsic parameters, resulting in ${}^{CCD}P = (P_x, P_x, P_z, 1)^T$. This point can further be defined with respect to the robot's origin using the homogeneous transformation matrix relation ${}^{O}P = {}^{O}H_{CCD} \cdot {}^{CCD}P$.

- Area. The number of pixels contained within a contour, which is also given by the image moment M_{00}.

- Rotation. The orientation of a minimum bounding box corresponding to the contour.

- Maximum Axis. The measurement, in pixels, of the largest side of the previously obtained minimum bounding box.

- Minimum Axis. The measurement, in pixels, of the shortest side of the previously obtained minimum bounding box.

- Aspect Ratio. The Aspect Ratio is calculated as the ratio between the Minimum Axis and the Maximum Axis.

- Rectangularity. Rectangularity is calculated as the ratio between the area of the contour and the area of the minimum bounding box.

- Solidity. Solidity is calculated as the ratio between the area of the contour and the area of its corresponding convex hull.

- Arc. The arc of the contour corresponds to the length, in pixels, of the perimeter of the object.

- Radius. The radius, in pixels, of the minimum bounding circle corresponding to the contour.

- RGB. Average and standard deviation of red, green, and blue directly obtained from the original RGB pixels within the original contour.

- HSV. Average and standard deviation of hue, saturation, and value obtained from the transformation of the original RGB pixels within the original contour.

Note that while scale-invariant features such as SIFT [99] or SURF [100] descriptors have obtained popular use within the computer vision community, the initial aim of feature extraction in RIS has been to obtain human-readable and human-understandable information.

4.2 Grounding Core

The global architecture of RIS has been seen in chapter 3 of this thesis. A diagram and description of the flow of data between components was included in section 3.3. Following this description, the Grounding Core receives a continuous flow of features extracted from the Computer Vision component, which are stored in a database upon the arrival of descriptive words from the State Machine with a "push" command. The Grounding Core is actually a flexible infrastructure that can work with sensory information of diverse nature (visual, haptic, or even auditory) and origin (e.g. real or synthetic). The Grounding application will be formally defined in subsection 4.2.1. The Grounding Core stores data in the Feature Space, which will be introduced in subsection 4.2.2. The Semantic Point Cloud will be presented and formally defined in subsection 4.2.3. Finally, how this Semantic Point Cloud is populated will be seen subsection 4.2.4.

4.2.1 The Grounding Application

Let E be the space of External stimuli from the robot's perspective, and W be the space of Words of a given society (which plays a crucial role in language acquisition [101]), the process of grounded semantic knowledge acquisition in Perception is the Grounding application G, which can be seen in Eq. 4.1.

$$G : (E \times W) \rightarrow K \tag{4.1}$$

Where K is Knowledge. Knowledge in RIS is stored in the Semantic Point Cloud S, which will be seen within the following subsections.

4.2.2 The Feature Space

The Feature Space is one of the semantic storage spaces of the Grounding Core component. Formally, let the Feature Space be $F \epsilon \mathbb{R}^n$, where n corresponds to the dimensionality of F and also to the maximum quantity of scalar features that arrive from the sensory components. For instance, n automatically reaches 22 dimensions when connecting the Computer Vision component and activating all of its feature extraction possibilities for 2D camera images (except for the centroid's depth and related conversions). This is because 22 is the sum of individual extracted features taking into account that the 2D centroid is two-dimensional, that average red, green and blue each generate different dimensions, and so on. Before any grounded semantic process is performed, the dimensionality n of F is null.

4.2.3 The Semantic Point Cloud

The Semantic Point Cloud is a finite set of points, where each point is associated with a position in the Feature Space and also with a discrete label that represents a specific descriptive word. This is similar, but generalized to $\mathbb{R}^n$, to the notion of "labeled point cloud" in $\mathbb{R}^3$ Cartesian space recently

defined by Fober et al. for visual classification in the field medical imagery [102]. Formally, the Semantic Point Cloud S is a set of points $\{s_1, s_2, ..., s_p\}$ with two associated functions, Eq. 4.2 and Eq. 4.3.

$$R_F : S \rightarrow F \epsilon \mathbb{R}^n \tag{4.2}$$

Where the application R_F denotes a feature retrieval function that corresponds to retrieving the labeled point's associated features or coordinates in F (the sensory data of an object that has been described to the robot).

$$R_D : S \rightarrow D \tag{4.3}$$

Where D is the semantic storage space for the robot's acquired dictionary, and the application R_D denotes a word or symbol retrieval function that corresponds to retrieving the labeled point's name (the associated descriptive word) from this dictionary D.

In terms of labeled data points in the Semantic Point Cloud, the point cloud of n-dimensional labeled points that share the same label constitute a Semantic Subspace. Let d be words which are actually the maximum set of *non-repeated* words that have been registered in the dictionary D. The number of words d will always be less or equal to p (the number of labeled points in S), and corresponds to the number of Semantic Subspaces.

4.2.4 Populating the Semantic Point Cloud

The Semantic Point Cloud is initially empty. The Grounding Core generates and stores new n-dimensional labeled points given $< \boldsymbol{w}, \boldsymbol{f} >$ incoming pairs and a "push" command, where $\boldsymbol{w}$ is a vector of descriptive words, and $\boldsymbol{f}$ is a vector of scalar features extracted from sensory data upon description. Given one $< \boldsymbol{w}, \boldsymbol{f} >$ incoming pair, let m be the number of descriptive words contained in $\boldsymbol{w}$, then m new labeled points populate the Feature Space (F). Each word w_i becomes the label in the dictionary (D) of one the newly

generated points of the Semantic Point Cloud (S). For the given $< \boldsymbol{w}, \boldsymbol{f} >$ incoming pair, the newly generated points all share the same coordinates, which correspond to the direct mapping without permutation of each scalar feature f_j on F. The dimension n of $F \epsilon \mathbb{R}^n$ is increased if the number of scalar features contained in $\boldsymbol{f}$ is greater than its current value. This functionality of dynamically growing is useful to adjust F to the number of scalar features contained in $\boldsymbol{f}$ of the first incoming pair. However, on-line dynamic growth should be used with care, as the definition of old points becomes incomplete in $F \epsilon \mathbb{R}^n$ when n grows, and algorithms should take this into account.

Figure 4.4 depicts an example where a $< \{green, square\}, \{0.97, 120\} >$ first incoming pair arrives with a "push" command. The vector of descriptive words $\boldsymbol{w}$ contains the two words "green" and "square", so $m = 2$ and therefore two new points populate the Semantic Point Cloud ($p = 2$). The vector of scalar features extracted from sensory data $\boldsymbol{f}$ contains the two scalars "0.97" and "120", so the dimension n of F grows to $n = 2$ if it was less than 2 (which is the case, as it is a first arriving pair). The word "green" becomes the label of the first point created in S and thus the first entry in the dictionary D, and the "square" becomes the label of the second point created and thus the second entry in D. The coordinates in F of the two new points are *identical* and precisely $\{0.97, 120\}$ in F, which matches with $\boldsymbol{f}$, as the two points originate from the same sensory input.

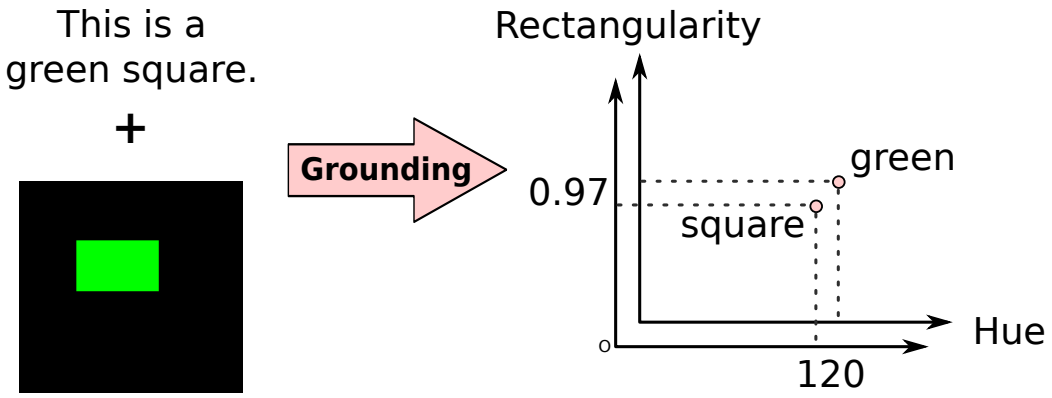

Figure 4.4: Populating the Semantic Point Cloud with labeled points.

It is interesting to highlight that although Figure 4.4 depicts that "0.97" corresponds to "rectangularity", and "120" corresponds to "hue", the Semantic Point Cloud does not know about "rectangularity" or "hue", nor does it need to not know about these concepts. This agnosticism provides flexibility with respect to the nature and origin of sensory data.

4.3 Chapter Summary

In this chapter, focus has been given to the Computer Vision component and the Grounding Core component. Each of these components play a fundamental role in the Perception block. Within Computer Vision, methods for object segmentation in 2D and 3D camera images have been described, as well as concepts and mathematical expressions for extracting human-understandable features from segmented objects. The Grounding Core has lead to the definitions of the Feature Space, the Semantic Point Cloud, and the process of grounded semantic knowledge acquisition in Perception. In RIS, the Grounding Core enables embodied semantic knowledge and thus cognitive processes that are grounded in perception.

Chapter 5

Inference Block

Inference is studied within philosophy, cognitive and developmental fields of psychology and neuroscience, and several branches of computer science such as artificial intelligence and the semantic web. Within the robotics community, a great variety of large-scale research projects have been funded, and a growing number of publications can be found within the proceedings of recent major international conferences.

A specific kind of inference that lacks in existing non-robotic and robotic systems was identified in in section 1.1 of this thesis. This shortcoming was further incarnated into main objective (B): "To enable inference mechanisms that allow a robot to work with combinations of words used to describe objects, even if these words have never been previously taught together". In chapter 3, we introduced the term *robot imagination* to denote a robot's capacity to "imagine" or generate mental models through mental imagery given combinations of words that have never been taught together, but have been taught separately. Robot imagination precisely tackles our second main objective, and is the aim of the the Inference block in RIS. The Inference block is composed by a single component, the Imagination Core. This component will be further described in the next section.

5.1 Imagination Core

Let the Robot Imagination application be I, which can be seen in Eq. 5.1. Robot Imagination has the potential to create previously unexisting Knowledge K' from existing Knowledge K and Words W.

$$I : (K \times W) \rightarrow K' \tag{5.1}$$

Robot Imagination in RIS involves a sequence of processes. The first process performed within the Imagination Core is Prediction. In the context of objects in RIS, this is the process of determining the features of an object, given a query of words that describe it. Prediction in RIS currently looks for a single solution in the Feature Space $F \epsilon \mathbb{R}^n$, the point that best represents the words of the query. The Prediction application is I_1 in Eq. 5.2.

$$I_1 : (K \times W) \rightarrow F \tag{5.2}$$

The subset of words W contained in a query should have been previously taught to the robot and thus contained in the robot's acquired dictionary D (this notation follows the one given in section 4.2.3). Recall that Robot Imagination becomes interesting when generating mental models from combinations of words that have never been taught together, but have obviously been taught at some time. Two different algorithms to perform the Prediction process will be presented. First, the basic prediction algorithm, which was presented by the author in [83], will be reviewed in subsection 5.1.1. Then, a new enhanced version of the original algorithm will be presented in subsection 5.1.2.

Once the features of the object of query have been predicted, the Imagination Core may perform either Object Recognition or Object Reconstruction. Object Recognition in the Imagination Core is a relatively straightforward process. The Computer Vision component streams the extracted features

of objects in a scenario. These features can be directly compared to the predicted features in $F \epsilon \mathbb{R}^n$ using, e.g. an n-dimensional Euclidean distance as a metric. Finally, the Imagination Core returns the 3D Cartesian space position (obtained from the Computer Vision component) of the object of that minimizes the selected metric. The Object Recognition application is I_{2A} in Eq. 5.3.

$$I_{2A} : F \rightarrow \mathbb{R}^3 \tag{5.3}$$

Object Reconstruction is a more complex process, that will be seen in subsection 5.1.3. The Object Reconstruction application is I_{2B} in Eq. 5.4, which maps to a set of k 3D Cartesian space positions that represent the reconstructed object.

$$I_{2B} : F \rightarrow (k \times \mathbb{R}^3) \tag{5.4}$$

Newly generated Knowledge K', which is composed by 3D Cartesian space coordinates in both Eq. 5.3 and Eq. 5.4, is not stored in RIS. It passed on to the Execution Core for real world action, which will be seen in chapter 6.

5.1.1 Basic Prediction Algorithm

In a nutshell, the basic prediction algorithm can be described as a sequence of the following four specific steps.

1. Generalization. Queries to the Imagination Core contain words that describe an object. Supposing they have been previously taught (even if not taught together, each can be found in D), each word denotes a Semantic Subspace in S. Each point cloud in F corresponding to each Semantic Subspace is generalized by fitting a hyperplane to it.

2. Intersection. The intersection of these hyperplanes creates the geometrical construct of valid solutions, M.

3. Projection. The center of masses of the original point clouds are orthogonally projected on M.

4. Averaging. An equally-ponderated average of the projected points returns the final predicted point in F.

The rest of this subsection will be dedicated to explaining the assumptions that lead to these four steps, the mathematical artifacts involved in implementing such a prediction, and certain special cases where steps can be skipped for optimization.

Generalization

Generalization in the RIS basic prediction algorithm is performed by fitting a hyperplane h_w of order $n-1$ to each point cloud in $F \epsilon \mathbb{R}^n$ that corresponds to the Semantic Subspace of a given query word. The use of hyperplanes can be criticized for several reasons (e.g. it is possible that the shape of a given point cloud may not resemble a hyperplane, and there are also risks of over-generalization since hyperplanes extend to infinite). However, fitting to hyperplanes provides the following three benefits.

- From the algebraic definition of a hyperplane, and with the assumption that the n-dimensional points of a given Semantic Subspace fit relatively well to this selected model, all the linear dependencies and couplings that can occur between features for the word w are simultaneously captured, represented, and "extended" across $F \epsilon \mathbb{R}^n$.

- Hyperplanes are defined in all space, so the "meanings" of words are also extended across, and along, all features. While artifacts of machine learning typically seek for similarity upon elements of a clustering task, an important aspect of RIS is on generating new knowledge, extending meanings to explore the unexplored, even if that means generalizing towards infinite values.

- Another practical benefit, which involves the intersection step performed after generalization, is that we assure the existence of intersections between these representations and thus "meanings" of words. This is precisely because hyperplanes h_w are of order $n-1$ in $F \epsilon \mathbb{R}^n$. Lower order geometrical constructs would not assure intersection, e.g. intersection of 1D lines in 3D Cartesian space.

Let q be the number of words of a given query, a total of q hyperplanes are generated. An example of three generalized words ($q = 3$) as hyperplanes in a 3D Feature Space ($n = 3$) can be found in Figure 5.1. Hyperplanes are equivalent to planes in this example. Extra emphasis should be given on the fact that the generalized representation of words are the actual hyperplanes, and *not* the resulting divisions of n-dimensional space.

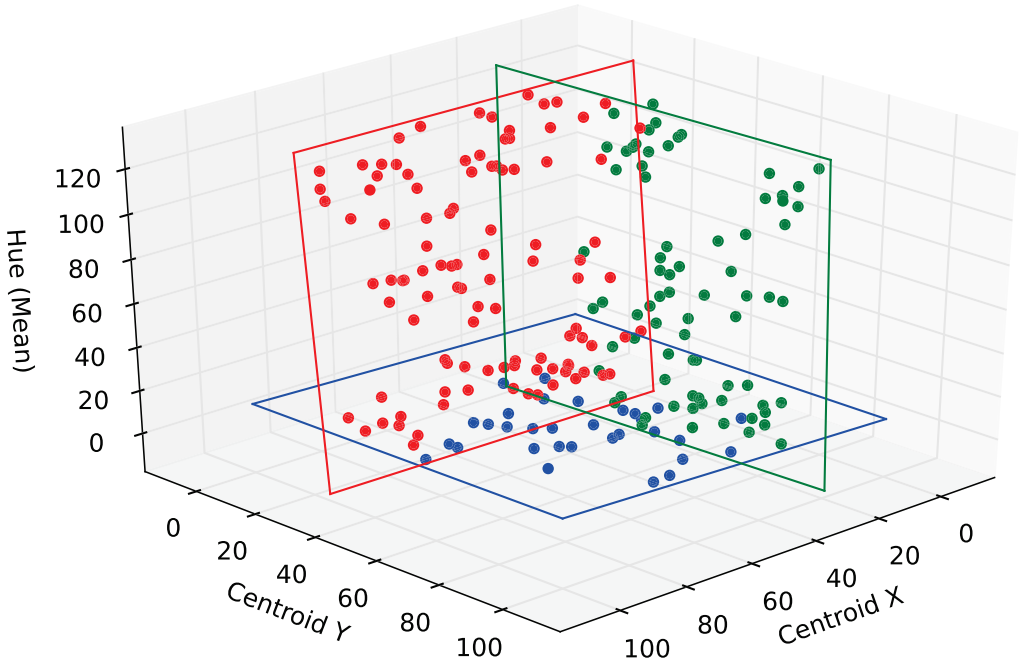

Figure 5.1: Generalized words in a 3D Feature Space are planes that represent the area that is relevant to each word.

A fair amount of machine learning regression algorithms can be used to obtain a hyperplane which fits to a set of points (e.g. orthogonal distance regression, which requires an initial guess and may result in local minima).

We make use of the linear Principal Component Analysis (PCA) algorithm due to its statistical and non-recursive nature. By definition, PCA is made to convert a set of observations into a set of uncorrelated representative variables called Principal Components (PC) [103]. The statistical point model of linear PCA assumes that the set of permissible shapes form a Gaussian distribution, i.e., all possible shapes can be written as a linear combination of a set of eigenshapes obtained by doing principal component analysis on the training data set [104]. The linear assumption is the most common variant of PCA, so the term "linear" will be dropped within the rest of this document. PCA is mathematically identical to ANOVA-simultaneous component analysis (ASCA) [105], but differs in semantics and purpose. The most common use of PCA is to supply a lower-dimensional representation space of an n-dimensional multivariate dataset with the minimum loss of information. For this purpose, the dataset is projected on a hyperplane that is defined by the Principal Components (PC). PC are the vectors that explain the maximum variances of the original dataset. We do not need to perform the whole PCA process of projection in space; we simply exploit one of the intermediate steps. We take advantage of PC computation and use the $n-1$ generated components as the defining vectors of each h_w hyperplane. A summary of PCA computation for a word query w is performed as follows.

1. Mean Subtraction. Given a point cloud that corresponds to the word w (its corresponding Semantic Subspace in F), its mean for each feature is calculated as in Eq. 5.5.

$$\bar{\boldsymbol{f}} = \frac{1}{p_w} \cdot \sum_{i=1}^{p_w} \boldsymbol{f_i} \tag{5.5}$$

   Where p_w is the number of points of the Semantic Subspace that corresponds to w. This process is repeated for every query word w. With this calculation we obtain the center of mass of each point cloud, which is each subtracted from the original point cloud.

2. Covariance Matrix Calculation. It discovers the relation between the features. As previously, F is the Feature Space, and we now denote the data of all the words for a single feature as F_i (Eq. 5.6).

$$\mathbf{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_q \end{bmatrix} \tag{5.6}$$

Where q is the number of words of the query. We define the covariance between two features as in Eq. 5.7.

$$\text{cov}(F_i, F_j) = \text{E}\left[(F_i - \text{E}(F_i))(F_j - \text{E}(F_j))\right] \tag{5.7}$$

Where $\text{E}(F_i)$ is the expected value of the corresponding feature F_i. Currently, all our features weight the same in the calculation, so the expected value can be substituted by the previous average. The covariance matrix Σ is expressed as in Eq. 5.8.

$$\Sigma = \begin{pmatrix} \text{cov}(F_1, F_1) & \cdots & \text{cov}(F_1, F_q) \\ \text{cov}(F_2, F_1) & \cdots & \text{cov}(F_2, F_q) \\ \vdots & \vdots & \vdots \\ \text{cov}(F_q, F_1) & \cdots & \text{cov}(F_q, F_q) \end{pmatrix} \tag{5.8}$$

3. Eigenvectors and Eigenvalues. The eigenvectors of the covariance matrix represent the directions of the variances of the data, and their associated eigenvalues inform about the quantity of the variance explained by each eigenvector. These eigenvectors, in descending order of variance explanation, are the previously described PC. We also order the eigenvectors in descending order by means of their eigenvalues. An eigenvector is defined as a vector that, when multiplied by a matrix, results in a vector equivalent to the original one multiplied by a scalar

(called eigenvalue). An eigenvector of a square matrix A is a non-zero vector v that, when multiplied by A, gives as a result the same vector v multiplied by a single number λ. This is expressed in Eq 5.9, the "eigenvalue equation".

$$A\boldsymbol{v} = \lambda\boldsymbol{v} \tag{5.9}$$

Where λ is a scalar called eigenvalue of v. Eigenvectors are actually the vectors for which matrix A elongates or shrinks. This amount of modification is represented by the eigenvalue. So, eigenvectors are those vectors which are most elongated by the original data of the point cloud. As the covariance matrix represents couplings between features, the maximally elongated vectors are the best vectors explaining the dispersion.

These sorted eigenvectors are the PC. We extract the largest ordered $n-1$ components, from which we generate the corresponding h_w. The main hypothesis of the author, that leads to discarding the smallest component, is that objects are characterized by invariants, and object descriptions explain the invariants of objects[1]. The smallest component represents invariance of features across training samples that correspond to the same descriptive word. We want to conserve this invariance when we extend "meaning" of each word. This occurs if the hyperplane is not extended upon the smallest components (passing from n to $n-1$, this refers to only the smallest of the components). On the other hand, we extend along the components with greater variance, because the same hypothesis indicates it is safe to vary where there is already variance, because this does not represent to object and in turn greatly promotes generalization and exploring the unexplored.

[1]The author's additional lemma is that high variance could be used to understand and automatically acquire action names, e.g. "move" would imply principal components with strong decompositions along axes related with object centroid positions, and "paint" would have strong decompositions along axes related with object colors.

Formally, a hyperplane can be constructed with a perpendicular vector $\bar{a} = (a_1, a_2, ..., a_n)$ and a scalar b. This combination defines the general scalar equation of a hyperplane (Eq. 5.10), with n being the highest dimension.

$$\boldsymbol{a}_1\boldsymbol{x}_1 + \boldsymbol{a}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{a}_n\boldsymbol{x}_n = \boldsymbol{b} \tag{5.10}$$

This equation can be expressed in matrix form as $Ax = b$. Therefore, our aim is to obtain this perpendicular vector and the point, in order to obtain the hyperplane. To achieve it, we use the properties of the dot product (Eq. 5.11).

$$\bar{\boldsymbol{c}} \cdot \bar{\boldsymbol{d}} = \sum_{i=1}^{n} \boldsymbol{c}_i\boldsymbol{d}_i = \boldsymbol{c}_1\boldsymbol{d}_1 + \boldsymbol{c}_2\boldsymbol{d}_2 + \cdots + \boldsymbol{c}_n\boldsymbol{d}_n \tag{5.11}$$

According to the dot product definition, two non-zero vectors are orthogonal if their dot product is equal to zero. To define a hyperplane of $n-1$ dimensions, in an n dimensional space, we need $n-1$ vectors v contained in the hyperplane. From PCA, we obtained a set of vectors which are contained inside the hyperplane. A dot product is performed for each of these vectors with a, because a must be orthogonal to each of them. Let us remember that a is the vector we want to find, the vector that is orthogonal to the hyperplane. These Principal Components multiplied with a lead to a system of linear equations defined as in Eq. 5.12.

$$\begin{aligned}
\bar{\boldsymbol{v}}_1 \cdot \bar{\boldsymbol{a}} = 0 &\Rightarrow \boldsymbol{v}_{11}\boldsymbol{a}_1 + \boldsymbol{v}_{12}\boldsymbol{a}_2 + \cdots + \boldsymbol{v}_{1n}\boldsymbol{a}_n = 0 \\
\bar{\boldsymbol{v}}_2 \cdot \bar{\boldsymbol{a}} = 0 &\Rightarrow \boldsymbol{v}_{21}\boldsymbol{a}_1 + \boldsymbol{v}_{22}\boldsymbol{a}_2 + \cdots + \boldsymbol{v}_{2n}\boldsymbol{a}_n = 0 \\
&\quad\vdots \qquad\quad \vdots \qquad\qquad \vdots \\
\bar{\boldsymbol{v}}_k \cdot \bar{\boldsymbol{a}} = 0 &\Rightarrow \boldsymbol{v}_{k1}\boldsymbol{a}_1 + \boldsymbol{v}_{k2}\boldsymbol{a}_2 + \cdots + \boldsymbol{v}_{kn}\boldsymbol{a}_n = 0
\end{aligned} \tag{5.12}$$

Where $k = n - 1$. This system of equations is indeterminate, because it has n unknowns and $n-1$ equations. To cross this gap, we include an extra constraint $a_1 = 1$, which is equivalent to elongating one variable of a and

letting the rest get adapted, obtaining a system with a unique solution. This trick does not affect to the solution, and it only changes the magnitude of the resulting vector. The vector's direction is maintained, which is in what we are interested. The obtained vector $\bar{a}$ is orthogonal to the hyperplane.

The only remaining unknown parameter is b, which is a scalar. If we substitute A in $Ax = b$ (the hyperplane equation) with $\bar{a}$, and substitute x with one point in the hyperplane, b is directly obtained and the hyperplane is now completely defined. The point in the hyperplane selected has been the previously calculated center of mass of the point of the cloud (Eq. 5.5).

Intersection

Up to this point we have defined the steps followed to generalize words in the Feature Space. This second step in the prediction algorithm is achieved by determining the geometrical figure that contains all of the valid solutions. In this basic prediction algorithm, this geometrical figure is the construct that results from the intersection of the hyperplanes that represent the words of the query. In the resulting geometrical construct, the "extended meanings" of different words meet. Given a query composed by q words (q would be 2 in a query such as "Imagine a yellow square"), we define the geometrical figure that contains all of the valid solutions M as the one resulting from the intersection of the hyperplanes of the corresponding query words. Formally,

$$M(w_1, ..., w_q) = h_{w_1} \cap ... \cap h_{w_q} \tag{5.13}$$

With q being the number of query words. This geometrical figure that results from the intersection of hyperplanes may vary in its number of dimensions, depending of the number of hyperplanes intersecting. The intersection of q hyperplanes can lead to different figures, in function of the number of hyperplanes, and the order of the space.

Projection

The order of M can be calculated as $n - q$, and the possible resulting figures are points, lines, planes, or high-dimensional hyperplanes (this is, M is in fact always a $(q - n)$-dimensional "flat" surface). No matter what "shape" the resulting figure has, we orthogonally project the center of mass of each n-dimensional point cloud, corresponding to each query word, on M. There are two reasons behind this orthogonal projection of the center of mass.

- Orthogonal projection gives us the solution on the valid solution geometrical construct that is closest to the original data of a given point cloud.

- Projecting the center of mass promotes ranges of values that are common in objects, which is useful when not all features are specified by words (e.g. when the number of query words q is much smaller than the dimension n of the Feature Space).

A number of algorithms can be used to orthogonally project a point on a high-dimensional "flat" surface (e.g. Gram-Schmidt). The technique used is the Modified Gram-Schmidt process for orthogonalization (MGS), due to its numerical stability properties with respect to the original Gram-Schmidt process. MGS performs an orthogonal projection of a vector onto another one, aimed at constituting an orthogonal base. A simple projection of a vector v into another vector u is defined as in Eq. 5.14.

$$\text{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u} \tag{5.14}$$

Where $\langle \cdot, \cdot \rangle$ represents the dot product. But, when the searched projection must be orthogonal, MGS states that a vector v orthogonally projected into u, is defined as in Eq. 5.15.

$$\mathbf{u} = \mathbf{v} - \text{proj}_{\mathbf{u}}(\mathbf{v}) \tag{5.15}$$

This equation involves only two vectors, but this formula can be recursively applied to create an orthogonal base u from k non-orthogonal vectors v (Eq. 5.16). This multivector projection is necessary because M can be highly dimensional.

$$\begin{aligned} \mathbf{u}_k^{(1)} &= \mathbf{v}_k - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_k) \\ &\vdots \\ \mathbf{u}_k^{(k-1)} &= \mathbf{u}_k^{(k-2)} - \text{proj}_{\mathbf{u}_{k-1}}(\mathbf{u}_k^{(k-2)}) \end{aligned} \tag{5.16}$$

The projection is obtained by subtracting the final $\mathbf{u}_k$ from the center of mass to project, because this $\mathbf{u}_k$ is perpendicular to the hyperplane and its modulus is the orthogonal distance between the two. For q query words, q points are projected on M.

Averaging

Prediction in RIS currently looks for a single solution in the Feature Space $F \epsilon \mathbb{R}^n$, the point that best represents the words of the query. The projected points will be on different zones of M. An average with equal weights is applied to these points, obtaining a single solution, which is also contained in M (see an example in Figure 5.2). The weight of specfic words can be augmented, if stress in speech can be detected and quantified. The following three advantages of the solution obtained in the Feature Space F are highlighted.

1. Complete. Because it is completely defined in all the Feature Space.
2. Relevant. Because it is contained in the area where the extensions of meanings intersect.
3. Balanced. Because all the words influence the same way in the final solution.

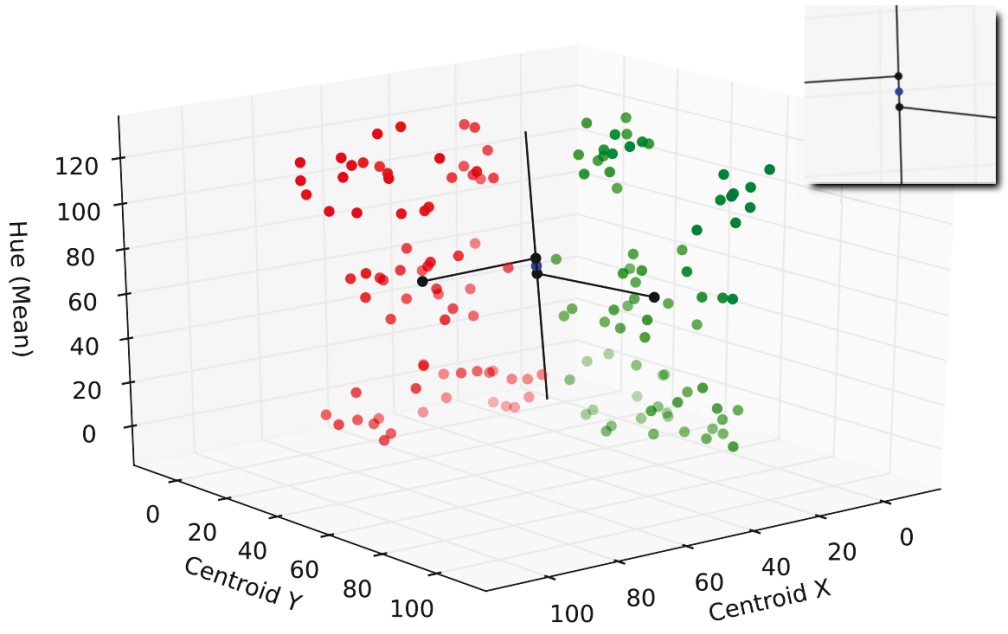

Figure 5.2: Unique solution using MGS for two query words but three dimensions. The augmented region shows the orthogonal projection of centers of mass.

Special Cases

The basic prediction algorithm can be used regardless the dimension n of F and the number of query words q. However, in order to reduce computation time, some special cases for optimization can be defined. We consider special cases, those where the order of M leads to situations that can be simplified. It is important to notice that following cases are pure simplifications to improve efficiency, but could be also solved with the previous method.

- Special case: **q=1**

  In this case, there are no intersections, because a single hyperplane is generated from a single query word. In absence of more information, the center of mass of the point cloud becomes the solution. The reason behind this decision is that we suppose that for a point cloud, the center of the cloud will be the most relevant area possible. Note that the effects on non-convex point clouds should be studied.

- Special case: **q=n**

  In this situation, the order of M is null (remember that its dimensionality is $q - n = 0$), so the resulting figure is a point. Thus, further computation is not necessary, and the solution is this point. The interpretation of this situation is that in the description provided as query, there is one word representative of at least one of the features in the space, so no further interpolation is required to supply undefined features. Figure 5.3 depicts an example of this special situation, where $n = 2$ and $q = 2$. Figure 5.4 depicts a different example of this special situation, where $n = 3$ and $q = 3$.

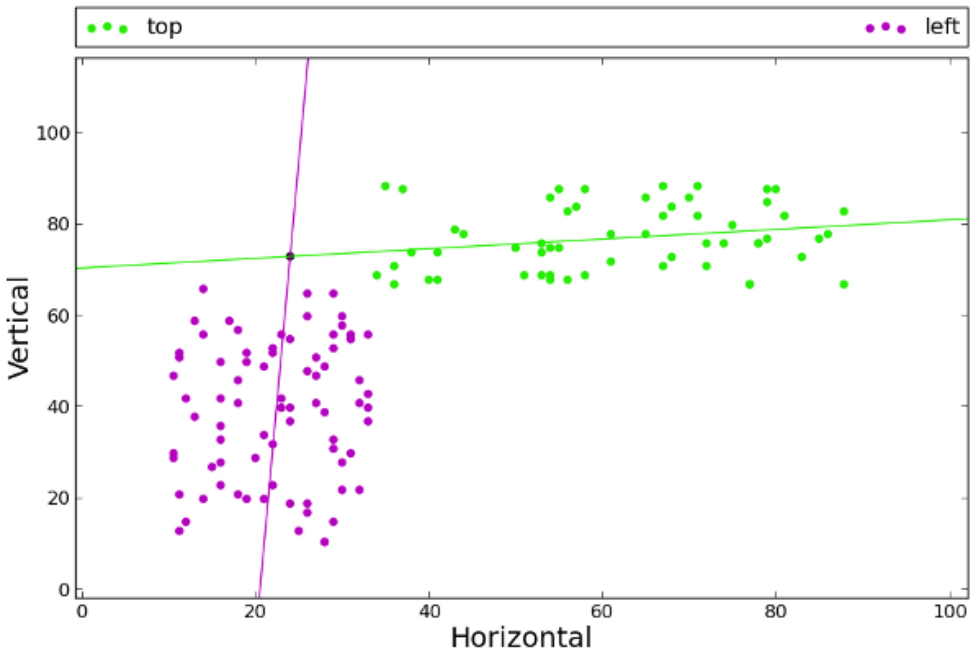

Figure 5.3: In this special case where $n = 2$ and $q = 2$, the intersection point (black) becomes the solution in the Feature Space.

The mentioned special cases are pure simplifications for efficiency. Specifically, for $q = n$ we avoid orthogonally projecting the center of masses of the words on a point, as it will return the same point. Similarly, we save several steps in the case of $q = 1$. In this case, only one hyperplane is generated. The center of mass of the word is contained on M, so its orthogonal projection is

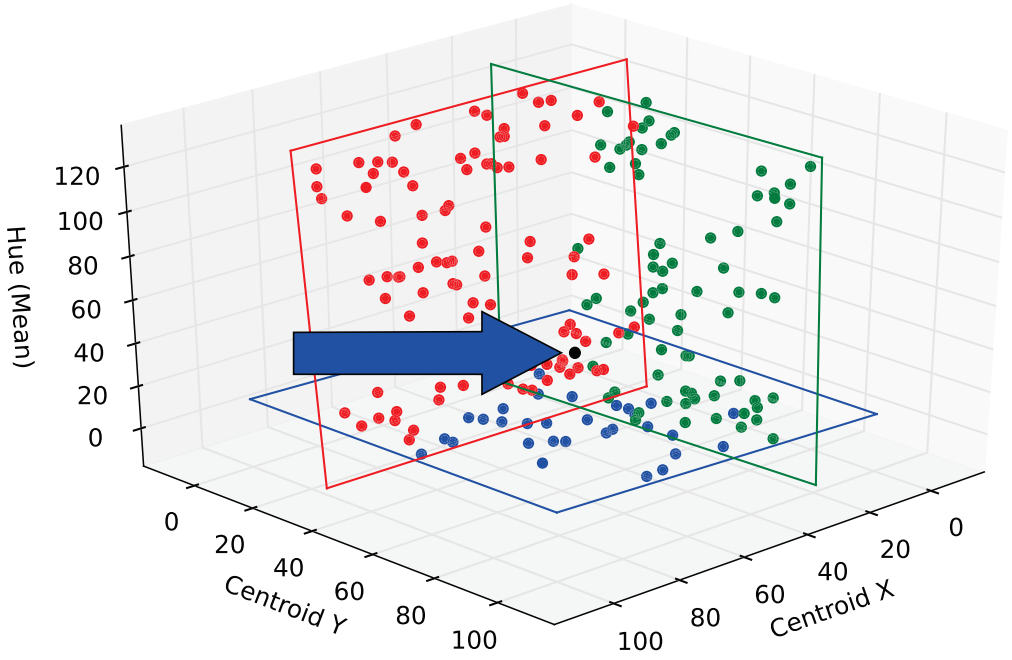

Figure 5.4: In this other special case where $n = 3$ and $q = 3$, the intersection point (black) also becomes the solution.

again itself. Note that $q > n$ is currently not treated.

It is important to notice that our systems improves the results through time as it gets more samples. The more dense a point cloud is, the better the analysis that can be performed, with better regressions and intersection. When more training is performed, the extended algorithm presented below is also more robust. Note that training can be performed in concurrence with inference queries that involve prediction.

5.1.2 Enhanced Prediction Algorithm

The presented basic prediction algorithm has established the basis for prediction in Robot Imagination, but some issues may be outlined.

- Context dependency: The basic prediction algorithm is not capable of correctly managing words that change depending on the context. An example of this could be the word “green”, which could refer to an

object's color, or to its state of ripeness. This may affect the Robot Imagination System as it can represent a total loss of linearity of the words with respect to any of the features.

- Periodicity: The basic algorithm is susceptible to malfunction in the presence of repeating patterns. For example, in the HSV color space, the hue "red" can be represented as 0° or as 360°. A hyperplane fit to these values can tend to settle at 180°, which is obviously wrong.

To solve these limitations, or at least, reduce their influence, we have developed two extensions which enhance the algorithm solving capabilities: Context Detection, and Hyperspherical Shape Detection through Eigenvalue Analysis. The rest of this subsection will be dedicated to explaining these extensions.

Context Detection

To avoid the context dependency and periodicity explained, we have created a context detector, which selects the appropriate point cloud to model, in function of previous experiences. This context detector implies two modifications of the basic prediction algorithm: one in generalization, and the other in the intersection finder.

In the previous basic prediction algorithm, when generalizing a word, we used all of the points labeled with this word to perform the generalization. This is, all of the points of a Semantic Subspace were treated as a single cluster. We now perform a previous step of clustering of these points, to enable an independent generalization of each cluster. Points are clusterized using Aglomerative Single-Linkage Hierarchical Clustering [106]. For the whole raw data, the algorithm will look for the two most similar data points and merge them to create a new pseudopoint (the average of both points). Each iterative step takes the next two closest points (or pseudopoints) and merges them. This process is generally continued until a stop criterion is

reached. In our case, the stop criterion is executed when the distance between the two most closest clusters ω is bigger than a threshold ω_{max}.

Once the clusters have been detected, we have to decide which cluster to use for the hyperplane intersection, among those available. For that purpose, we use an adaptation of word bigram co-occurrences. A bigram is a sequence of two adjacent elements in a set of elements (e.g. red:big), and the co-occurrence CO() is the frequency of occurrence of two elements in a series of elements (e.g. CO(red:blue) = 0.5). Bigrams and co-occurrences are broadly used in semantic analysis [107]. Our adaptation is called "Cluster Bigram Co-Occurrence". For this, we associate a new function to the Semantic Point Cloud S (in addition to Eq. 4.2 and Eq. 4.3), which can be seen in Eq. 5.17.

$$R_A : S \rightarrow A \tag{5.17}$$

Where A is the Accompanying words list, which itself contains lists of words present in the description from which the point was generated. This can be seen in the following example.

1. An n-dimensional point is generated from sensor information, with the description: "red", "big", "long".

2. Three points with the same coordinates are created in the Feature Space F, and three entries are created in the Dictionary D: "red", "big" and "long", respectively.

3. The first entry in A will contain the words "big" and "long".

4. The second entry in A will contain the words "red" and "long".

5. The third entry in A will contain the words "red" and "big".

The number of Bigram Co-Occurrences (the number of times an accompanying word appears for the set of points of the cluster) are counted for

each cluster of a query word, and the cluster for which the sum of its Bigram Co-Occurrences of accompanying words (of the query) is biggest is selected.

This is best understood with the example depicted in Figure 5.5. In this example, we have set two clusters. Both clusters are formed with points containing the word "red". The accompanying words of some points are shown, which we will use for counting (in practice, all of the points and words are taken into account). The Context Detector algorithm must decide which of the two clusters will be used for regression. For solving this, it sums the number of accompanying words present in each of the clusters. If, for instance, we are using "big" and "long" as accompanying words, it has to compare the number of points containing "long" or "big" as accompanying words. For the 5 points depicted in the figure, the left cluster would have a score of 4 (3 from "big", plus 1 from "long") and the right cluster would have a score of 1 (from "long"). Our algorithm, only considering these depicted points, would choose the left cluster.

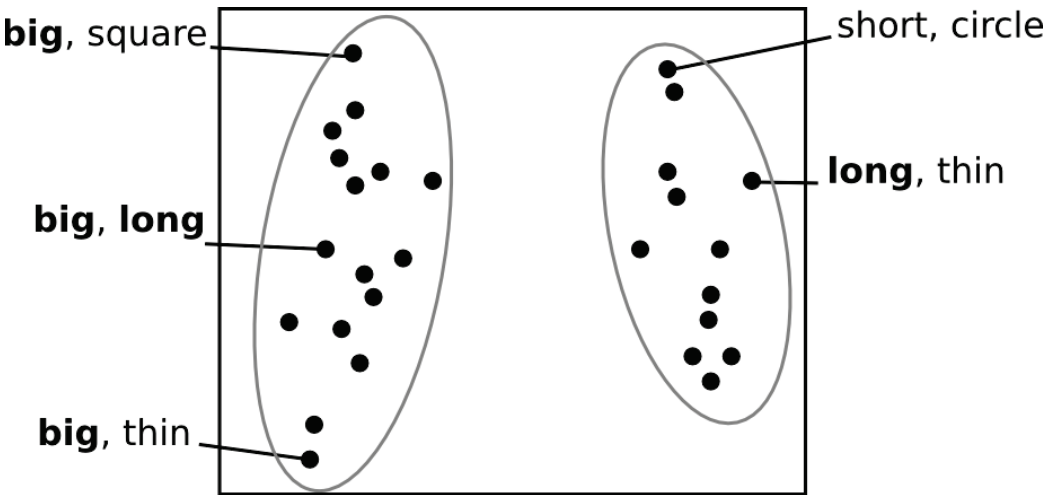

Figure 5.5: The Context Detector accumulatively counts the accompanying words of a query word within each cluster.

Continuing with the same example, and setting two clusters for the word "red", the selected one for hyperplane generation and intersection would be:

$$Cluster = \arg\max_{c \in C}(CO(red_c : big) + CO(red_c : long)) \tag{5.18}$$

Where C is the set of all clusters for the word "red", and "red_c" is the name given to each cluster of word "red". The points belonging to the selected cluster will be the ones used for hyperplane generation in this particular request. The selected cluster may vary depending on the accompanying words.

Hyperspherical Shape Detection through Eigenvalue Analysis

Sometimes, when calculating the Principal Components of a point cloud, we detect that the points are extended in all spatial directions with similar length, as in a hypersphere, so the point cloud cannot be extended in any meaningful direction. This detection is computed by comparing all sorted eigenvalues $v \in E$, in the set of eigenvalues E for a point cloud. We sort them in descending order and, for a n-dimensional space, we check Eq. (5.19).

$$\forall v \in E, j = [0, n-1] : |v_j| - |v_{j+1}| < \Theta_{min} \tag{5.19}$$

Where Θ_{min} is a manually set parameter. In the case that all eigenvalues evaluated are similar (Eq. (5.19) would be true for all eigenvalues), we reduce the point cloud to the single point, formally the center of mass of the cloud. Words where this ocurrs do not participate in hyperplane generation and intersection, and instead the center of mass is directly projected on the intersection element M resulting from the intersection of the rest of the hyperplanes generated from the query words. This mechanism helps work with Semantic Subspaces that have a shape resembles more of a hypersphere than a hyperplane, respecting this shape while maintaining compatibility with the rest of the algorithms.

5.1.3 Object Reconstruction

From Eq. 5.4, we have seen that Object Reconstruction is the process of generating a set of k 3D Cartesian space positions that represent the reconstructed object, given the features of the object. Up to this step in the process, we have obtained the values of the features of the object of imagination through the basic or enhanced prediction algorithm. However, we have not obtained a "mental model", or at least not a human-understandable mental model of the object as a product of Robot Imagination.

To achieve this, the features values inferred in the previous prediction step are set as inputs to an evolutionary computation algorithm (EC), which performs a steady state selection algorithm [108]. The particularity of this algorithm is that, in each generation, after the selection and crossover process, the resulting individuals replace only a few other individuals at a time, instead of other algorithms which replace most of them. The following is a summary of the algorithm's operator details.

1. Selection. A tournament is performed between random individuals. Their fitness values are compared, and winners are selected for crossover.

2. Crossover. Winners are crossed, and their children substitute the worst values from the previous tournament.

3. Mutation. Finally, each child may be mutated with a certain degree of probability.

The termination condition can be set to a certain number of generations without improvement in the fitness value. For the process of generating the mental model, the EC controls the coordinated Cartesian space positions of a set of points inside a 2D or 3D canvas. These points are linked to form a shape. Then, we extract features from the generated image and compare them with the target features. Termination comes once these points are

considered "fit" according to the fitness function, a weighed cost function on the feature errors.

To accelerate the process, we can also act on two levels, similar to the two levels of the visuo-spatial sketchpad presented by R.H. Logie [59]. First, the EC can control the coordinated Cartesian space positions of a set of points inside a *small* 2D or 3D canvas, the equivalent to Logie's "visual cache". These points are linked to form a shape. Then, omitting spatial features, the rest of the features are extracted from the generated image and compare them with the target features. Termination comes once these points are considered "fit" according to the fitness function, a weighed cost function on these feature errors. Finally, these points are spatially plotted on a larger canvas equivalent to Logie's "inner scribe". The algorithm can converge without this two-level scheme, but times may increment significantly as it reduces computational efforts.

5.2 Chapter Summary

The Inference block has been introduced in this chapter, which is composed by the Imagination Core. Robot Imagination first involves prediction of object features, given a query of words. A basic prediction algorithm and an enhanced prediction algorithm have been presented. These predicted features can be used in a straightforward manner for Object Recognition, or to generate "mental models" through Object Reconstruction. The next chapter will present how these products of Robot Imagination can be used to perform execution of tasks in real world environments.

Chapter 6

Execution Block

The Execution block is composed by three components: the Visual Output component, the Speech Output component, and the Execution Core component. Existing components can be used as the Visual Output component, as 2D and 3D visualizers can be found within a number of robot architectures. Similarly, several existing text-to-speech components can be used as the Speech Output component. The Execution Core component *per se* has no equivalent within robotic architectures. It requires a detailed description, which will be given in the following section.

6.1 Execution Core

In RIS, the Execution Core is in charge of Action, which is executed given Knowledge K', potentially creating external stimuli E that close the Perception cycle seen in chapter 4. The Action application is A in 6.1.

$$A : K' \rightarrow E \tag{6.1}$$

Where K' maps to the knowledge that has been generated in the Inference Core. Commands that come from the State Machine modify the Execution Core's behavior, invoking different actions.

The fact that the chain of Action ends in the environment, and not within the robot itself, makes us subscribe to the recently co-authored paradigm of Continuous Goal-Directed Actions (CGDA). In section 2.4 we presented this paradigm as a useful way to teach robots the *consequences* of an action, which is in line with what is required in the Execution Core. Throughout this section, we will present the intrinsics of the algorithm of this work presented in [80], and describe how this paradigm merges with the fabrics of the RIS architecture. Subsection 6.1.1 explains how actions are encoded in CGDA, which corresponds to an action generalization process. The algorithm for recognition of performed actions is described in section 6.1.2. Execution, which will be explained in subsection 6.1.3, is performed through evolutionary computation, that includes action recognition in every iteration.

6.1.1 Action Generalization

In order to teach the Execution Core a new action, we must demonstrate the action we want to teach the robot with several repetitions. This is currently performed off-line, and action names are currently hard-coded. These practices are in line with trends within the robot imitation [14] presented in section 2.4, where the demonstrator could either be the guided robot itself, a human with sensors attached, or video sequences with human movements. In CGDA, the environment is continuously tracked (segmented objects, and optionally, the human's movements) in a Feature Space $F \epsilon \mathbb{R}^n$, analagous to the Feature Space for objects which has been presented in subsection 4.2.2.

Since the nature of electronic camera sensors is in practice discrete (samples are taken at a given rate), the result of several repetitions is actually a point cloud in F. For generalization purposes, we need to extract a representative n-dimensional trajectory from the point cloud that is formed after several repetitions. This generalization process itself is composed by a sequence of three subprocesses: time rescaling, average in temporal intervals, and Radial Basis Function (RBF) interpolation.

Time Rescaling

Before inserting a new action repetition in the point cloud, it is normalized in time (range $[0, 1]$). With this time rescaling, every action execution gets bounded by the same temporal limits, making the algorithm independent of the speed of the repetition. These normalized trajectories are the ones introduced in the Feature Space F to form the action point cloud.

Average in Temporal Intervals

To model the point cloud, we split it up in temporal intervals, fixing one interval per second. The number of seconds is computed from the average duration of the original repetitions. Each interval contains points of all repetitions, in the same percentage of execution. As the repetitions are normalized in time, each interval represents a percentage of action execution, and the number of intervals allows preserving a notion of the original action duration. The representative point of each interval is extracted as the average for each dimension of all points of the interval, as seen in Figure 6.1.

Radial Basis Function Interpolation

These temporal averaged points of the previous step must be joined to form a generalized action, i.e. an object feature trajectory we can consider a generalization. An interpolation in robot motor joint space could create a jerky joint trajectory, so literature, e.g. [65], commonly uses regressors such as Gaussian Mixture Regression (GMR). However, working in the Feature Space, we use an interpolator to assure the trajectories pass through the target points (which are the states of the object in an instant). We use a Radial Basis Function (RBF), which is an interpolation technique based on measuring the influence of every known point over the queried point [109]. The RBF interpolation $f(x)$, which will become the final generalized trajectory, is mathematically expressed as a sum of radial basis functions, Eq. 6.2.

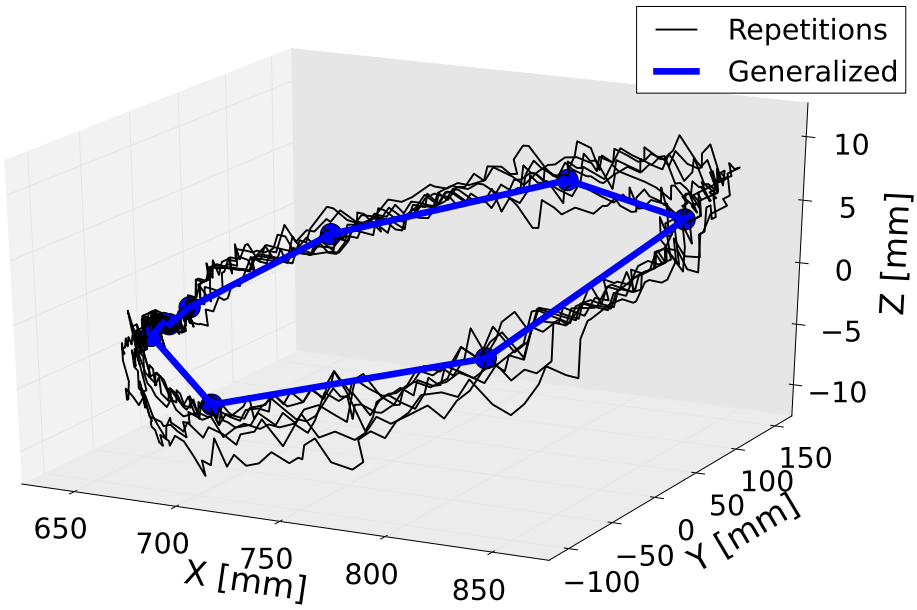

Figure 6.1: Plot representing a three feature trajectory. Black lines are training action repetitions. The blue line is the generalization of all the repetitions.

$$f(x) = \sum_{i=1}^{N} w_i \, \phi(r) \tag{6.2}$$

Where the radial basis function is denoted as ϕ, and N is the number of radial basis functions, equal to the number of intervals. We have selected the linear radial basis function, seen in Eq. 6.3, as opposed to other alternatives (e.g. Gaussian, multiquadratic, polyharmonic spline...), because the smoothness of the trajectory in the feature space is not relevant.

$$r = \|x - x_i\| \tag{6.3}$$

Where x_i represents the coordinates of each interval's known point, so the linear function radial basis r is the distance between the known point x_i and the queried point x measured with the L^2 norm. Eq. 6.2 is further expanded to Eq. 6.4 with the linear radial basis function of Eq. 6.3.

$$f(x) = \sum_{i=1}^{N} w_i \, \phi(\|x - x_i\|) \tag{6.4}$$

Where the coefficients w_i are the weights of the specific known points over the queried point x, and are the values to be solved. As the interpolation is known at known points, the weight problem is solved as a set of N linear equation with N unknowns, Eq. 6.5.

$$\begin{aligned} f(x_1) &= \sum_{i=1}^{N} w_i \, \phi(\|x_1 - x_i\|) \\ &\vdots \\ f(x_N) &= \sum_{i=1}^{N} w_i \, \phi(\|x_N - x_i\|) \end{aligned} \tag{6.5}$$

Once the interpolated function is returned, we consider this output as the generalized function of an action. Its physical meaning is how the state of the object, regarding its features, changes across the execution of the action.

6.1.2 Action Recognition

Here we aim to recognize an action by comparing a query action with the previously generalized or synthetically created ones. Action Recognition in CGDA is achieved by comparing feature trajectories, which is equivalent to comparing the evolution of feature changes "continuously in time". Given a set of generalized or synthetically created actions, the one with the highest score of feature trajectory similarity with respect to that of a query action is the one that is returned as the recognized action.

The initial treatment of the query action is the same as explained. As they are normalized in time, t values along time can be taken for each action for comparison. The technique used in the comparison is Dynamic Time Warping (DTW). DTW is an algorithm usually used to optimally align two

temporal sequences [110]. Our use of DTW is to compare two time-dependent sequences of points $X = \{x_1, .., x_N\}$ and $Y = \{y_1, .., y_M\}$ with $N, M \in \mathbb{N}$. To compare two elements, a local cost measure (a distance $d(x, y)$) is needed. A lower cost represents a bigger similarity of the sequences. Evaluating all pairs of points between the sequences, using in this case a L^2 norm, we obtain a cost matrix CM, with a size of $N \times M$, Eq 6.6.

$$CM = \begin{pmatrix} d(x_0, y_0) & \cdots & d(x_N, y_0) \\ \vdots & \vdots & \vdots \\ d(x_0, y_M) & \cdots & d(x_N, y_M) \end{pmatrix} \tag{6.6}$$

Once having this matrix, the goal is to find the lowest cost alignment path, which intuitively should run along the lowest cost cells. This alignment is called the warping path. DTW includes some constraints in the path calculation to assure a monotonic advance of the path and to assure that the first elements as well as the last elements are connected to each other. The total path cost $c_P(X, Y)$ is calculated as the sum of the local costs c, Eq 6.7.

$$c_P(X, Y) = \sum_{l=1}^{L} c(x_{nl}, y_{ml}) \tag{6.7}$$

Where L is the length of the path. For programming reasons, the path is usually calculated in an accumulated cost matrix, where each cell represents the cost of the correspondent pair (x, y) plus the cost to reach this cell. In the accumulated matrix, the normalized cost $c_{P_{norm}}(X, Y)$ of the optimal path is expressed as in Eq 6.8.

$$c_{P_{norm}}(X, Y) = \frac{c(x_n, y_m)}{N + M} \tag{6.8}$$

In our case, we use this normalized cost of the optimal path as a measure of discrepancy between dimensions. Figure 6.2 depicts an example accumulated matrix, with its optimal path in red. As DTW is computed between two signals for one dimension only, we consider the total cost of alignment

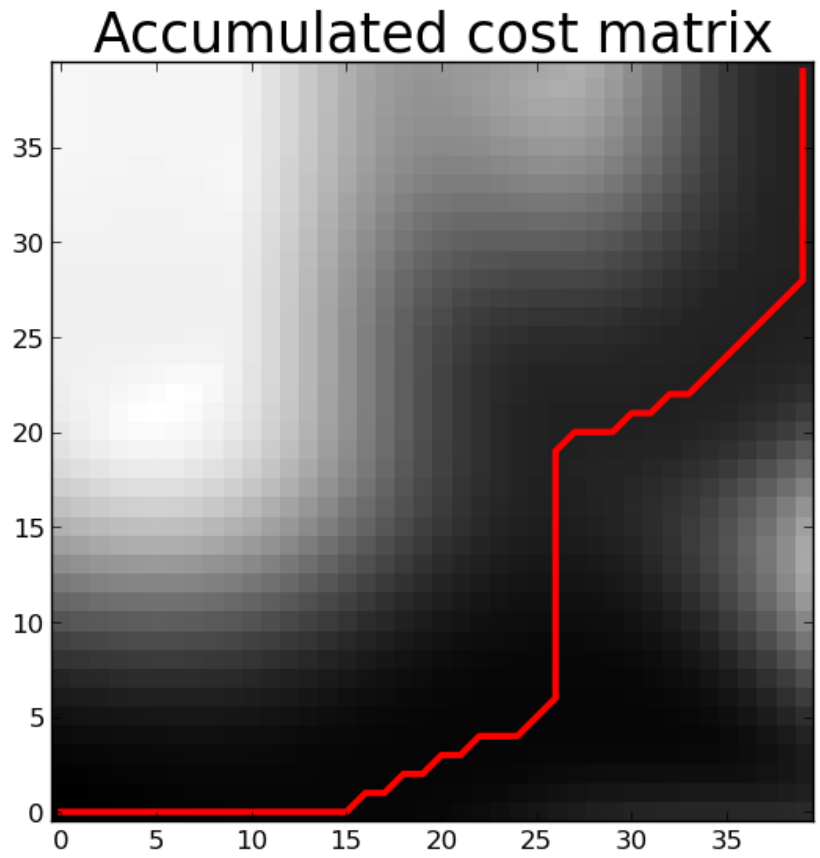

Figure 6.2: Example of accumulated cost matrix for two sequences. White cells represent high cost, while dark cells are low cost ones. The red line is the lowest cost path.

between two n-dimensional trajectories as the sum of the costs of the optimal paths of each dimension, obtaining a single score d.

$$d = \sum_{i=1}^{n} c_{P_{norm}}(X_i, Y_i) \tag{6.9}$$

This score is used as the measurement of discrepancy between two trajectories in the n-dimensional space. The highest score of feature trajectory similarity for Action Recognition is given by the compared Feature Space trajectory with the smallest discrepancy measurement with respect to the action of a given query.

6.1.3 Execution

The Execution Core, fueled by CGDA, has allowed to learn actions in the Feature Space, and also to compare actions in terms of trajectories in this space. However, at this point, we actually lack the robot motor commands for executing an action that generates the desired effects on the environment.

A similar situation was seen in Object Reconstruction of section 5.1.3. We had a reached a solution in the Feature Space, an n-dimensional point, but had no way of porting this solution to the real world. Finally, Evolutionary Computation (EC) algorithms were adopted to find solutions in Cartesian space. While the Object Reconstruction problem presented an n-dimensional point, in Execution we have an n-dimensional trajectory in the Feature Space. With this analogy in mind, EC is again used to to perform a steady state selection algorithm (few individuals are replaced after the selection and crossover process), this time to discover full robot motor trajectories that allow executing a given action.

Instead of using the location of a set of 2D or 3D points in Cartesian space as the parameters that evolve in EC, the robot motor joint positions (the joint space values) are the parameters that evolve to achieve the effects of the action on the environment at each step. To achieve a trajectory composed by k n-dimensional Feature Space points, k *times* the number of used robot motors evolve. At each cycle of the evolution, the effect of the performed action is measured through one of the steps of Action Recognition: the discrepancy score d is used as the fitness parameter of the EC.

As performed in Object Reconstruction, the termination condition is set to a number of generations without improvement in the fitness value. An example fitness evolution for a given action can be found in Figure 6.3. A summary of the operator details of this algorithm (selection, crossover, mutation) was given in section 5.1.3. For practical concerns (e.g. time given the high number of iterations, and danger for the robot or environment),

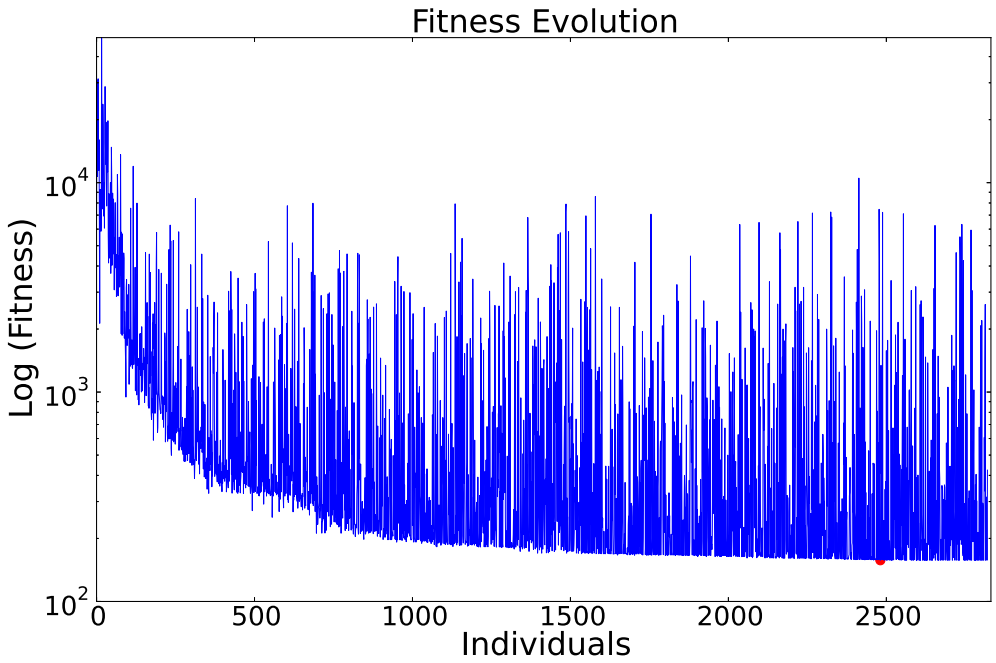

Figure 6.3: Fitness value through evolution. The red point is the minimum value achieved by evolution.

this evolution may be performed in a virtual environment. This virtual environment can be synchronized with the real world and implement object persistence, as studied in section 2.3. The final solution, a full robot motor trajectory, can be executed in the real world environment.

6.2 Chapter Summary

In this chapter, the Action application has been seen, as well as how it is performed in the Execution Core. To accomplish Execution as expected, we have incorporated the recently co-authored Continuous Goal-Directed Actions (CGDA) paradigm. Action Generalization of CGDA has been explained, as well as how Action Recognition is performed. Finally, we explain is computation and performance of Execution through an Evolutionary Computation approach, which uses Action Recognition within the cycles of the evolutionary computation algorithm.

Chapter 7

Experiments

To evaluate the system, we have performed several experiments. Computational implementation aspects that have been shared among experiments will be seen in section 7.1. The Imagination Core's basic prediction algorithm and object reconstruction will be put to work through an experiment that consists in having a manipulator robot draw an imagined object in a simulated environment in section 7.2. Spatial language will be taught to the humanoid robot Teo in section 7.3, proving the benefits of the Imagination Core's enhanced prediction algorithm. The Execution Core's functionality through CGDA will be seen with the replication of a waxing task in section 7.4. A final global validation experiment will be performing a version of a psychological experiment, the Token Test [111], in section 7.5.

7.1 Computational Implementation

Each RIS component has been implemented as a YARP module [112]. These modules are independent and interconnected, interchanging relevant information through YARP ports, and are direct software implementations of the mathematical formalizations explained throughout the previous chapters. For all experiments, object segmentation and processing on 2D images has been implemented using the OpenCV library through the Travis wrap-

per [113], obtaining a periodic transduction and processing loop within a 20 millisecond margin. Object segmentation and processing on 3D images has been implemented using the PCL library [114], additionally accessing to the underlying 3D vision library called Visualization Toolkit (VTK) and developed by Kitware [115], obtaining a periodic transduction and processing loop within a 10 second margin.

7.2 Basic Prediction Algorithm: Drawing

To test the Imagination Core's basic prediction algorithm, we have a robot draw an object it has never seen before. A fundamental aspect of this process is enabling the robot to externalize a mental model of an object it has never seen before. This experiment is performed using the ASIBOT robot [116] in an OpenRAVE-based [117] simulated environment. This robot is a 5 degree-of-freedom medium sized manipulator arm, developed to assist disabled and elderly people in domestic environments.

For the application, a training dataset has been composed, consisting in 300 synthetically generated flat images (Figure 7.1). To accelerate the process, this initial population has been generated automatically. These images are colored figures, which vary in shape and position, over a black background. Linked to the images, 7-word automated descriptions are attached to each image (only representative words, e.g. top-left-dark-blue-fat-straight-box). These words are nouns and adjectives automatically added when the image is generated. Each descriptive word could take upon 3 different values, thus this training population of 300 images represents less than 13.7% of all of the possible combinations. This fact is deliberate, to prove that our algorithms do not need to be trained with all of the existing possibilities, and that even the basic prediction algorithm can manage this type of situation of incomplete knowledge. An additional Gaussian noise of 1% has been added on all features to emulate real world noise.

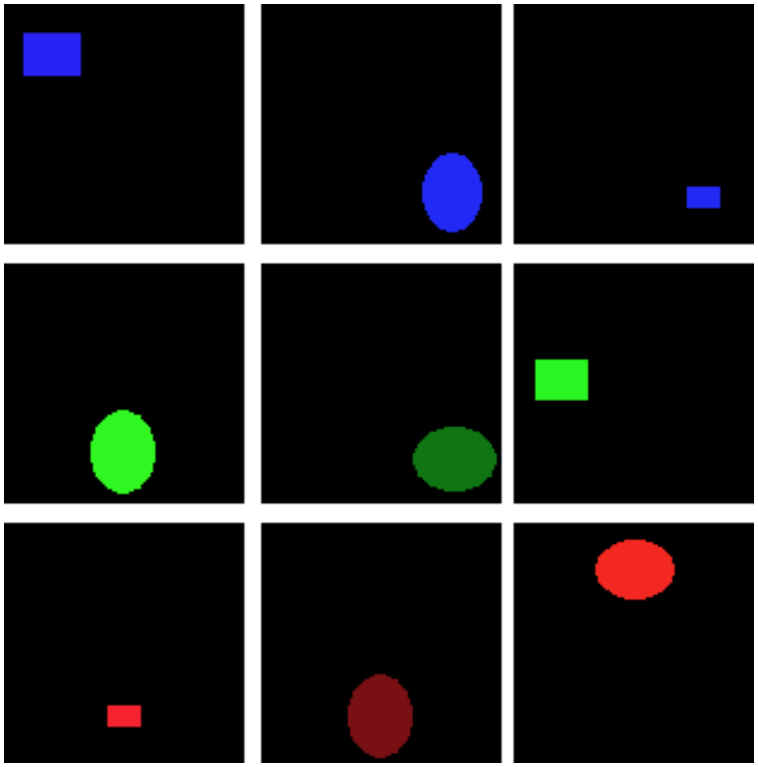

Figure 7.1: Example of training images used to populate the space. For instance, the first image is labeled top-left-dark-blue-fat-straight-box.

In this experiment, 12 features are extracted (see subsection 4.1.3) from the training images. The first two correspond to the vertical and horizontal position of the centroid of the object in the image. The average hue and value are used to define the color, and another 8 features (aspect ratio, area, rectangularity, maximum and minimum axis, solidity, arc, and radius) are used to characterize the contour of the object.

For generating the mental model from a query, the Evolutionary Computation (EC) algorithm of object reconstruction controls the 2D coordinate positions of a set of points inside a flat canvas. These points form a shape together, and the features of the generated image are extracted. The fitness is calculated as the simple sum of feature differences between the target data and the generated one. These target values of the EC are the values of the features returned from the basic prediction algorithm of subsection 5.1.1. The EC tournament size is set to 3 individuals, the mutation probability is

set to 70%, and the termination condition is set to 50 generations without improvement in the fitness value. Several generated mental models can be seen in Figure 7.2.

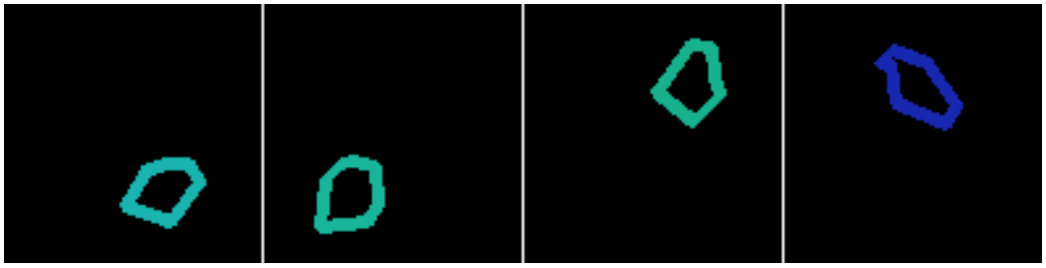

Figure 7.2: Mental models generated from different queries: (a) "bottom right", (b) "bottom left", (c) "top right", (d) "top blue".

Notice that unnamed features in the query words, such as color or shape, remain uncertain but bounded in their values. Despite they can be also modified in the mutation process, the target values with which they are compared are undefined values from the intersection of hyperplanes, but take intermediate values due to the orthogonal projection step of the prediction algorithm. These mental models are composed by sets of 2D coordinate points. Fixing a height for a flat painting surface, these points can be used directly as a trajectory for the drawing application. Figure 7.3 depicts a screenshot of the simulated ASIBOT robot performing the drawing task.

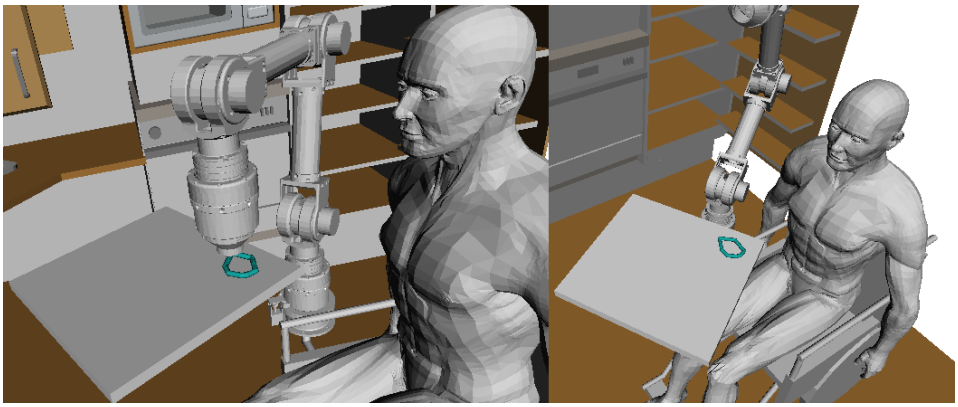

Figure 7.3: Simulation of the ASIBOT robot arm drawing the mental model generated from a "bottom left" query.

7.3 Enhanced Prediction: Spatial Language

To test the Imagination Core's enhanced prediction algorithm, we perform a spatial language experiment. Spatial language is complicated to understand, because its meaning depends on the person describing and the used possessive adjectives. This is the case of "my left side" as opposed to "your left side". In this experiment, we teach the robot how to distinguish between perspectives (the egocentric versus allocentric problem), and obtain a real spatial meaning from the descriptions.

7.3.1 Synthetic Test

The training phase is performed using computer-generated samples of a white circle on a black background (Figure 7.4). The white circle is randomly situated in each image, which is then automatically classified according to its coordinates. The samples, whose circle centroid coordinates are their features, are labeled with their spatial characteristics.

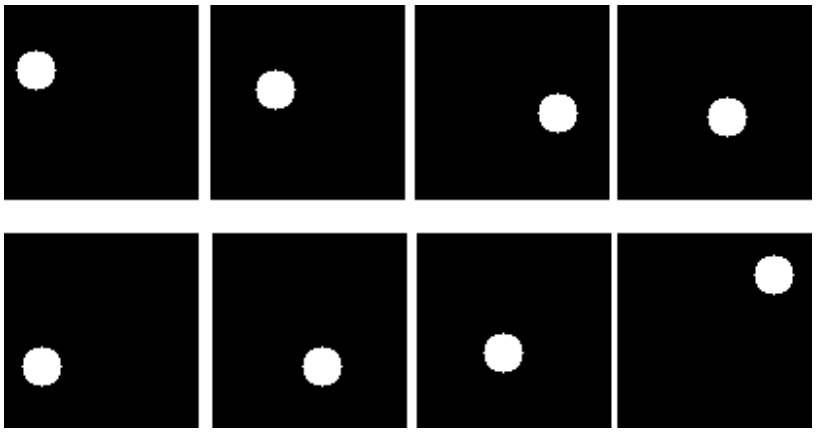

Figure 7.4: Synthetic samples used for training and testing spatial language.

Images are 100 by 100 pixels size and the classification is carried out as follows: height and width are virtually divided in 3 equal parts (resulting in 9 areas). Words are assigned to the samples, based on circle coordinates, and they are: "top", "bottom", "right", "left" and "middle". This last word,

"middle", is used twice, or in with two different "meanings", because it can represent vertical but also horizontal middle space. The dataset is composed of 300 samples, of which 70% are used for training. The test phase uses the remaining 30%. In this case, all possible combinations of words are fed to the grounding database, except for contradictory pairs (e.g. "left-right").

The first result, shown on Figure 7.5, represents the basic prediction algorithm (subsection 5.1.1) output, in 2D coordinates, for queries that combine the different spatial words ("top", "middle", etc.).

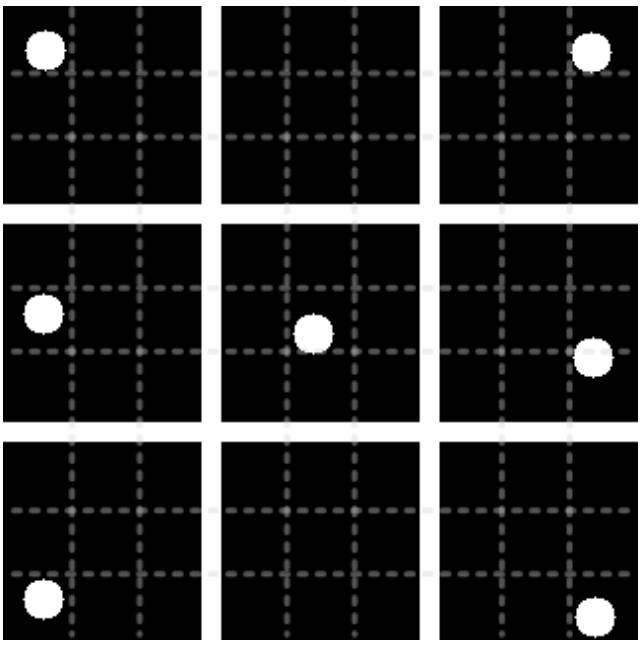

Figure 7.5: Basic prediction algorithm. Visual output (circle at x, y) for two-word queries. Every image is positioned in its respective place in the figure. For instance, the upper left image corresponds to a "top-left" query. For a better comprehension of the results, the dotted gray lines represent the approximate margin for different areas in each square.

Once having these values, they can be now compared with the samples reserved for the test phase. For this task, Root-Sum-Square (RSS) results are compared (Eq. 7.1), once for each coordinate x and y.

$$f(x) = \sqrt{\sum_{i=1}^{n}(x_i - \mu)^2} \tag{7.1}$$

Where n is the number of samples, x_i is the single coordinate value for a test sample, and μ is the result obtained of the training phase (shown on Table 7.1) for these words combination.

Table 7.1: Basic prediction algorithm. Numerical output (x, y) for a "word-word" query.

	top	bottom	left	right	middle
top	51.3 , 78.0	*	21.0 , 78.5	76.1 , 77.6	-40.7 , 79.5
bottom	*	51.5 , 21.3	20.9 , 21.6	78.3 , 12.1	111.4 , 20.8
left	21.0 , 78.5	20.9 , 21.6	20.9 , 50.6	*	20.9 , 55.7
right	76.1 , 77.6	78.3 , 21.1	*	77.1 , 50.1	77.8 , 33.8
middle	-40.7 , 79.5	111.4 , 20.8	20.9 , 55.7	77.8 , 33.8	46.7 , 45.8

Eq. 7.1 results on Table 7.2, where values marked with (*) correspond to undefined fields because of the absence of samples to compare with (e.g. no "left-right" sample).

Table 7.2: RSS results in x, y when test dataset samples are compared with the basic prediction algorithm output.

	top (t)	bottom (b)	left (l)	right (r)	middle (m)
t	105.8 , 37.1	*	**16.5** , **19.9**	**9.4** , **7.2**	342.8 , **31.6**
b	*	131.7 , 39.3	**24.5** , **28.8**	**14.2** , **16.5**	190.5 , **20.3**
l	**16.5** , **19.9**	**24.5** , **28.8**	42.5 , 130.2	*	**30.5** , 40.1
r	**9.4** , **7.2**	**14.2** , **16.5**	*	**23.7** , 85.8	**13.6** , 37.8
m	342.8 , **31.6**	190.5 , **20.3**	**30.5** , 40.1	**13.6** , 37.8	134.6 , 155.7

A closer look to the figures reveals some interesting details:

- The initial automatic sample classifications were performed dividing the 100 pixels in three equal parts, so any value lower than 33 (for "bottom" and "left"), or higher than 66 (for "top" and "right"), represents a correct parameter generation for unambiguous words. These correct values are marked in **bold** in Table 7.2.

- When the word "middle" is present, values are misplaced due to their ambiguity in terms of the used vertical and horizontal features.

- Best results are given when both words are clearly defined and unambiguous (e.g. "top-right", "bottom-left", etc.).

The same database is used to test the enhanced prediction algorithm (subsection 5.1.2), which includes the context detector and the hyperspherical shape detector. In order to check the context dependence robustness of the system, we add a flip coin probability for every generated image in the database to be described using an egocentric spatial reference ("this is my left"), or an allocentric spatial reference ("this is your left"). Both spatial feature values are similar for vertical positions, but opposite for horizontal descriptions. Results from the demonstrator point of view (egocentric) of spatial positions using "my" as descriptor are shown in Figure 7.6.

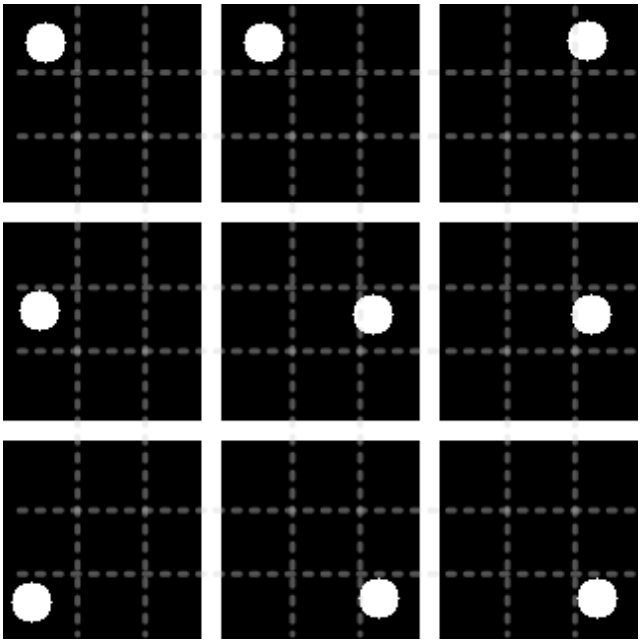

Figure 7.6: Enhanced prediction algorithm. Visual output (circle at x, y) for three-word queries. Demonstrator point of view. "My" used as descriptor (egocentric). Every image is positioned in its respective place in the figure. For instance, the upper left image corresponds to a "my-top-left" query. For a better comprehension of the results, the dotted gray lines represent the approximate margin for different areas in each square.

Results from the robot's point of view (allocentric) of spatial positions using "your" as descriptor are shown in Figure 7.7.

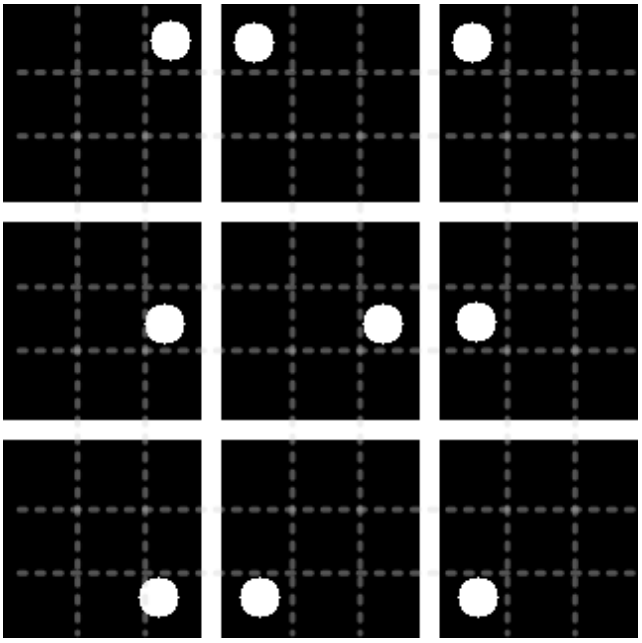

Figure 7.7: Enhanced prediction algorithm. Visual output (circle at x, y) for three-word queries. Robot point of view. "Your" used as descriptor (allocentric). Every image is positioned in its respective place in the figure. For instance, the upper left image corresponds to a "your-top-left" query. For a better comprehension of the results, the dotted gray lines represent the approximate margin for different areas in each square.

With the enhanced prediction algorithm, there is an improvement in the description containing "middle" word, as can be seen in Tables 7.3 and 7.4.

Table 7.3: Enhanced prediction algorithm. Numerical output (x, y) for a "my-word-word" query.

	top	bottom	left	right	middle
top	21.2 , 81.1	*	21.2 , 81.1	74.1 , 82.2	21.2 , 81.1
bottom	*	79.5 , 21.3	14.3 , 19.9	79.5 , 21.3	79.5 , 21.3
left	21.2 , 81.1	14.3 , 19.9	17.4 , 47.6	*	18.4 , 56.1
right	74.1 , 82.2	79.5 , 21.3	*	78 , 38.5	76.6 , 54.7
middle	21.2 , 81.1	79.5 , 21.3	18.4 , 56.1	76.6 , 54.7	76.6 , 54.7

A great improvement comes from the hyperspherical shape detector deciding to use the word "middle" as a hypersphere. The additional capacity of working with possessive adjectives ("my" and "your") is provided by the context detector mechanism of the enhanced prediction algorithm.

Table 7.4: Enhanced prediction algorithm. Numerical output (x, y) for a "your-word-word" query.

	top	bottom	left	right	middle
top	16.8 , 81	*	84 , 82.5	16.8 , 81	16.8 , 81
bottom	*	19.8 , 21.1	78.7 , 21.3	19.8 , 21.1	19.8 , 21.1
left	84 , 82.5	78.7 , 21.3	81.1 , 48.5	*	81.2 , 49.8
right	16.8 , 81	19.8 , 21.1	*	18.4 , 46.7	18.3 , 50.8
middle	16.8 , 81	19.8 , 21.1	81.2 , 49.8	18.3 , 50.8	81.2 , 49.8

7.3.2 Real Test

Once tested synthetically, the experiment was taken one step further by trying the experiment with a real robot. The experiments were performed with UC3M's full-sized humanoid robot, Teo [15] (see Figure 7.8), equipped with a Primesense Kinect depth and RGB camera sensor for vision, and a microphone and speaker for speech. The objective is to teach Teo spatial 3D references (X, Y, Z) by showing it a visual marker, and spoken interaction. After some initial training, the robot can be asked to point to a certain position in space through a spoken query. The combinations of words may have not been heard by the robot before, such as pointing to the "front-right" having learned "front" and "right" separately.

To create the population of training samples, a human operator shows a colored visual marker to the robot, and, at the same time, describes the sample. The words used to describe the sample are all the words used in the previous experiment, plus two additional words that represent the depth variable ("front" and "back"). Computer Vision uses the 2D method (1) layer B minus layer G explained in subsection 4.1.1, and additionally uses the Kinect sensor's depth measurements. CMU Pocket Sphinx [118] is used for the Speech Recognition component. The Grounding Core links the user's words with the spatial characteristics of the marker. Figure 7.9 depicts the real process, while Figure 4.2 depicts what the robot actually perceives.

After the training phase, human operator asks Teo, using the Speech Recognition component, to point to a specific position. The robot obtains a

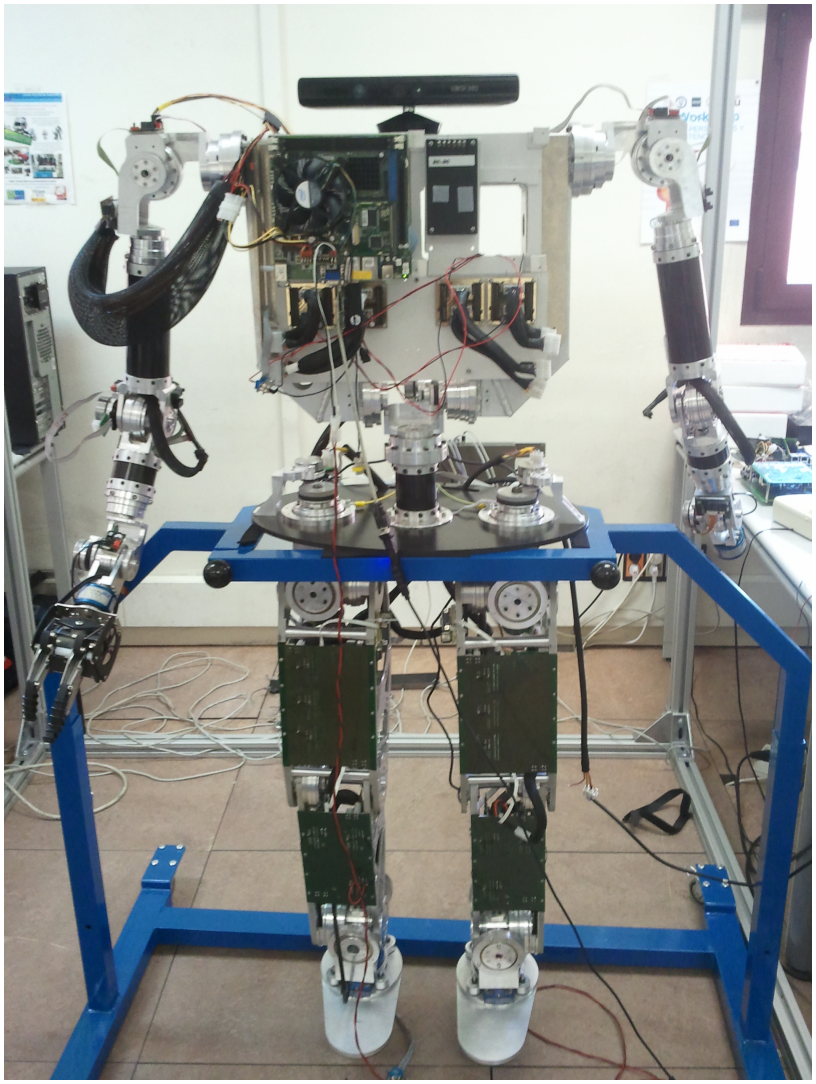

Figure 7.8: Teo is a full humanoid robot from Robotics Lab research group, Universidad Carlos III de Madrid.

representative set of coordinates, and moves its arm to this position (Figure 7.10). Some comments can be made about this experiment.

- The system can generalize a word with as little as 2 training samples in the 2D case, and 3 training samples in the 3D case.

- The component used for Speech Recognition requires a pre-defined corpus of words to recognize. We have fed the corpus with the words we knew would be used to describe spatial positions. This temporal hack currently limits the possibilities of spontaneous teaching and learning.

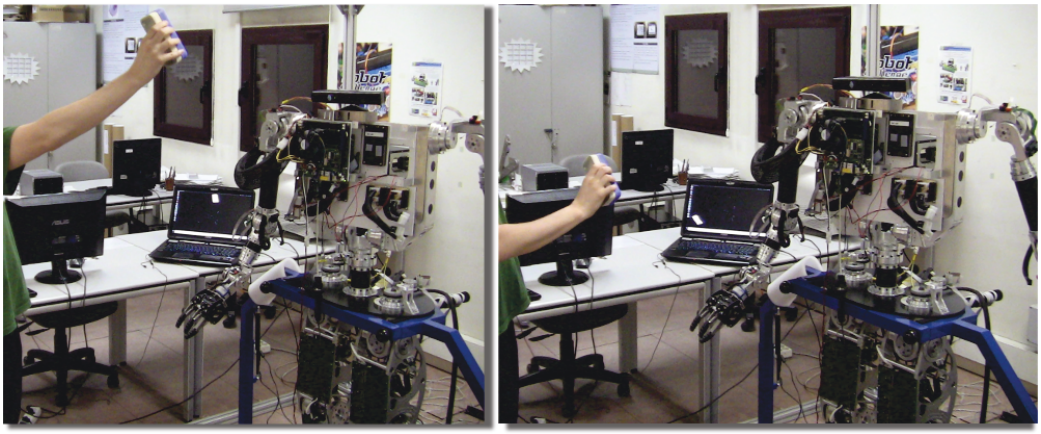

Figure 7.9: Human operator teaching spatial positions to Teo. In the laptop on the background, the streaming object segmentation is shown.

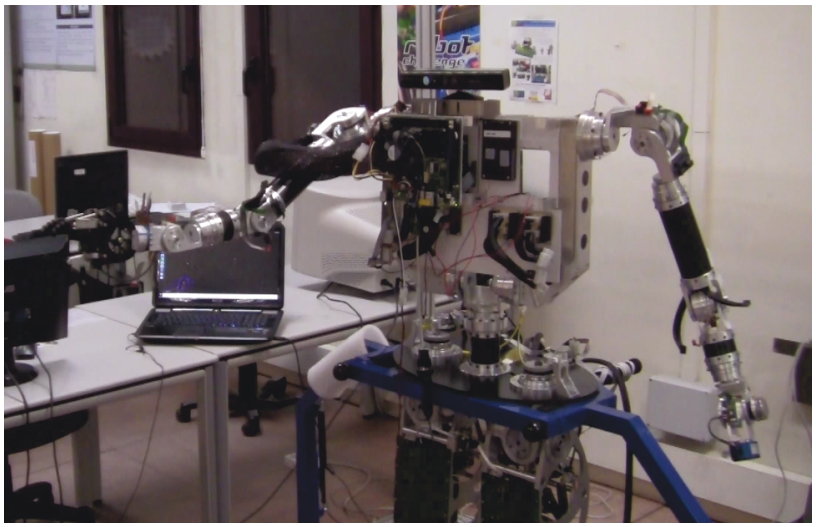

Figure 7.10: Teo pointing in response to a "your-front-right" query.

7.4 Execution Core: Waxing

To test the Execution Core, the action chosen to be learned and executed was to "wax". The features used were again purely spatial. Figure 6.1 depicts the generalized trajectory given 10 real world repetitions of the action.

Evolutionary Computation (EC) was performed in the OpenRAVE-based [117] simulated environment with a model of the humanoid robot Teo (see Figure 7.11), evolving 30 values to generate the trajectory (3 joint positions, times 10 timesteps). Fitness is evaluated when the full robot motor joint

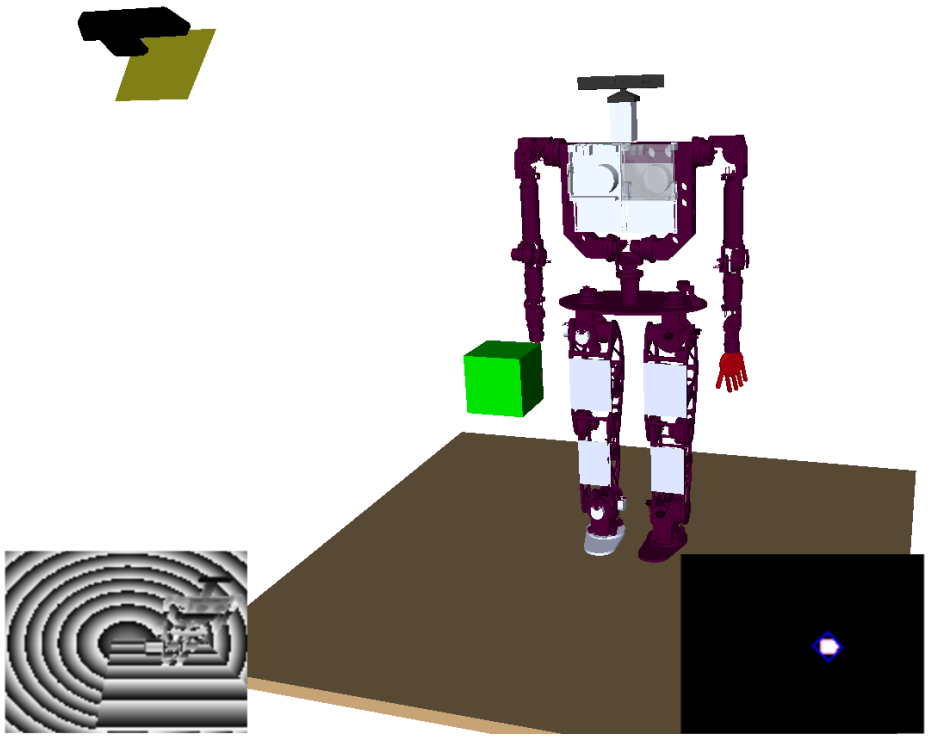

Figure 7.11: The plot shows the experimental scenario with the robot, the object (green) and the Kinect camera. The bottom left square is the Kinect depth map and the bottom right square shows the object segmentation.

trajectory is executed, by measuring the spatial features of an object (the green box) using a simulated Kinect camera in the environment. The reference action and the measured one are compared using CGDA recognition, and the score of discrepancy is used as the fitness value to minimize. After convergence, the winner action is executed. The performance of the winner action compared to the generalized reference is depicted in Figure 7.12.

The resulting trajectories, when executed on the robot, are not human natural movements. This is an expected behavior, as the aim of the experiment was to replicate object states during execution, and not the performance of human-like movements. Note that in this thesis we have only included spatial features as targets for the Execution Core. However, the CGDA infrastructure allows more complex tasks such as painting based only on color features (included in unpublished material, submitted for review).

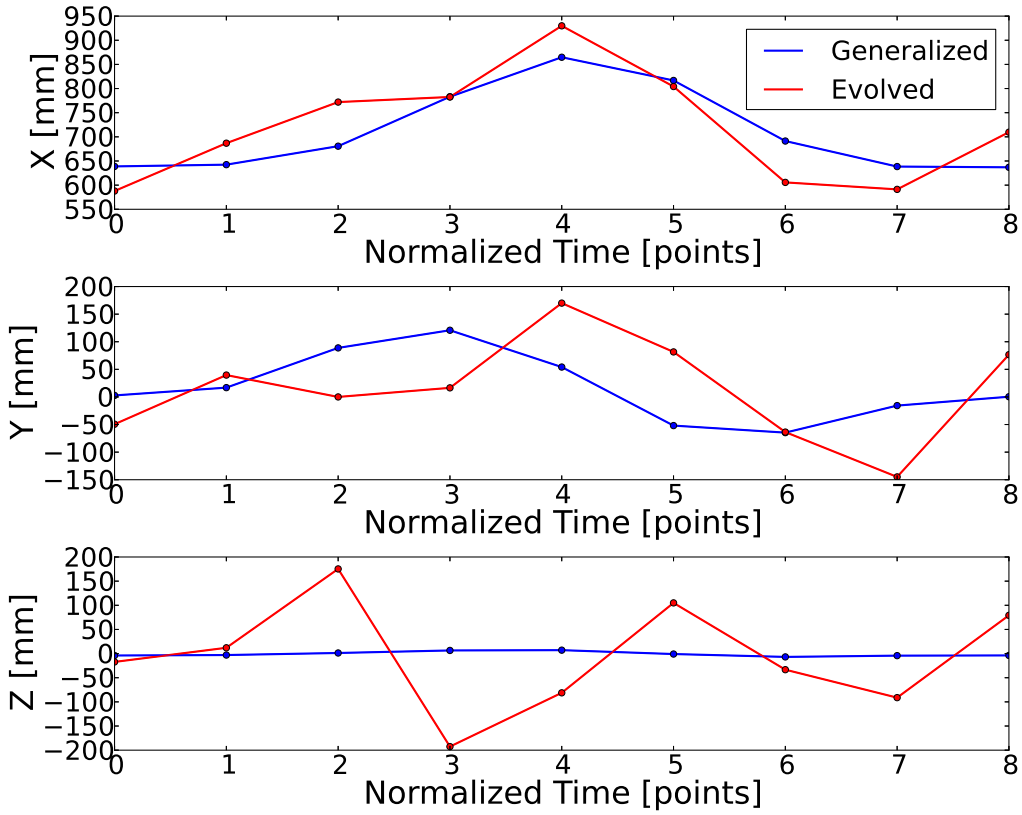

Figure 7.12: Unidimensional temporal plots of the generalized action (blue), and the object feature space trajectory resulting from the execution of the EC winner robot motor joint trajectory (red). The Z dimension gives the worst results, the system was not able to reduce the error in this dimension.

7.5 The Token Test

As a final system validation, and inspired by the procedure followed in [119], the "Token Test" [111] was adapted to be used with a robot. Instead of the original Token Test, we used a simplified version, the "Shortened Token Test" [120]. The Token Test was created as an effective tool to diagnose aphasia. The shortened version differs from the original in that the number of items is limited to 36, and that the commands have been reduced in several parts of the experiment. This version includes one part more than the original test. We have slightly modified the material of the test: squares are used

instead of rectangles, and black tokens are used instead of blue tokens. The token set is composed by 20 items, where 10 are circles and 10 are squares. There are 5 large circles and 5 large squares, and 5 small circles and 5 small squares. Each set of 5 items includes black, white, red, yellow and green shapes. Figure 7.13 depicts a possible setup when all of the tokens are used.

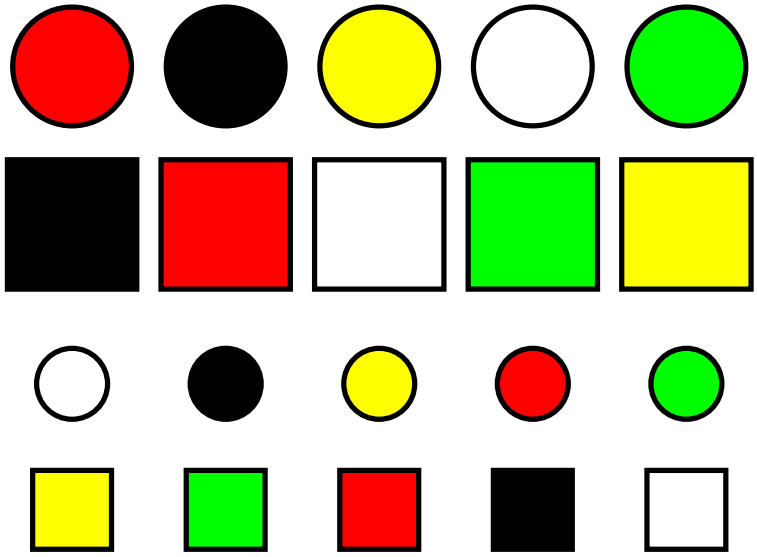

Figure 7.13: Tokens used in the Token Test. There are 2 shapes (squares and circles) and 2 sizes (small and large). The tokens are colored in 5 colors (red, black, yellow, white and green). All size-color-shape combinations can be formed (e.g. small-red-circle, large-black-square, etc.).

The experiment is divided into the following six parts.

- **Part 1:** All 20 tokens are displayed, as in Figure 7.13.

    1. Touch a circle
    2. Touch a square
    3. Touch a yellow token
    4. Touch a red one
    5. Touch a green one
    6. Touch a black one
    7. Touch a white one

- **Part 2:** The small tokens are removed.

  8. Touch the yellow square
  9. Touch the black circle
  10. Touch the green circle
  11. Touch the white square

- **Part 3:** The small tokens are placed again.

  12. Touch the small white circle
  13. Touch the large yellow square
  14. Touch the large green square
  15. Touch the small black circle

- **Part 4:** The small tokens are removed.

  16. Touch the red circle and the green square
  17. Touch the yellow square and the black square
  18. Touch the white square and the green circle
  19. Touch the white circle and the red circle

- **Part 5:** The small tokens are placed again.

  20. Touch the large white circle and the small green square
  21. Touch the small black circle and the large yellow square
  22. Touch the large green square and the large red square
  23. Touch the large white square and the small green circle

- **Part 6:** The small tokens are removed.

  24. Put the red circle on the green square
  25. Touch the black circle with the red square

26. Touch the black circle and the red square
27. Touch the black circle or the red square
28. Put the green square away from the yellow square
29. If there is a blue circle, touch the red square
30. Put the green square next to the red circle
31. Touch the squares slowly and the circles quickly
32. Put the red circle between the yellow square and the green square
33. Touch all the circles, except the green one
34. Touch the red circle - no - the white square
35. Instead of the white square, touch the yellow circle
36. In addition to touching the yellow circle, touch the black circle

Regarding score, in the original paper, they give 1 point for a correct performance on the first presentation and 0.5 point if the performance is correct only on the second presentation. We have preserved this evaluation and give 1 point if the first indicated result is the correct, 0.5 if the correct solution is the second one and 0 for the rest of the cases. Beyond question 23, the test evaluates motor actions like "put on", "put between", and language logic like "instead" and "if". As our system is not designed for this behavior, nor had it been the aim of the development, we only check up to question 23. Therefore, the results are within the range of $[0, 23]$.

Two different parameters of the enhanced prediction algorithm presented in subsection 5.1.2 must be fixed, as results vary depending on them. The Θ_{min} parameter is the threshold to consider the eigenvalues as equal (forming a hypersphere) or different enough (forming a hyperplane). The ω_{max} parameter is the maximum distance allowed for hierarchical clustering to consider a set of points as a cluster. Figure 7.14 depicts a schematic representation of the settings of these parameters in a 2-dimensional space.

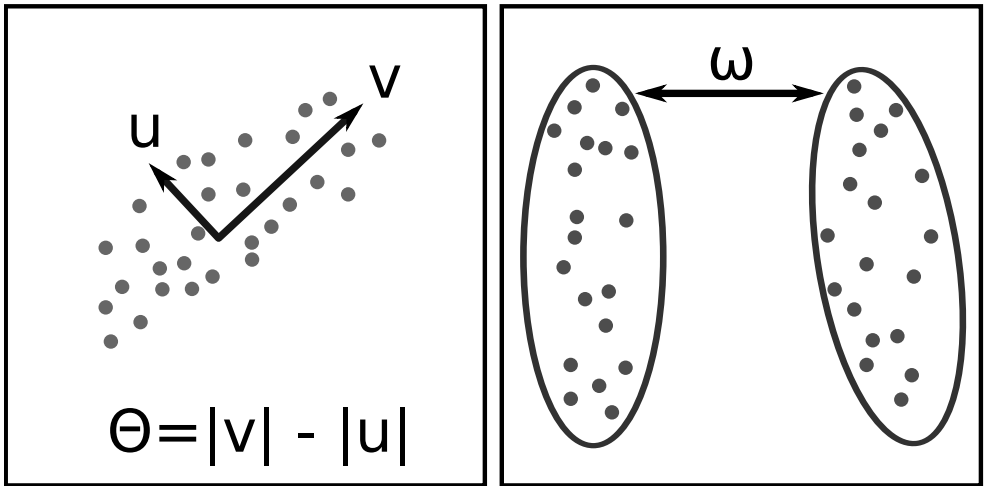

Figure 7.14: Example of parameter setting in a 2-dimensional space.

Teo, the humanoid robot, is trained by showing each printed token at random distances one single time, provided individual descriptions with three words (e.g. "red-large-square", "small-yellow-circle", etc.). During each step of the training, the Computer Vision component detects the contour of the token (through Canny edge detection) and analyses 5 different features for prediction: area/distance ratio, rectangularity, and mean hue, saturation and value. All of the 20 tokens are described during the training except for one, the "small black circle", left out to provide a proof-of-concept demonstration of the RIS creative inference capabilities.

Teo is then placed in front of the full printed Token Test, as seen in Figure 7.15. The testing phase consists in performing the 23 test queries of the test described above. The Speech Recognition component enhances realism. Upon object recognition, Teo's arm moves to the token, using its centroid position as a spatial feature for reference. The system's results varying the parameters Θ_{min} and ω_{max} are presented on Table 7.5 and 7.6. As seen on these results, our system can correctly identify all of the tokens with the particular set of parameters ($\Theta_{min} : 0$, $\omega_{max} : 0.75$). The low value of Θ_{min} forces all of the point clouds to be extended as hyperplanes. In the training and test space, this makes sense, as there are no words that should

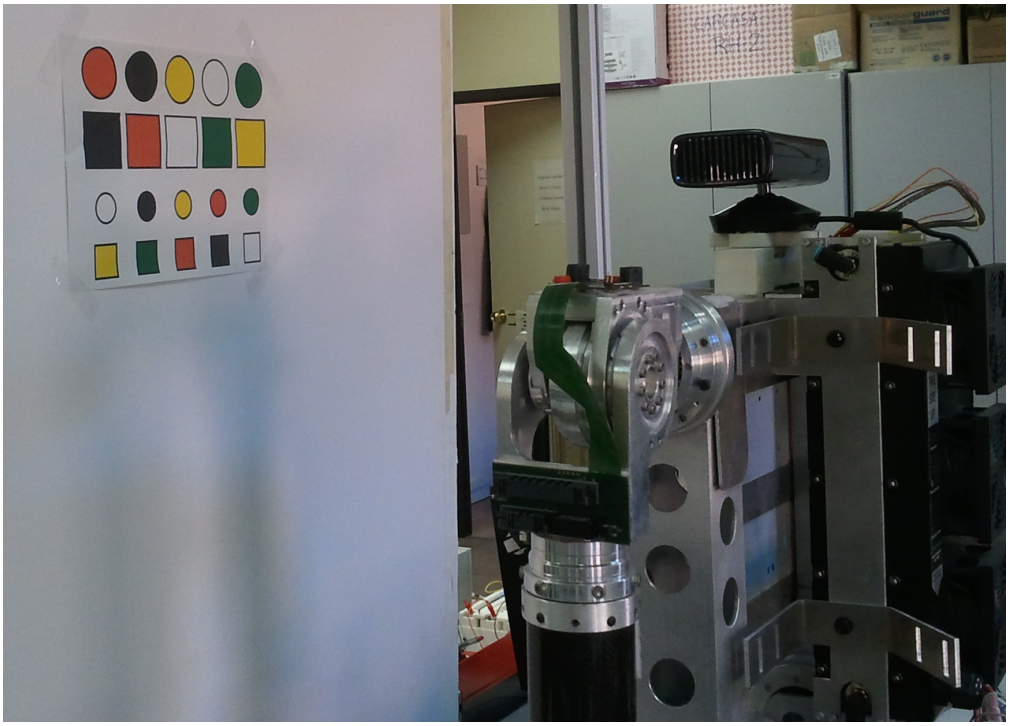

Figure 7.15: Experimental setup with the humanoid Teo for the Token Test.

actually be treated as hyperspheres. On the other hand, our assumption on an improvement of the results with a lower ω_{max} is that it serves to filter outliers that appear during the training and recognition phases with real sensor data.

Table 7.5: Token Test results in function of Θ_{min} and ω_{max}.

	1	2	3	4	5	6	7	8	9	10	11	12	13
$\Theta_{min}: 0$, $\omega_{max}: 1.25$	1	1	1	1	1	1	1	0	1	1	1	1	0
$\Theta_{min}: 0.5$, $\omega_{max}: 1.25$	1	1	1	1	1	1	1	0	0	0	0	0	0
$\Theta_{min}: 0$, $\omega_{max}: 0.75$	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 7.6: Token Test results in function of Θ_{min} and ω_{max}.

	14	15	16	17	18	19	20	21	22	23
$\Theta_{min}: 0$, $\omega_{max}: 1.25$	0.5	1	1	1	0	1	1	0	0	0
$\Theta_{min}: 0.5$, $\omega_{max}: 1.25$	1	0	0	0	0	0	0	0	0	0
$\Theta_{min}: 0$, $\omega_{max}: 0.75$	1	1	1	1	1	1	1	1	1	1

Chapter 8

Results and Conclusions

This thesis has presented a system for providing robots with imagination, generating mental models through object feature inference and semantic descriptions, and executing actions based on consequences. The system has been developed to learn the vocabulary of an end-user, grounded in sensory information. Barsalou predicted that cognitive science will increasingly witness the integration of its different paradigms, with competition between them decreasing [39]. Pezzulo, Barsalou et al. very recently proposed the alliance between grounded cognition and computational modeling as a unified view of cognition, and emphasize the importance of using the methodology of Cognitive Robotics [121]. The system presented in this thesis learns and manipulates values of features extracted from objects that are present in the environment, and is able to execute actions that transform the environment, providing it with a *grounded* and *embodied* nature.

8.1 Progress Beyond the State of the Art

A novel system has been presented in this thesis, and its components have been presented. These components are grouped in three blocks that aim to overcome shortcomings that have been identified within classical robot programming and certain modern approaches, which are the following.

A. **Perception.** A framework that allows linking physical characteristics of objects and the words used to describe them has been developed. It includes scalable mechanisms that provide grounded acquisition of symbols that the cognitive processes can manipulate to perform actions in the real world. Grounded symbol acquisition can be performed in concurrence with inference and execution.

B. **Inference.** Inference mechanisms that allow a robot to work with combinations of words used to describe objects, even if these words have never been previously taught together have been enabled. Both a basic word-based feature prediction algorithm an enhanced version have been presented. Predicted object features are used for directly performing object recognition, and also to generate novel robot mental images and as part of the Robot Imagination process.

C. **Execution.** Through the use of robot task execution based on action consequences, in RIS robots can act according to the effect desired on an object, instead of relying on pre-programmed trajectories alone. The core of execution uses this paradigm to close the perceptual loop with the environment.

8.2 Future Lines of Research

The author considers it vital to continue with the evolution of the research performed in this thesis. The following is a list of elements that have been identified to help further enhance the presented developments.

1. **Origin of data.** Sensory information of various nature such as haptic and rugosity should be involved in the inference algorithm. Additionally, web-based search mechanisms such as the developed "Onewebmicrodata" platform (an undergraduate student project development of a

web crawler based on HTML5 microdata and camera Exif parameters) [122] should be integrated.

2. **Data representation.** The YUV color space seems to be more human-inspired than RGB or HSV. Utterances could be used instead of words. As mentioned in subsection 4.1.3, SIFT [99] and SURF [100] descriptors have not been used, but this could be attempted without any modification of the presented algorithms.

3. **Techniques from which inference and execution may benefit.** Feature selection (e.g. via LASSO) is considered within the immediate future of RIS. Similarly, Gaussian Mixture Models instead of PCA seem to promise generalization without over-generalization, but alternatives such as manifolds or non-linear PCA (e.g. with Neural Networks) will also be studied. In fact, Neural Networks as a general approach seems more bio-inspired. The author has plans to integrate execution with force control, e.g. integration with Dynamic Motion Primitives.

4. **Desired functionalities and specifications.** Further extensions should be able to manage vectorial information (e.g. texture, patterns) and dynamic properties (e.g. motion). Multi-modal imagery (e.g. auditory output) would also be interesting. As a final general guideline, future work should include minimizing the number of parameters that need tuning, such as those of clustering mechanisms and the evolutionary computational frameworks.

The author envisions a future where robot imagination systems are used for advanced tasks and beyond: learning and action grounded in perception, action, and perhaps even emotion. A first step in this direction has been robotic recognition and action upon objects that have never been perceived before, and further steps will be given towards integration in everyday domestic environments. Regarding discussions deep into the future, the time has come for humans to predict and robots to imagine.

Bibliography

[1] Deb Roy. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1):170–205, 2005.

[2] Deb Roy. Learning visually grounded words and syntax for a scene description task. *Computer Speech & Language*, 16(3-4):353–385, Jul 2002.

[3] Alan D. Baddeley. The episodic buffer: a new component of working memory? *Trends in cognitive sciences*, 4(11):417–423, 2000.

[4] Margaret A Boden. *The creative mind: Myths and mechanisms*. Routledge, 2004.

[5] Paul Thagard and Terrence C Stewart. The aha! experience: Creativity through emergent binding in neural networks. *Cognitive science*, 35(1):1–33, 2011.

[6] E. Bruce Goldstein. *Sensation and Perception*. Cengage Learning, 8th edition, 2009.

[7] Luca Iocchi, Javier Ruiz-del Solar, and Tijn Zant. Domestic service robots in the real world. *Journal of Intelligent & Robotic Systems*, 66(1):183–186, 2012.

[8] Yuka Ariki, Tetsunari Inamura, and Jun Morimoto. Learning humanoid robot interface to generate many-dof movements from fewer-dof command inputs. In *Workshop/Tutorial on Semantics, Identification and*

Control of Robot-Human-Environment Interaction at IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 2013.

[9] Juan G. Victores, Alberto Jardón, Santiago Morante, Martin F. Stoelen, Santiago Martínez, and Carlos Balaguer. Interacción humano-robot a través de interfaces en la nube. In *Robocity2030 9th Workshop Robots colaborativos e interaccion humano-robot*, pages 75–92, 2011.

[10] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. "give me the red can": Assistive robot task creation through multi-modal interaction. In *International Congress on Design, Research Networks, and Technology for all - DRT4ALL*, Madrid, 2013.

[11] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. Creación de tareas de asistencia robótica mediante la interacción multimodal. In *Congreso Iberoamericano de Tecnologias de Apoyo a la Discapacidad - Iberdiscap*, Santo Domingo, Republica Dominicana, 2013.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2nd edition, 2009.

[13] Sylvain Calinon. *Robot programming by demonstration*. EPFL Press, 2009.

[14] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, Jun 1999.

[15] Santiago Martínez, Concepción A. Monje, Alberto Jardón, Paolo Pierro, Carlos Balaguer, and D. Muñoz. Teo: Full-size humanoid robot design powered by a fuel cell system. *Cybernetics and Systems*, 43(3):163–180, 2012.

[16] Harold Cohen. The further exploits of aaron, painter. *Stanford Humanities Review*, 4(2):141–158, 1995.

[17] Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. Mental imagery for a conversational robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics.*, 34(3):1374–1383, 2004.

[18] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems : Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation*, 11(2):151–180, 2007.

[19] Francisco J. Varela. Whence perceptual meaning? a cartography of current ideas. In *Understanding origins*, pages 235–263. Springer, 1992.

[20] Craig Schlenoff, Edson Prestes, Raj Madhavan, Paulo Goncalves, Howard Li, Stephen Balakirsky, Thomas Kramer, and Emilio Migueláñez. An ieee standard ontology for robotics and automation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1337–1342. IEEE, 2012.

[21] Douglas B. Lenat and Ramanathan V. Guha. *Building large knowledge-based systems; representation and inference in the Cyc Project*. Addison-Wesley, 1989.

[22] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical report, W3C, 2004.

[23] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1284–1289. Ieee, May 2012.

[24] Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

[25] C. Burgess, K. Livesay, and K. Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2-3):211–257, 1998.

[26] John R. Anderson. Act: A simple theory of complex cognition. *Cognitive modeling*, 49, 1995.

[27] John E. Laird. *The Soar cognitive architecture*. MIT Press, 2012.

[28] Stevan Harnad. Category induction and representation. *Cognition and Brain Theory*, 5:535–565, 1987.

[29] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990.

[30] John R. Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424, 1980.

[31] R. Katarzyniak. The language grounding problem and its relation to the internal structure of cognitive agents. *Journal of Universal Computer Science*, 11(2):357–374, 2005.

[32] Luc Steels and Manfred Hild. *Language grounding in robots*. Springer, 2012.

[33] Michael Ramscar and Daniel Yarlett. Semantic grounding in models of analogy: an environmental approach. *Cognitive Science*, 27(1):41–71, 2003.

[34] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In

Proceedings of 7th International Semantic Web Conference (ISWC), pages 615–631. Springer, 2008.

[35] Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

[36] José M. Musacchio. The ineffability of qualia and the word-anchoring problem. *Language Sciences*, 27(4):403–435, 2005.

[37] Rong Zhao and William I. Grosky. Narrowing the semantic gap-improved text-based web document retrieval using visual features. *IEEE Transactions on Multimedia*, 4(2):189–200, 2002.

[38] Changhu Wang, Lei Zhang, and Hong-Jiang Zhang. Learning to reduce the semantic gap in web image retrieval and annotation. In *Proceedings of 31st annual ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 355–362. ACM, 2008.

[39] Lawrence W. Barsalou. Grounded cognition: past, present, and future. *Topics in Cognitive Science*, 2(4):716–724, 2010.

[40] Jaehun Joo. Adoption of semantic web from the perspective of technology innovation: A grounded theory approach. *International Journal of Human-Computer Studies*, 69(3):139–154, 2011.

[41] Silvia Coradeschi, Amy Loutfi, and Britta Wrede. A short review of symbol grounding in robotic and intelligent systems. *KI - Künstliche Intelligenz*, 27(2):129–136, Mar 2013.

[42] Alan M. Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950.

[43] Gerd Herzog and Peter Wazinski. Visual translator: Linking perceptions and natural language descriptions. *Artificial Intelligence Review*, 8(2):175–187, 1994.

[44] Luc Steels. Language games for autonomous robots. *Intelligent Systems, IEEE*, 16(5):16–22, 2001.

[45] Deb Roy. A trainable visually-grounded spoken language generation system. In *Proceedings of ICSLP*. Citeseer, 2002.

[46] Deb Roy. Grounding words in perception and action: computational insights. *Trends in cognitive sciences*, 9(8):389–396, 2005.

[47] Shiwali Mohan, Aaron H. Mininger, James R. Kirk, and John E. Laird. Learning grounded language through situated interactive instruction. In *Robots Learning Interactively from Human Teachers (AAAI Fall Symposium Series)*, pages 30–37, 2012.

[48] Shiwali Mohan, Aaron H. Mininger, James R. Kirk, and John E. Laird. Acquiring grounded representations of words with situated interactive instruction. *Advances in Cgnitive Systems*, 2:113–130, 2012.

[49] Kenneth James Williams Craik. *The nature of explanation*. Cambridge University Press, 1943.

[50] Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6. Harvard University Press, 1983.

[51] Brian J. Scholl. Object persistence in philosophy and psychology. *Mind & Language*, 22(5):563–591, 2007.

[52] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *AAAI*, volume 93, pages 689–695. Citeseer, 1993.

[53] Robert Burke, Damian Isla, Marc Downie, Ivanov Yuri, and Bruce Blumberg. Creature smarts: The art and architecture of a virtual brain. In *Game Developer's Conference*, 2001.

[54] Kai-Yuh Hsiao, Nikolaos Mavridis, and Deb Roy. Coupling perception and simulation: Steps towards conversational robotics. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 928–933. IEEE, 2003.

[55] James Jerome Gibson. *The ecological approach to visual perception*. Psychology Press, 1986.

[56] Stephen M. Kosslyn. Mental images and the brain. *Cognitive Neuropsychology*, 22(3-4):333–347, 2005.

[57] David G. Pearson, Catherine Deeprose, Sophie Wallace-Hadrill, Stephanie Burnett Heyes, and Emily A. Holmes. Assessing mental imagery in clinical psychology: a review of imagery measures and a guiding framework. *Clinical psychology review*, 33(1):1–23, 2013.

[58] Stephen Michael Kosslyn. *Image and mind*. Harvard University Press, 1980.

[59] Robert H. Logie. *Visuo-spatial working memory*. Psychology Press, 1995.

[60] Alan D. Baddeley and Graham J. Hitch. Working memory. volume 8 of *Psychology of Learning and Motivation*, pages 47–89. Academic Press, 1974.

[61] Kim Binsted, Helen Pain, and Graeme Ritchie. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition*, 5(2):305–354, 1997.

[62] Tomoaki Nakamura, Takayuki Nagai, and Naoto Iwahashi. Multimodal object categorization by a robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2415–2420. Ieee, Oct 2007.

[63] Nikolaos Mavridis and Deb Roy. Grounded situation models: Where words and percepts meet. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3–3. Ieee, October 2006.

[64] Damian A Isla and Bruce M Blumberg. Object persistence for synthetic creatures. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pages 1356–1363. ACM, 2002.

[65] Sylvain Calinon, Florent D'halluin, Eric L. Sauser, Darwin G. Caldwell, and Aude G. Billard. Learning and reproduction of gestures by imitation. *Robotics & Automation Magazine*, 17(2):44–54, 2010.

[66] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2769–2774. IEEE, 2004.

[67] Sylvain Calinon and Aude Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of 22nd International Conference on Machine learning*, pages 105–112. ACM, 2005.

[68] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of ACM/IEEE International Conference on Human-robot Interaction*, pages 255–262. ACM, 2007.

[69] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):286–298, 2007.

[70] Kartick Subramanian and Sundaram Suresh. Human action recognition using meta-cognitive neuro-fuzzy inference system. *International journal of neural systems*, 22(06), 2012.

[71] Daniel Stephen Chivers. *Human Action Recognition by Principal Component Analysis of Motion Curves*. PhD thesis, Wright State University, 2012.

[72] Stefan Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer, 2006.

[73] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[74] Chrystopher L. Nehaniv and Kerstin Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In *Interdisciplinary Approaches to Robot Learning*, volume 24, page 136. World Scientific, 1999.

[75] Harold Bekkering, Andreas Wohlschlager, and Merideth Gattis. Imitation of gestures in children is goal-directed. *The Quarterly Journal of Experimental Psychology: Section A*, 53(1):153–164, 2000.

[76] Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi, and Giacomo Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119(2):593–609, 1996.

[77] Magali J. Rochat, Fausto Caruana, Ahmad Jezzini, Ludovic Escola, Irakli Intskirveli, Franck Grammont, Vittorio Gallese, Giacomo Rizzolatti, and Maria Alessandra Umiltà. Responses of mirror neurons in area f5 to hand and tool grasping observation. *Experimental brain research*, 204(4):605–16, Aug 2010.

[78] James M. Kilner, Alice Neal, Nikolaus Weiskopf, Karl J. Friston, and Chris D. Frith. Evidence of mirror neurons in human inferior frontal gyrus. *The Journal of Neuroscience*, 29(32):10153–10159, 2009.

[79] Sylvain Calinon, Florent Guenter, and Aude Billard. Goal-directed imitation in a humanoid robot. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 299–304. IEEE, 2005.

[80] Santiago Morante, Juan G. Victores, Alberto Jardón, and Carlos Balaguer. Action effect generalization, recognition and execution through continuous goal-directed actions. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

[81] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. Semantic action parameter inference through machine learning methods. In *Robocity2030 12th Workshop Robotica Cognitiva*, 2013.

[82] Matthew Johnson and Yiannis Demiris. Abstraction in recognition to solve the correspondence problem for robot imitation. In *Proceedings of Towards Autonomous Robotic Systems*, pages 63–70. Springer, 2004.

[83] Juan G. Victores, Santiago Morante, Alberto Jardón, and Carlos Balaguer. Towards robot imagination through object feature inference. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013. IEEE.

[84] Serge Thill, Daniele Caligiore, Anna M Borghi, Tom Ziemke, and Gianluca Baldassarre. Theories and computational models of affordance and mirror systems: An integrative review. *Neuroscience and biobehavioral reviews*, 37(3):491–521, January 2013.

[85] Alva Noë. *Action in perception*. MIT press, 2004.

[86] John Robert Stewart, Olivier Gapenne, and Ezequiel A. Di Paolo. *Enaction: Toward a new paradigm for cognitive science*. MIT Press, 2010.

[87] Jesse Prinz. Putting the brakes on enactive perception. *Psyche*, 12(1):1–19, 2006.

[88] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2006.

[89] Robert Sternberg. *Cognitive psychology*. Cengage Learning, 6th edition, 2012.

[90] Ian E. Gordon. *Theories of visual perception*. Psychology Press, 2004.

[91] Larry R. Squire. *Fundamental neuroscience*. Academic Press, 4th edition, 2013.

[92] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.

[93] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. Basic Books, Inc., New York, NY, USA, 1993.

[94] Juan G. Victores, Alberto Jardón, Martin F. Stoelen, Santiago Martínez, and Carlos Balaguer. Asibot assistive robot with vision in a domestic environment. In *Robocity2030 7th Workshop Visión en Robótica*, pages 61–74, 2010.

[95] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

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

[97] Zoltan Csaba Marton, Radu B. Rusu, and Michael Beetz. On fast surface reconstruction methods for large and noisy datasets. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

[98] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[99] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[100] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[101] Paul Bloom. *How Children Learn the Meanings of Words*. MIT Press, 2002.

[102] Thomas Fober, Gerghei Glinca, Gerhard Klebe, and Eyke Hullermeier. Superposition and alignment of labeled point clouds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1653–1666, 2011.

[103] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[104] Yogesh Rathi, Samuel Dambreville, and Allen Tannenbaum. Comparative analysis of kernel methods for statistical shape learning. In ReinhardR. Beichel and Milan Sonka, editors, *Computer Vision Approaches to Medical Image Analysis*, volume 4241 of *Lecture Notes in Computer Science*, pages 96–107. Springer Berlin Heidelberg, 2006.

[105] Age K. Smilde, Jeroen J. Jansen, Huub C.J. Hoefsloot, Robert-Jan A.N. Lamers, Jan Van Der Greef, and Marieke E. Timmerman. Anova-simultaneous component analysis (asca): a new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13):3043–3048, 2005.

[106] John C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, pages 54–64, 1969.

[107] Adnan El-Nasan, Sriharsha Veeramachaneni, and George Nagy. Word discrimination based on bigram co-occurrences. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 149–153. IEEE, 2001.

[108] D. Whitley and J. Kauth. *GENITOR: A different genetic algorithm.* Colorado State University, Department of Computer Science, 1988.

[109] William H. Press. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

[110] Thomas Albrecht. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, 2009.

[111] Ennio De Renzi and Luigi A. Vignolo. Token test: A sensitive test to detect receptive disturbances in aphasics. *Brain: a journal of neurology*, 1962.

[112] Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45, 2008.

[113] Santiago Morante. Interfaz y librería para visión artificial, navegación y seguimiento en robótica. Technical report, Universidad Carlos III de Madrid, Leganés, 2012.

[114] Radu B. Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011.

[115] William J Schroeder, Lisa Sobierajski Avila, and William Hoffman. Visualizing with vtk: a tutorial. *Computer Graphics and Applications, IEEE*, 20(5):20–27, 2000.

[116] Alberto Jardón, Juan G. Victores, Santiago Martínez, Antonio Giménez, and Carlos Balaguer. Personal autonomy rehabilitation in home environments by a portable assistive robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):561–570, 2012.

[117] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, Aug 2010.

[118] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W. Black, Mosur Ravishankar, and Alex I. Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages I–I. IEEE, 2006.

[119] Nikolaos Mavridis and Deb Roy. Grounded situation models for robots: Bridging language, perception, and action. In *AAAI-05 workshop on modular construction of human-like intelligence*, 2005.

[120] Ennio De Renzi and Pietro Faglioni. Normative data and screening power of a shortened version of the token test. *Cortex*, 14(1):41–49, 1978.

[121] Giovanni Pezzulo, Lawrence W Barsalou, Angelo Cangelosi, Martin H Fischer, Ken McRae, Michael J Spivey, et al. Computational grounded

cognition: a new alliance between grounded cognition and computational modeling. *Frontiers in psychology*, 3:612–612, 2013.

[122] Julián Caro Linares. *Onewebmicrodata: Una araña web de extración de microdatos HTML5 y parámetros Exif*. Trabajo de Fin de Grado. Universidad Carlos III de Madrid, 2014.

Short Biography

Juan G. Victores was born in Frederick (Maryland, USA). He obtained the Industrial Engineer degree from Universidade de Vigo in 2008, and MSc. in Robotics and Automation at Universidad Carlos III de Madrid (UC3M) in 2010. He participated in the FP6 EU robotics in construction project TunConstruct (2007–2008), and is currently involved in FP7 EU robotics in construction project Robinspect (2013–2016). He has been an active researcher for the ASIBOT project in assistive robotics, and has published over 26 conference articles, 4 first-quartile journal papers and 3 book chapters, a patent, and several artistic works, obtaining over 62 citations (h-index 4, and i10-index 2). He currently works with the humanoid robot Teo with interest in cognition that is grounded in perception, action and emotion. Research stays include 3 months at the Robotics, Brain and Cognitive Sciences department at IIT (Genova, Italy). Juan additionally contributes to several open source projects (e.g. YARP and Robot Devastation at GitHub), and is the leader of the ASROB robotics student society at UC3M.