Universidad
Carlos III de Madrid

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

TESIS DE MÁSTER

GENERALIZATION AND INFERENCE OF OBJECTS AND OBJECT-CENTERED ACTIONS

Autor: Santiago Morante Cendrero
Tutor: Prof. Carlos Balaguer Bernaldo de Quirós
Director: Juan Carlos González Víctores

MÁSTER OFICIAL EN
ROBÓTICA Y AUTOMATIZACIÓN

LEGANÉS, MADRID
SEPTIEMBRE 2013

UNIVERSIDAD CARLOS III DE MADRID
MÁSTER OFICIAL EN ROBÓTICA Y AUTOMATIZACIÓN

El tribunal aprueba la Tesis de Máster titulada "**GENERALIZATION AND INFERENCE OF OBJECTS AND OBJECT-CENTERED ACTIONS**" realizada por **Santiago Morante Cendrero**.

Fecha: Septiembre 2013

Tribunal: _____
Dr./Dra.

_____
Dr./Dra.

_____
Dr./Dra.

"The true knowledge is not in the things, but in finding the connections between the things"

Daniel H. Wilson, Robopocalypse

Contents

Index of Tables

Index of Figures

Acknowledgments

Thanks to all the people supporting this thesis, specially Juan G. Víctores and Carlos Balaguer. Thanks to my family and friends, because of their love and comprehension. But, undoubtedly, thanks to Lur...

Resumen

Esta tesis desarrolla los algoritmos necesarios para construir las bases de un sistema de generalización e inferencia de objetos y acciones. La generalización de objetos es llevada a cabo en un espacio n-dimensional llamado espacio semántico-característico, porque aúna las percepciones del robot, en forma de valores de características, con la descripción semántica asociada al objeto. La inferencia de objetos se asume como una búsqueda de áreas relevantes, las cuales se encuentran como intersección de elementos altamente dimensionales, en la forma de modelos de palabras. La inferencia de acciones sigue los mismos principios que en los objetos, pero midiendo los cambios sólo en el objeto afectado por la acción (aproximación que hemos denominado "centrada en objetos"). Esta inferencia se lleva a cabo mediante capturas estáticas principio-fin de la acción, aplicando distintos métodos de aprendizaje máquina. Finalmente, algunas aplicaciones robóticas finales han sido desarrolladas, como el dibujo de modelos mentales con un brazo manipulador a través de algoritmos evolutivos, o el aprendizaje de vocabulario espacial por un robot humanoide.

Abstract

This thesis develops the necessary algorithms to build the basis for a generalization and inference system of objects and actions. Object generalization is performed in a n-dimensional space called semantic-feature space, because it mixes the robot perceptions, in the form of features values, with the semantic description associated with the object. Objects inference is assumed as a search of relevant areas, which are found as an intersection of high dimensional elements, in the form of keyword models. Action inference follows the same principles that objects, but measuring the changes only in the object affected by the action (approach we have named "object-centered"). This inference is carried out as static start-end snapshot, applying several machine learning methods. Finally, some robotic end applications have been tested, like drawing mental models with a manipulator arm through evolutionary algorithms, or the learning of spatial vocabulary by a humanoid robot.

Chapter 1

Introduction

A real artificial intelligence for robots is still a pendant task. The most we can aspire in these days is to create modules which reproduce small parts of the human cognition. Let us cite some of these modules, with high influence of human cognition, but isolated in their field of study.

In (Dominey, Metta, Nori, & Natale, 2008) they try to imitate anticipation. They use a dialog system where the human can ask the robot to execute a specific action in order to collaborate in completing a task. This dialog is recorded and the action sequence is extracted. When they perform another dialog with the robot and ask it to execute more actions, the robot searches for the same sequence of actions in its "interaction history". If a positive coincidence is returned, the robot asks the user if the current sequence is the same as found in its history. If a positive answer is given, from this moment, there is no need to explicitly ask the robot for following steps of the sequence.

In (Tenorth, Clifford Perzylo, Lafrenz, & Beetz, 2012), the capability to imitate is semantic reasoning. They present RoboEarth, which is a European project (whose slogan is 'A Worldwide Web for Robots') that attempts to be an open-source network repository where robots can share information and learn from each other, about their behaviors and their environments. Its way of encoding

information makes use of an ontology (semantic knowledge). In the technical part, the system is similar, in the way of working, to any cloud computing system, with distributed units (the robots), and a central system on the internet. RoboEarth allows the robot to connect to a web service, and upload (or download) data information: maps for navigation, manipulation strategies, object model, etc. Inside the cloud, data is linked with semantics through an ontology language, which allows semantic inferences.

In (Pape et al., 2012), curiosity is the focus, and they consider it as a reinforcement learning system, where the robot must decide which actions to take in which states to maximize its expected reward. They consider the reward, not as a externally-specified task that needs to be solved, but as as a learning of different behaviors based on patterns discovered in the sensory inputs. A very similar scheme is followed in (Ngo, Luciw, Forster, & Schmidhuber, 2012), where they implement a robotic arm with some colored blocks, and the reward is managed as the learning of new behaviors.

Mental models (Roy, Hsiao, & Mavridis, 2004) (Mavridis & Roy, 2003), a kind of imagination of objects and situations not currently being perceived, have also attracted researchers. Mental models have their own section later, so we will not further describe them yet.

As can be seen, separate developments have been able to reproduce small consequences of cognition, but the initial formation of thoughts, or knowledge development of the brain, remains as an unachieved challenge, and maybe the underlying mechanisms are undiscovered even for neuroscientists.

To manage these developments involving machine cognition, there are some computational architectures, or processing systems, with a real focus on endowing cognitive capabilities to robots. Most notable are Soar (Laird, 2012), Cogbot (Goertzel, Garis, Pennachin, & Geisweiller, 2010), TRoPICALS (Caligiore, 2011), KnowRob (Tenorth & Beetz, 2009), and the one developed for the iCub (Sandini, Metta, & Vernon, 2007). Each of them takes different initial assumptions about

how the cognition must be tackled or developed. These distinct views are influenced by the theories of artificial cognition stated by AI and robotics researches. Let us overview the evolution of these theories through time.

1.1 Artificial Cognition Through Time

There are several theories of how cognition should be implemented in robots, or in machines in general, with two leading, more influential, theories: cognitivism and the embodied cognition. Although they are nowadays superposed, we can establish a temporal line about computer researches involving cognition, and how they have evolved through time.

1.1.1 Cognitivism

Some projects have provided, as a solution, models based uniquely on symbols. According to (Vernon, Metta, & Sandini, 2007), the origin of this vision comes from cybernetics, with the idea of an intelligence based only on logic. This is a purely symbolic approach to the *mind-body problem* (with a kind of *mind-mind* solution). This group of researchers have been called cognitivist. They believe that a truly artificial consciousness, an intelligent system, can be achieved exclusively by the treatment of symbols, and no body is required to achieve it. Most of their investigations can be labeled as belonging to the computer scientist and artificial intelligence (AI) world.

They rely on the manipulation of symbolic representations of the state and behavior of the external world to facilitate appropriate interactions. The storage of new knowledge gained from the experience is used to reason (manipulate symbols) more effectively in future computations. Perception is understood as a previous step in the process of abstraction of representations of the external world, because the reasoning is symbolic, as concepts are manipulated to infer changes in the world, as a sum of symbol consequences.

One characteristic of these systems is that knowledge is human-designed, which benefits of direct understanding by humans. This fact limits the capability of the system to learn beyond the structure given. As an example, an automated planing system, makes assertions about the world state during its execution, like "block A is on block B". This statement, which is clear for an idealized environment, can be not so defined in real world. Assuming two blocks with similar size, "block A is on block B", represents lots of different possibilities, all of them fulfilling the statement e.g. blocks can be twisted with respect to each other (in a range that in continuous), blocks can be inclined due to the action of putting in there, blocks can be not exactly aligned, leaving a gap in one side, and a salient in the other, one block can be deteriorated in a corner by humidity, which makes it less reliable to be stacked. How can a system represent this wide, and continuous, range of aspects by saying "block A is on block B"?

Examples of cognitivist architectures are the ACT-R (Anderson, 1995) or Soar architecture (Laird, 2012). In cognitivist architectures, the focus is on invariant and task-independent aspects of cognition. For these systems, the knowledge must be introduced by the programmer, leading to a model of symbols, and rules to manage and transform them. For instance, Soar behavior is a combination of rules produced by the system, in 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 the solution. Every cycle is composed by two phases: elaboration (knowledge recollection) and decision (choosing next action to be taken).

On the other hand, ACT-R is composed of several specialized modules, each processing a different kind of information: vision, manual module for controlling hands, declarative module for retrieving information and goal module (internal state when achieves a task). There is a coordinator which manages all others modules. Reasoning is a cyclic process where patterns of information are identified, and a production of new rules is fired. One example of the gap between the world and the symbol management is the perceptuomotor system of

ACT-R. It does not code direct sensor information, but assumes that perception has translated the visual data into objects (symbols), centering only in attention and recognition.

The catalog of cognitivist tools include, among others, the ones from automated planning, logic-based and knowledge-based, such as expert systems, case-based reasoning or ontologies. To put the reader in context about how symbolic tools obtain results, let us explain how ontologies work.

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 the parent-child relationships (Schlenoff et al., 2012). One module of the ontology is the language which it is written. The languages used are called *descriptive languages* because they represent the logic of the ontology in form of facts and rules, instead of representing the control flow as traditional, *imperative languages* (C/C++, Java, etc.). Once built the structure, we can extract, also called infer, information from the ontology. Reasoners, also called inference engines, can deduce some information based on the provided ontology and the input world facts. As an example, let us imagine a simple ontology with only one rule: "If X croaks and eats flies then it is a frog", where X is undefined and represent a potential concept. Now, as a world fact we could input that "*Kermit* croaks" and "*Kermit* eats flies", where *Kermit* is an instance of a class, a real entity beyond the ontology. With this information, and by composition of facts, a reasoner could deduce that "*Kermit* is a frog". Some famous ontologies languages are CycL, used by Cyc (Lenat & Guha, 1989), one of the oldest computer ontologies in the world, and OWL (McGuinness & van Harmelen, 2004), very popular nowadays because of the rise of the semantic web and because it is being used in the robotic cloud computing project known as RoboEarth (Tenorth et al., 2012).

One problem the cognitivist have to face with is the separation from the real world, called *symbol grounding*, but also called *semantic gap* or *language grounding*.

Let us name what it means, and what it implies.

1.1.2 Symbol Grounding Problem

The symbol grounding problem, first defined by Harnad in 1990 (Harnad, 1990), is the concept of linking words with abstract concepts, to mix words we use to define concepts with what we can perceive through our senses, somehow like the structures in our brain which allow us to keep and retrieve the information. It deepens in the most intimate concept of meaning, as an idea or a mental model.

One idea Harnad expressed is that *names* and *categories* alone cannot represent the meaning of a concept, because they cannot be recombined, and there must be a sensorial information attached. For example. imagine ”Horse” as a mix of categorical and iconic representation, and imagine the same for ”Stripes”. With these conditions, a symbolic new category ”Zebra” can be formed as the conjunction of both previous categories. A person who has never seen a zebra could identify one by joining both representations, which cannot be done only with a symbolic approach.

Symbol grounding problem have been studied in such different fields like psychology (Barsalou, 2010) or semantic web (Cregan, 2007), but our focus is their application to the robotics world (Coradeschi, Loutfi, & Wrede, 2013). As an opposition to cognitivism, and trying to solve its problems, it emerges a new theory called embodied cognition.

1.1.3 Embodied Cognition

Moving away from cognitivism, some new theories emerged claiming that mind is not isolate in the world, and that the process of learning is intrinsically influenced by the interaction with the things to be studied. They believe that cannot exist a concept without their physical experimentation (e.g. one cannot really know what “roughness” means without a tactile interaction) (Pfeifer &

Bongard, 2007). Those researchers defend the embodied cognition. The brain affecting the world through the body, and the world changing the mind with these actions, always with some kind of self-organization of knowledge in the middle.

For them, cognition is understood as the process of a system to adapt to an environment. These system are self-organized structures which continuously readapt itself in function of the environment stimuli. In some way, behaviors are a kind of own ontology, but specific to this embodiment, because they also use the external inputs which leads to an end, but the roads in embodied cognition are self-constructed and time-varying.

This movement can find its roots, regarding robotics, in the works of Rodney Brooks at *Massachusetts Institute of Technology* (MIT) about the *subsumption*, or reactive, architecture in 1991 (Brooks, 1991). Instead of symbols, they worked with behaviours, inspired on how insect works (without a high cognitive brain like us). But, in the last decade, one of the most passionate defender of embodied cognition has been Rolf Pfeifer, publishing several books about it (Pfeifer & Bongard, 2007) (Pfeifer & Scheier, 1999). Embodied cognition usually groups some different and related psychological and philosophical approaches to cognition (further information in (Barsalou, 2008)):

- **Embodiment**: supports that the human mind is largely influenced by the form of the human body. The way this body interacts with the world largely determines how learning processes are performed.

- **Emergence**: states that complex systems acquire their complexity and patterns as a results of performing simple interaction in the real world. These complicated behaviors *emerge* as the sum of the more simple interactions with the environment.

- **Connectionism**: Very related with emergence, it believes that mental phenomena is the result of interconnecting simple units in large networks. The

knowledge is produced by interaction, because networks are connected to the world. The most notable development in this area is the Artificial Neural Networks (NN) paradigm.

- **Enaction**: This concept declares that knowledge is produced by motor skills, through perception-action in the environment. Coordination of several senses produces the association of input-effects that defines cognition.
- **Situated Cognition**: This extends the embodiment definition claiming that cognition is modelled as a result of (and inseparably from) physical interactions, but also of social and cultural contexts.
- **Distributed Cognition**: proposes that cognition is not confined to the individual and that it is distributed by placing knowledge, like memories or facts, on the objects, on other individuals, and on tools from our environment.

Common techniques used by researchers in this area are: neural networks (connecting world info to behaviors), learning by demonstration (imitation of human actions), reinforcement learning (cyles of repetition and correction of tasks), affordances (property of an object to be used in a task (Gibson, 1977)) and motor babbling, among others. iCub platform is an example of this paradigm, with several publications applying these principles. iCub is a humanoid robot, whose dimensions, 90 cm of height and a weight of 23 kg, are similar to a 3.5-year-old child and its developments have a special focus on cognition, specially the embodied one. It has 53 degrees of freedom, with 9 in each hand. Once defined the robot, we can now introduce some of these cognitive publications.

In (Marocco, Cangelosi, Fischer, & Belpaeme, 2010), they equipped iCub with a recurrent artificial neural network (RNN, networks learning through time) and let the robot interact with objects located on a desk. The network has, as input: 3 units to code the values of three joints (shoulder, pan-neck and tilt-neck), 1 unit to code a binary tactile information of the hand, 1 unit coding the roundness of

the object and 3 units to code a binary encoding for the objects ([1 0 0] for the sphere, [0 1 0] for the cube, and [0 0 1] for the cylinder). There are 10 hidden units and the 8 outputs units are trained to predict its next input state. The aim of this work is to learn sensorimotor variations produced by the robot manipulation of the object presented. After the training phase, the network input is connected to actual encoders, and the output is used to determine next joints movements. According to their results, iCub is able to reproduce object dynamics by acting on the environment and, if linguistic input is suppressed in the testing phase, it is also able to associate certain temporal sensorimotor dynamics to different categories, a kind of grounding.

Another work is (Stramandinoli, Marocco, & Cangelosi, 2012) where they also use RNN in two different ways. The aim of this experiment is to teach iCub the meaning of high-order concepts, as a combination of primitives, defined also semantically. There is a first training of the network that links words with motor primitives from a library. Input of the network is a encoding of 13 words and the output is a encoding of 7 basic action primitives. Output values are linked back to the hidden layer in order to achieve a temporal model. The second training set consists of sequences of temporal patterns that encode sequences of actions. That means, basic action words activate a single output neuron (associate to a primitive) and higher-order concepts activate a temporal sequences of action primitives. Imagine KEEP word as a combination of GRASP and STOP. In a first training, when the network has as input GRASP it activate GRASP output neuron, and the same for STOP. In a second training, starting from the first training, the network is stimulated with an input that represents KEEP, and the output is the activation of, first, the neuron for GRASP, and next, in the next step, with the neuron of STOP. With this scheme they are able to teach the robot how to perform actions which are decomposable in motor primitives.

Other works has also used iCub as platform. We can briefly name two more. In (Stoelen, Bonsignorio, Balaguer, Marocco, & Cangelosi, 2012), they train a

network with time-delayed input layers, to mix low level motor joitns and semantic information (called labels). Another work is (Calinon, D'halluin, Sauser, Caldwell, & Billard, 2010), which will be further analyzed in the state of the art chapter, but just as an advance, it uses a combination of machine learning techniques to generalize cyclic movements to the robot, who is able to reproduce them later.

Leaving iCub and coming back to the theories of cognition, we want to summarize the opposite points in both theories. Thanks to the review done in (Vernon et al., 2007) about artificial cognitive systems, we can list some differences between cognitivism and embodied cognition:

- **Operation**: while cognitivists use symbol rules, embodied use distributed, self-organized networks.
- **Representation**: cognitivist use descriptive patterns of symbols that refer to events in the world. Embodied representations are global states encoded in the distributed network.
- **Grounding**: cognitivits use symbols, embodied use non-direct interpretable networks.
- **Temporal Constrains**: cognitivist's symbols are not related with time, while embodied representation is produced because of real-time interactions.
- **Embodiment**: cognitivists do not need a physical entity, but this is the first requirement for embodied.
- **Actions**: for cognitivist, actions are consequences of symbols, and for embodied, an action is a perturbation of the system.
- **Adaptation**: cognitivism considers the acquirement of new knowledge (symbols or rules) as adaptation, while in embodied system, it implies a structural adaptation of the network.

1.2 Objectives

In this master thesis, the author aims to continue the line started by the embodied cognition paradigm, because of the use of robot perceptions, but also use symbols to ground and infer knowledge from objects and actions. The symbols used are the ones related with language. As stated in Fig. 1.1 we take, from the real world, objects, words and actions and try to extract common characteristics to infer knowledge.

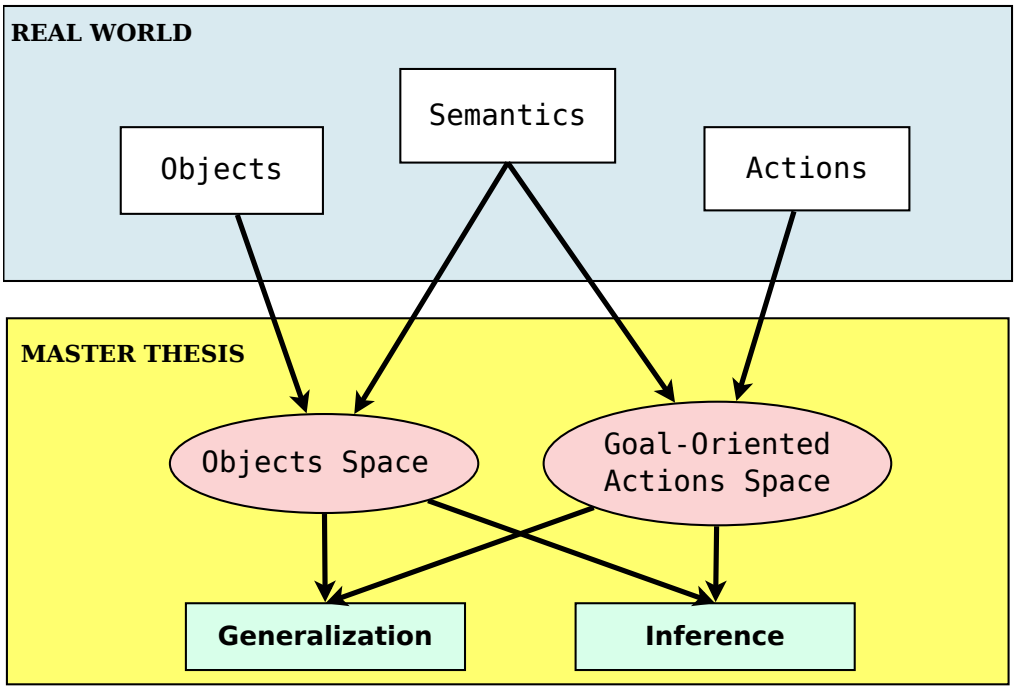

Figure 1.1: *Diagram representing objectives of this Master Thesis.*

The technical targets of this project are the formalization and implementation of a system able to:

1. **Generalize objects features** based on robot perceptions, by using a n-dimensional feature space and semantic spoken information. The char-

acteristics of the objects space will be explained in Section 3.1, while the object generalization in Section 3.2.

2. **Infer objects characteristics** based on their semantic descriptions and create mental models. Theoretical objects feature inference is tackled in Section 3.3, with experimental tests in Section 5.1 and 5.2.

3. **Generalize actions** by their consequences in objects (object-centered) in a n-dimensional feature space. This will be explained in Section 4.1.

4. **Infer actions influence** in the environment based on their semantic label (action name). Those inferences are achieved in static start-end snapshots (a kind of goal orientation). This will be also explained in Section 4.1, with experiments in Section 5.3.

5. Use those processes in **real world applications**, specially the robotics one. The whole experiments set is contained in Chapter 5.

These are the specific targets of this master thesis, but in a more general way, the researches performed by the *Robotics Lab* tries to develop more intelligent robots with a neuroscientist inspiration. Our long term goal is to replicate the behavior of mirror and canonical neuron system.

Chapter 2

State of the Art

The works developed in this thesis join many areas of study, some involving traditional AI, semantics, neuroinspired and embodied cognition, etc. Because of this, we have divided the related works in two categories: Neurorobotics and Cognitive Machines.

2.1 Neurorobotics

Neurobotics is the design of computational structures for robots inspired by the study of the nervous systems of humans and other animals (Arbib, Metta, & Smagt, 2008). That is, robotics with neuroscientist inspiration. From among the variety of topics it involves, our focus is only on the mirror and the canonical neuron system. After an introduction of this kind of neurons, there will be presented some computational models which tries to imitate their behavior.

2.1.1 Mirror and Canonical Neurons

Mirror and canonical neurons are allocated in the motor area of the brain (Gallese, Fadiga, Fogassi, & Rizzolatti, 1996). If they are important to our work, it is because they support the idea of a goal-oriented coding of actions in the

human brain. There are some differences in their behavior, defining:

- **Canonical Neurons**: induce motor action preparation by activating not only during action execution but also as a consequence of the perception of affordances (understood as the property of an object to be used in a task, e.g. a knob affords twisting).
- **Mirror Neurons**: fire both when the animal performs an action and when he observes the same action performed by others.

Some empirical evidence of this conduct is shown in (Umiltà et al., 2001), where mirror neurons were proved to fire when the beginning of an action is shown, but not the end, demonstrating their goal-oriented coding. Evidence can also be found in (Umiltà et al., 2008) and (Rochat et al., 2010), where monkeys are trained to grasp food with different tools, activating these same neurons when observing the action perform by the experimenter with another tool. The monkeys use pliers, which leads to activate muscles in a certain way, as long as the pliers need to be press to close the two mobile parts, and be release to open them again. On the contrary, the human demonstrator uses an inverse-plier, whose mechanism to be used is exactly the contrary, pressing to open, releasing to close. No matter the plier, or the action executor, same group of neurons get activated, so the way-to-do-it does not seem the reason for their fire, and the goal of the action seems to be the motive. The relevance of mirror neuron system has been noticed by the AI and robotics community, developing some computational models imitating them.

2.1.2 Computational Models

From (Thill, Caligiore, Borghi, Ziemke, & Baldassarre, 2013) we highlight two computational architectures inspired in the mirror neuron system: MNS and the one by Metta. MNS (Oztop, Kawato, & Arbib, 2006) is an extension of a previous computational neuromodel called FARS (Fagg & Arbib, 1998). It

extends FARS by adding a simulation of the behavior of brain areas supporting mirror neurons. One of its keys, is the functionality of hand-state estimation based on vision analysis. The parameters extracted are two, the purely related with the hand (e.g. aperture, velocity) and the relative relation to objects (e.g. hand-object distance, angle difference between hand and object principal axis). They use a simulator that generates trajectories to grasp objects (prehension mode), while training the mirror neurons (standard error backpropagation from neural networks). When they want to recognize actions (action recognition mode), the system observes hand moving towards an object, while processing information referent to the hand and the object. The information provided allows to recognize the action. The system was added with two extensions in MNS2-I (Bonaiuto, Rosta, & Arbib, 2007): the mirror neuron were fired not only when the action is observed, but also when it is heard and they were fired also when the last part is not observed (a reproduction of Umiltà experiment (Umiltà et al., 2001)). A second revision of the system, MSN2-II (Bonaiuto & Arbib, 2010) lead to include other capabilities such as an output monitoring system to check the results of action. With this movement, they are trying to discover a way to adapt known functions to new environments. Some other additions are a kind of reinforcement-learning module, where the system can reward successful actions, and also motivational states, a kind of drives which regulate the weights of the network, in function of current state, to influence towards a particular result.

Metta's model (Metta, Sandini, Natale, Craighero, & Fadiga, 2006) is very similar in the conception to MNS, and the enactive cognition plays an important role in its development, with a robotic platform interacting with its surrounding. Action effects are measured using the approximation of (Fitzpatrick, Metta, Natale, Rao, & Sandini, 2003), which means that they use optical flow variation measures by a vision system, and extract properties of this movement. The technical system is composed by a camera and a motion capture glove, so the

variety of detectable affordances is reduced to object movements (rolling properties). This affordance is detected by a canonical system which acts as a forward model predicting the action outcome. Another posterior module compares the predicted and the sensed signals, and the results is the learning signal. The F5 monkey brain area (equivalent to Broca area in humans, which contains canonical and mirror neurons) is represented here as a closed-loop motor controller plus the learning signal.

Despite the different implementation techniques on both architectures, they share a common point in the calculation of relative parameters and the use of prediction models. The modules presented in this thesis aim to serves as the beginning of an action effects generalizer and recognizer. Apart from the neuroscientist inspiration of the work, we also need other technologies to build the system, tools integrated nowadays in the so called *cognitive machines*.

2.2 Cognitive Machines

Adapting a definition from (Haykin, 2005), a cognitive machine is an intelligent system that is aware of its surrounding environment and uses its understanding to learn through interactions with the environment, adapting its internal states to variations in input stimuli by making corresponding changes in the system in real-time with specific objectives (e.g. reliability, efficiency). Specifically, a cognitive robot is defined (Wang, 2010) as an autonomous intelligent system that mimics the cognitive mechanisms of the brain by using computational intelligence. As can be seen, the definition is not closed an involves several fields of investigation. There have been selected the most related to this thesis, to be presented as related works, but the topic is much bigger and grows everyday.

2.2.1 Semantic Analysis and Symbol Grounding

Involving semantics and AI, there are approaches like Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997) and Hyperspace Analogue to Language (HAL) (Burgess, Livesay, & Lund, 1998). These are works with purely symbolic management, a reason that was criticized because the absence of connection to the real world (no symbol grounding) (Glenberg & Robertson, 2000).

The first work with a link between sensorial information and language was VIsual TRAnslator (VITRA) (Herzog & Wazinski, 1995). Non-static situations are provided in the form of video sequences, and VITRA is able to analyse and perform automatically generate natural language descriptions for the scenes it recognize. After training, VITRA is able to answer simple questions about situation such as: traffic jams, soccer match and route planning. In (Steels, 2001), they ground information in the context of games, where the catalogue of interactions is very limited and constrained. Through simple interactions guided by the user, they are able to associate semantic commands to simple actions. Inference is not covered in these works, and just the linking semantic-action is considered.

We are also interested in how semantic can be linked to specific complex actions in complex worlds, beyond the constrained examples presented. In (Hasegawa, Rzepka, & Araki, 2009), starting from motor angle patterns, they transform the actions perceived into context-independent rules related to action verbs. On the same line, but using NN, (Marocco et al., 2010) constructs a fully-recurrent neural network, where the inputs are both, semantic and kinematic parameters. The words are encoded in a binary way, which is sometimes called a localist encoding, which means that each word is associated with a single input neuron, and its single activation (usually on/off as $0/1$) refers to the word being used. The number of words is bounded to the number of known objects. The output neurons control motor angles over time, creating trajectories. As explained before, recurrent neural networks are used in (Stramandinoli et al., 2012) in a double way. The first training of the network links words and mo-

tor primitives associated. The second training of the network uses the result of first training to perform high-order words understanding, by activating, for a high-order input, a composition of primitives as output.

Our system is highly dependent of semantics, as long as it allows a classification of object and actions by its name. The main difference of this thesis with the presented semantic works, is that, our system is able to understand a description of an object and returns coherent values. In the case of actions, it can predict how an action will affect an object, only by naming the action.

2.2.2 Mental Models

The use of mental models is an important aspect in cognitive machines. Mental models are the initial step for imagination in robots, and may enable real world applications, such as drawing. The topic has been studied, mainly, in conversational robotics by Deb Roy at MIT. For instance, in (Mavridis & Roy, 2003), they achieve what they have called *object permanence* (awareness of an object when it is not visible). A simulator is used in this case, as they instantiate objects detected with the vision system, and transform them into virtual equivalents. Virtual objects can be semantically described, adding two extra options. Different spatial perspectives are allowed in the descriptions ("touch the block on my left"), as well as time perspectives ("touch the block you were just holding"). In (Roy et al., 2004), the system is endowed with multimodal inference. Force necessary to manipulate objects is inferred by instantiating objects in the simulator and running a dynamics engine. Vision techniques allows to calculate object velocities in function of forces applied.

The most similar development to this project is shown in (Roy, 2002b). The main target of this system (called *DESCRIBER*) (Roy, 2002a) is to assign ranges of values of features to words. A training phase is carried and human-made descriptions are transcribed to define synthetic colored rectangles shown in a screen. Descriptions are filtered to associate only the relevant ones (e.g. sub-

stantives and adjectives are relevant, but prepositions are not). Every word is only potentially relevant and the word is linked to a subset of features, corresponding to the analysis of the rectangle. In this point, they need to check what word best represents, or describes, a feature. The distinction is made by comparing feature distributions between descriptions formed with these words to see their influence. They find the subset of features for which the distributions are maximally divergent (more different) when the target word is present in the description and when it is not. The maximum difference correspond to the most relevant word for these features. The results allows the generation of correct semantic descriptions of the rectangles selected by the user on a screen, also including spatial relations to other objects (Figure 2.1).

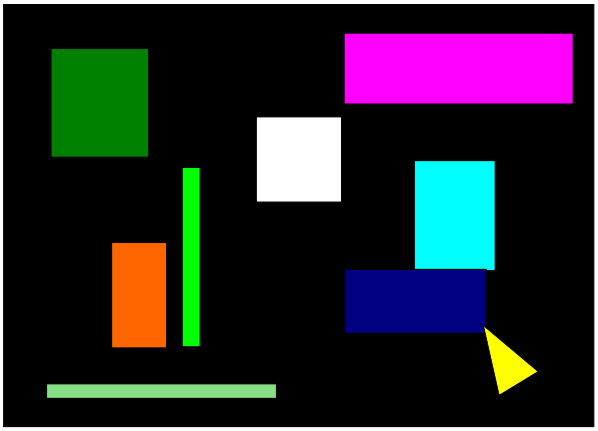

Figure 2.1: *DESCRIBER output is the generation of natural descriptions of selected rectangles. Figure based on (Roy, 2002b).*

The apparent limitation of the system is in the shape of the elements, working only with rectangles. A larger review about grounding symbol with perceptual categories can be found in (Roy, 2005).

Our system aims to be different of Roy's works in that, we follow the inverse process than DESCRIBER. As long as they select an object, the system analyses

and finally construct a description in natural language, we describe an object semantically, and the system returns the feature values expected of an object with this description.

2.2.3 Object Affordances

The Gibsonian definition of affordance (Gibson, 1977) is the action possibilities that an objects offers. In other words, the property, or quality, of an object to be used in a task. Typical examples of affordances are: knob affords twisting, button affords pressing, mug handles affords holding. The brain representation of affordances (Fagg & Arbib, 1998) are supposed to encode direct object properties (size, location) but also relative agent-objects properties (relative distance, actual contact). Canonical neurons seems to manage this process, because they fire with the simply presentation of an object, preparing motor actions to their execution.

This thesis is very related to affordance topic because of the goal-oriented coding of objects in actions. For example, (Fitzpatrick et al., 2003) try lo infer the action possibilities of object motion by pushing-pulling them and analyzing the results, acquiring affordances (what actions objects can "afford"). These affordances are detected as optical flow variation measures by with a camera, from which properties of this movement are extracted. In (Demiris & Dearden, 2005), they try motor babbling, which are randomly generated motor parameters, to measure action consequences and compare them different models. Object tracking in a scene is achieved by clustering regions in the image with similar position and movement properties. The random motor consequences are later added, as nodes, to a Bayesian Belief Network (probabilistic model that represents a set of random variables and their dependencies as a directed graph). They assure that robot learns the forward model for the movement of the robot gripper.

Although techniques are very diverse in affordance topic, connectionism, in different forms of neural networks, seems to be a big influence. In (Cos-aguilera,

Cañamero, & Hayes, 2004) a Self-Organizing Feature Map is trained to identify object features and link them to the success or the failure in the accomplishment of the task. In a similar way, (Dogar, Cakmak, Ugur, & Sahin, 2007) first executes motor primitives randomly to objects and then trains a Support Vector Machine to link actions and effects. In (Montesano, Lopes, Bernardino, & Santos-Victor, 2008) they take a development approach, which means that actions are constructed over the previous, and simpler, known actions. A visual system tracks an object, and by applying different random actions to an object, the robot groups tactile and vision variation using a type of graph (Bayesian Network).

Our system can be considered as a first step in the development of complete affordance module for robots. We code the action by its goals, and not by the way they are achieved. We are different from presented affordance works in that, they link an input, usually some kind of motor parameters, with an output, the change in the object. In our case the input considered is the name of the action, and the output is, equally, the change in the object. In future revisions of the works, internal changes in the robot when executing an action could also be considered.

2.2.4 Feature Inference

Object feature inference is not a very studied topic, and only two examples have been found in literature. One work is (Nakamura, Nagai, & Iwahashi, 2007), where they use an extension of probabilistic Latent Semantic Analysis (pLSA) to infer features using information from different sensorial channels. pLSA is usually used in statistical natural language processing, as a way to measure the probability of a word to appear in a certain document, or in a specific topic (co-occurrence data). They have modified this method to check the co-occurrence of sensorial data (as if they were words in a document): visual, auditory and haptic. There is a big embodiment vision of knowledge in this work,

as the robot constructs information based on interaction with the environment. In a first stage, the robot grasps an object and observes it from different perspectives, recording its visual, auditory and haptic features while moving it. Their aim is to categorize different objects in function of their features, but, with the use of pLSA they can also infer features e.g. how an object sounds, based on how the object looks. Despite they are highly focused on multimodal categorization, they provide a direct cross-feature mapping that allows to infer certain features in function of other.

The other work found, is an already cited work (Roy et al., 2004). Previously to explain how the inference is performed, let us overview the platform and the experiment. The robot is "Ripley", a manipulator with vision cameras mounted, and the environment of the experiment consist in a set of objects on a table. Image processing identify objects and human faces, and maps this information into a simulator (what they call *mental imagery*). The simulation environment contains the table, the objects and a representation of a human operator. It is also used a dynamics engine to predict future states of the system. Regarding inference module, when objects moves, the system infers forces necessary to produce this movement. This can give an idea of the dynamics features of the object (self-moving, rolling, etc.).

Our work is more focused in feature-semantic inference, that is, guess the features of an object, by its description. A way to achieve this is to use ontologies, which are very good at inferring semantic knowledge. The problem here is the ungrounded approach, remaining the process only in a symbolic ungrounded development.

2.2.5 Action Recognition and Imitation

Humans perform thousands of actions every day, and for a seamless Human-Robot Interaction and the introduction of robots in our daily life tasks, action recognition is a key factor. There are many computer approaches to the field,

and the literature is quite extensive. Here, there will be reviewed, specially, the goal-oriented actions, and the ones using learning but, for a larger survey, consult (Poppe, 2010).

Human Action Recognition

The learning techniques involving actions for generalization and imitation has been called *learning by imitation* or *programming by demonstration* (Calinon et al., 2010). The way these methods learns an action is by recording the kinematics of the actions that humans perform, such as arm trajectories of a human reaching an object, and applying then different machine learning algorithms. The mathematical techniques used are, among others, Hidden Markov Models (HMM, a kind of Bayesian Network), for modeling actions in terms of human and robot joint position trajectories (Calinon & Billard, 2004) (Calinon & Billard, 2005), or Gaussian Mixture Models (GMM, probabilistic model) as in (Calinon & Billard, 2007). Let us extend the process followed in one of the papers. In (Calinon et al., 2010) a human operator performs a task several times (e.g. hitting a ball) using a robotic arm. Positions, orientations and velocities of the arm are recorded and with HMM, they estimate the number of states, so that, representative steps. HMM is used to handle spatio-temporal variabilities of trajectories across several demonstrations. Finally, and in order to reproduce the generalized trajectory, Gaussian Mixture Regression is then used to create a regression function with HMM states. This reconstructed trajectory is the one the robot reproduces to imitate the human movement.

If we refer to direct action recognition, that is, recognize an action by external measurements, but typically observation, techniques used are more diverse. In (Subramanian & Suresh, 2012), they do a neuro-fuzzy classification of optical flow features between consecutive frames of human movement sequences. Neuro-fuzzy is a combination of fuzzy logic with neural networks, by using as input to the neural network, the classified output of a fuzzy system. Tracking

and filtering human hand and feet trajectories through Principal Component Analysis (PCA) is the method selected in (Chivers, 2012). First they record trajectories from a video by tracking key points. After a smoothing of trajectories, 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 for similar actions. For recognition purposes, they record an action, transform it as explained, project its vector in the reduced space, and finally, associate it with the closest cluster. Using a demonstration set of actions, in (Gräve & Behnke, 2012), they encode actions as states of HMM, which are later linked by their means, using Gaussian Process Regression.

The focus of all of these studies is also on learning the kinematics of actions, such as the trajectory of reaching for objects. The difference here is the sensorial way to obtain human data.

Goal/Object Oriented Action

However literature from the areas of neuroscience and psychology indicate that the human brain encodes actions as end-goals related to the affordances of objects. For example, 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 perform the grasping) (Bekkering, Wohlschla, & Gattis, 2000). As stated in (Thill et al., 2013), cognitive psychology has found empirical evidence (Elsner & Bernhard Hommel;, 2001) (Kiesel & Hoffmann, 2004) to support the idea that behavior, and motor actions are encoded in terms of goals. This topic was initiated with the Ideomotor Principle (Greenwald, 1970), which stated that actions are represented as perceptual consequences. It then evolved to the Theory of Event Coding (Hommel, Müsseler, Aschersleben, & Prinz, 2001), stating that perception and action have a common representation system in the brain, with networks of features codes (cognitive

structures) called *event codes*.

We differentiate about two closely related, but different in meaning, concepts about actions:

- **Goal/Object Oriented**: In the literature, they are the actions in which the only parameters to be considered, to be learned or imitated, are those of relative differences between the initial and the end (the goal) configuration. They can encode objects (Gallese et al., 1996) or environment parameters, but they always contain initial-end differences. The term Object Oriented is usually reserved, in neuroscientist field, to actions whose goal is an object e.g. point, stare (Faillenot, Toni, Decety, Grégoire, & Jeannerod, 1997) or grasp (Jeannerod, Arbib, Rizzolatti, & Sakata, 1995).

- **Object-Centered**: We denote them as the actions in which the only parameters to be considered are the ones related to the object affected by the action, ignoring the human parameters. It is based on a definition about Object-Centered representations in (Schaal, 1999). They can be goal-oriented (initial-end differences of objects configuration), or continuous-tracking (continuous tracking of object properties), but they refer solely to the object.

When talking about Goal-Oriented Actions in robotics, a goal codification is found in (Calinon, Guenter, & Billard, 2005) where, despite they learn the trajectory to perform an action, they also code some goals to be achieved, even in a distinct way than learned. The robot is programmed to achieve tasks, but not the way to do it. They replicate a psychological experiment (Bekkering et al., 2000) with children, where, in a table, there are colored dots which are alternatively touched by a human with both arms. When the dots stay on the table, children tend to imitate the goal (what dot to touch), and not the arm used to do it. In the replicated experiment, the demonstrator repeats the same task, and while observing the demonstration, the robot tries to extract a set of constraints for

the task, by extracting relevant features. Later, the robot computes the trajectory that best satisfies the constraints and generates a motion.

As the reader may notice, examples for affordances and Goal-Oriented Actions may look quite similar. As said before, both topics are closely related. But, despite that, there are also some not-so-close to affordance approaches to goal-directed actions. By using an architecture called HAMMER, in (Demiris & Khadhouri, 2006) they consider a hierarchical behavior-based architecture to achieve target tasks. In a posterior revision of the architecture, they included content-based selection of behaviors, based, not only in the performance, but in other parallel parameters, e.g reliability, cost and utility of a request (Demiris & Khadhouri, 2008).

The paradigm of Object-Centered Actions is partially covered by those Goal-Oriented Actions whose focus is on reaching same final situation of an object solely. There are no exclusively continuous-tracking Object-Centered Actions references in literature, to the author's knowledge, and the only one, slightly related, found uses a combination of object spatial tracking and demonstrator-hand movement tracking. In (Johnson, 2004), they build a system with a set of primitives actions, which are in fact inverse models. When the human demonstrator performs an action, they track the object and the hand spatially through time. At the same time, they run all inverse models during action stages to see the better performance of each model in each stage. Finally, they construct a high-level inverse model composed of those selected primitives, being able to imitate the action goal with similar spatial movements. Notice that the parameters to be imitated are not robot motor ones, like in previous presented references, the only target here is the hand position. The object tracking is used to identify grasping and releasing stages.

Our work aims to be more extensive than presented works in that, our system allows a complete object-centered paradigm with features beyond spatial ones, being able to include others like changes in color or shape in objects.

Chapter 3

Objects Space

In this section, the methods proposed to generalize and infer objects, taking into account their characteristics, will be explained. There is a core idea in this chapter about the assumptions made to link words and objects. This concept, named Semantic-Feature Space, will also be covered.

3.1 Semantic-Feature Space

This concept represents the basis for feature generalization and word-object linking[1]. The aim behind it is to find a way of mixing the embodied data perceived by the robot sensor, with an abstract symbolic representation, which allows a feasible manipulation, for recognition or inference purposes.

3.1.1 Semantic Labeling

The process we call "Semantic-Labeling" is, in fact, filling a grounding database which represents and stores tagged points in a n-dimensional feature space. Hu-

[1]The concepts named "Semantic-Feature Space" and "Semantic Labeling" are not original of the author, and these ideas are exclusively engineered by Juan González Víctores, director of this thesis. They will be explained, because they are key, and fundamental, concepts to understand the rest of the chapter.

man descriptions of objects are split into representative words and populate the grounding database. Words are combined with real sensor data, represented by labeled points in the n-dimensional feature space (an example with two words, and two features can be seen in Fig. 3.1).

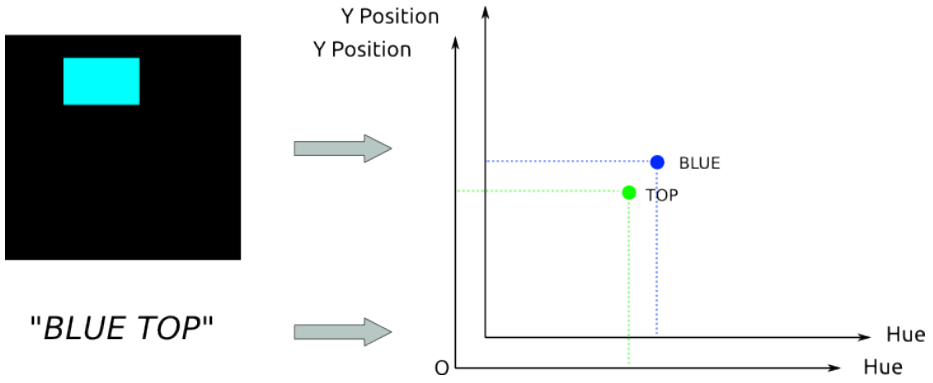

Figure 3.1: *Example of grounding database population.*

The information structure inside database is represented as $< \boldsymbol{f_w}, \boldsymbol{w} >$ pairs. Where f_w is the vector of feature values (in the form of scalars numbers), which becomes coordinates in the space, and w is the word which becomes the associated point label. All the points from a single descriptions share input sensor information, thus their coordinates are spatially the same. Let n be the number of sensed features and $F \epsilon \Re^n$ be the n-dimensional feature space. The space is populated with a set of f labeled points in F. The reason why we create several points, each with the same coordinates, but only differing in the label, is that we expect same-label points, when the populating is complete, will form a kind of cluster expandable in some axis.

We define S as the semantic space (the space of words, that is, the description) and m as the number of different words which have been stored in the grounding database. So, G denotes a grounding function which can be defined as:

$$\boldsymbol{G} : \boldsymbol{S} \rightarrow \boldsymbol{F} \tag{3.1}$$

The semantic-feature linking from the semantic space S to the feature space F is defined as follows:

$$\forall \boldsymbol{w} \epsilon \boldsymbol{S}, \boldsymbol{G(w)} = \{\boldsymbol{w_{f_1}}, ..., \boldsymbol{w_{fi}}, ..., \boldsymbol{w_{fn}}\} \tag{3.2}$$

Where w_{fi} is the value of a single feature for any w. That leads, for a single sample, to the creation of a number of points equal to the number of words used to define the sample. All these points have same coordinates, and the only difference is the word label. When a population has been performed with many samples, there are point clouds with the same label, in areas of the space.

3.2 Object Generalization

Our interpretation of generalization is the creation of a model, a kind of mental model, which describes a keyword in all feature space generalizing knowledge from previously presented objects. We aim to find what areas in the n-dimensional space are relevant for a word, that is, where this word is best represented in the space. These word models have the form of hyperplanes identified by their label words. For each word w, all points containing this word are used to create an hyperplane h_w, of order $n - 1$. The hyperplane is defined in all the space, so the "meaning" of the word is also extended across, and along, features. With "meaning" we are referring where the word represented by the cloud is still relevant in. With extension we refer to those axis where the hyperplane is expanded infinitely. The main difference with usual approach of clustering, is that they want to know where the word is relevant, but not where the word can be extended, conserving its relevance, fact that we explore in our work.

In all space, m hyperplanes are constructed. To review our publication associated to this part of the work, see (Victores, Morante, Jardón, & Balaguer, 2013a).

3.2.1 PCA Algorithm and Hyperplanes Creation

The Principal Component Analysis (PCA) is used to build the hyperplanes. PCA is made to convert a set of observations into a set of uncorrelated representative variables called principal components (PCs). Usual use of PCA is to supply a lower-dimensional representation of a dataset, with the less possible loss of information (Smith, 2002). We also make use of this property, by projecting our labeled points (the ones with the same word label) on hyperplanes defined by their most relevant principal components. Those PCs are the ones which explains the maximum variance of the dataset, that is, those uncorrelated vectors that best define the point cloud. Hyperplanes are $n-1$ dimensional, to assure intersection among them, and the process to build them is mathematically performed as follows:

1. **Mean Subtraction**: For all points of the cloud with the word w, the mean for each feature is calculated (Eq. 3.3).

$$\bar{\boldsymbol{f}} = \frac{1}{\boldsymbol{n}} \cdot \sum_{i=1}^{n} \boldsymbol{f_i} \tag{3.3}$$

   Where f represents a single dimension (feature) and i is the number of word samples. With this calculation we obtain the middle point of the cloud.

2. **Covariance Matrix**: Calculation of this matrix allows us to know the relations and coupling among features. We denote the data of all the words for a single feature as F_i (Eq. 3.4).

$$\mathbf{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} \tag{3.4}$$

A simple covariance between two features, which is no more than their correlation, is defined as in Eq. 3.5.

$$\mathrm{cov}(\boldsymbol{F_i}, \boldsymbol{F_j}) = \mathrm{E}\left[(\boldsymbol{F_i} - \mathrm{E}(\boldsymbol{F_i}))(\boldsymbol{F_j} - \mathrm{E}(\boldsymbol{F_j}))\right] \tag{3.5}$$

Where $\mathrm{E}(F_i)$ is the expected value of the corresponding feature F_i. All the feature weights have been established as equal, so the expected value is a simple average. Joining all features covariances, the resultant covariance matrix A is expressed as in Eq. 3.6.

$$A = \begin{pmatrix} \mathrm{cov}(\boldsymbol{F_1}, \boldsymbol{F_1}) & \cdots & \mathrm{cov}(\boldsymbol{F_1}, \boldsymbol{F_n}) \\ \mathrm{cov}(\boldsymbol{F_2}, \boldsymbol{F_1}) & \cdots & \mathrm{cov}(\boldsymbol{F_2}, \boldsymbol{F_n}) \\ \vdots & \vdots & \vdots \\ \mathrm{cov}(\boldsymbol{F_n}, \boldsymbol{F_1}) & \cdots & \mathrm{cov}(\boldsymbol{F_n}, \boldsymbol{F_n}) \end{pmatrix} \tag{3.6}$$

3. **Eigenvectors and Eigenvalues:** Covariance matrix eigenvectors represent point cloud PCs, and their respective eigenvalues associated inform about the quantity of the variance explained by each PC. That means, the best that each component describes the cloud.

   Let us define what eigenvectors are mathematically, and how they are calculated. 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). That is, an eigenvector of a square matrix A, is a vector v (non-zero) that, when multiplied by A, gives as a result the same vector v multiplied by a single number λ (Eq 3.7, called *eigenvalue equation*).

$$A\boldsymbol{v} = \lambda\boldsymbol{v} \tag{3.7}$$

   Where λ is an scalar called eigenvalue of v. Up to here is the mathematical definition, for interpreting the reason why eigenvector are data defining

directions, we can tackle the definition from other side. The eigenvectors are the vectors for which matrix A simply 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 (point cloud). As the covariance matrix represents couplings between features, the maximally elongated vectors are the best explaining the dispersion. As maths involved to calculate eigenvectors are quite extensive (for a full derivation of the algorithm, see (Golub & Van Loan, 1983)), a summary can be presented as:

- The matrix A is reduced to upper Hessenberg form H, which is an almost triangular matrix with zero entries below the first subdiagonal, as in Eq. 3.8. This is achieved by orthogonal similarity transformation $A = QHQ^T$, where Q is orthogonal and H has the same eigenvalues as A, but is easier to handle for the next algorithm.

$$\boldsymbol{H} = \begin{pmatrix} * & * & \cdots & * & * \\ * & * & \cdots & * & * \\ 0 & * & \cdots & * & * \\ 0 & 0 & * & \cdots & * \\ & & \vdots & & \\ 0 & \cdots & 0 & 0 & * \end{pmatrix} \tag{3.8}$$

- Then, the QR algorithm is performed to further reduce the matrix H to upper triangular matrix T (called Schur form). The resulting form is $H = STS^T$, with S being a unitary matrix.
- Eigenvalues are extracted from the resulting triangular matrix T. In this matrix, the main diagonal is formed by 1 by 1 and 2 by 2 blocks, which are the eigenvalues (1 by 1 if real or 2 by 2 if complex eigenvalues). An example can be found at Eq. 3.9. In this case, eigenvalues

are 1, $2 \pm i$ and 3.

$$\boldsymbol{T} = \begin{pmatrix} 1 & * & * & * \\ 0 & 2 & -1 & * \\ 0 & 1 & 2 & * \\ 0 & 0 & 0 & 3 \end{pmatrix} \tag{3.9}$$

Once the value of eigenvalues are known, eigenvectors of T can be computed as a linear system of equations with known coefficients, that is, finding the non-zero solutions of the eigenvalue equation, and then, pre-multiplying by QS to obtain eigenvectors of A.

4. **Sorting Eigenvectors:** Once we have eigenvectors defined, we order them in descending order, by means of their eigenvalues. From the sorted eigenvectors we extract the ordered $n - 1$ best components (called principal components), and generate the corresponding hyperplane h_w.

Once we have those defining vectors, the next step is to build a hyperplane. 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. 3.10), with m being the highest dimension.

$$\boldsymbol{a}_1\boldsymbol{x}_1 + \boldsymbol{a}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{a}_m\boldsymbol{x}_m = \boldsymbol{b} \tag{3.10}$$

This equation can be expressed in matrix form as $Ax = b$. So 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 dot product (Eq. 3.11).

$$\bar{\boldsymbol{c}} \cdot \bar{\boldsymbol{d}} = \sum_{i=1}^{n} \boldsymbol{c}_i\boldsymbol{d}_i = \boldsymbol{c}_1\boldsymbol{d}_1 + \boldsymbol{c}_2\boldsymbol{d}_2 + \cdots + \boldsymbol{c}_n\boldsymbol{d}_n \tag{3.11}$$

According to dot product definition: *two non-zero vectors are orthogonal if, and only if, their dot product is equal to zero.* To define a hyperplane of $n-1$ dimensions,

in a 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. For each of these vectors is performed a dot product with a, because a must be orthogonal to each of them. Let us remind that a is the vector we want to find, the one orthogonal to the hyperplane. These principal components multiplied with a lead to a system of linear equations defined as in Eq. 3.12.

$$\begin{aligned}
\bar{\boldsymbol{v}}_1 \cdot \bar{\boldsymbol{a}} = 0 \Rightarrow \boldsymbol{v}_{11}\boldsymbol{a}_1 + \ \boldsymbol{v}_{12}\boldsymbol{a}_2 + \cdots + \ \boldsymbol{v}_{1n}\boldsymbol{a}_n = 0 \\
\bar{\boldsymbol{v}}_2 \cdot \bar{\boldsymbol{a}} = 0 \Rightarrow \boldsymbol{v}_{21}\boldsymbol{a}_1 + \ \boldsymbol{v}_{22}\boldsymbol{a}_2 + \cdots + \ \boldsymbol{v}_{2n}\boldsymbol{a}_n = 0 \\
\vdots \qquad\quad \vdots \qquad\qquad \vdots \\
\bar{\boldsymbol{v}}_m \cdot \bar{\boldsymbol{a}} = 0 \Rightarrow \boldsymbol{v}_{m1}\boldsymbol{a}_1 + \boldsymbol{v}_{m2}\boldsymbol{a}_2 + \cdots + \boldsymbol{v}_{mn}\boldsymbol{a}_n = 0
\end{aligned} \tag{3.12}$$

With $m = n - 1$. The system of equations has n unknowns and $n-1$ equations, it is indeterminate. To cross this gap, we define $a_1 = 1$, which is equivalent to elongate one variable of a, and the rest get adapted, obtaining a unique solution. This trick does not affect to the solution, and it only changes the magnitude of the vector, and not its direction, which is what we are interested in. The obtained vector $\bar{a}$ is orthogonal to the hyperplane.

The only remaining unknown parameter is b, which is a scalar. If in $Ax = b$, the hyperplane equation, we substitute A 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, middle point of the cloud. An example of created hyperplanes for several clouds, in a 3 dimensional spatial-color space, can be found in Fig. 3.2

This generalization has two facets:

- It is a feature description of objects based on hyperplanes in a feature space. These hyperplanes define the cloud formed by all the points with the same word.

- It is a way to extend the meaning of words across the feature space.

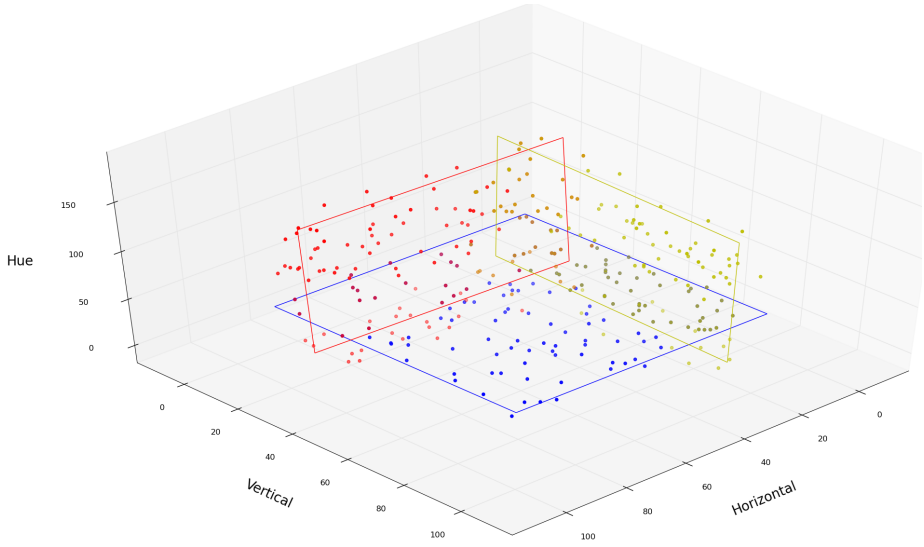

Figure 3.2: *Three planes representing the meaning of three different word point clouds.*

Up to here there have been defined the steps followed to generalize words in a feature space. Next step is to take advantage of these models to infer information.

3.3 Feature Inference

Once the database has been trained, that is, populated, and point clouds have been generalized as hyperplanes, there can be performed an inference step to extract information. We consider the inference of objects features as a kind of robotic imagination, where the features of unknown objects by generalizing characteristics from previously presented objects can be deduced.

In the context of hyperplanes, the inference is exploited as a search of relevant areas in the Semantic-Feature Space, that is, finding the geometrical figure that contains all the valid solutions, when words are used as query. In other words, we want to semantically describe an object, and obtain the features this object would have.

This geometrical figure M is defined as the one resulting from the intersection of the hyperplanes of the query words (Eq. 3.13).

$$\boldsymbol{M}(w_1, ..., w_q) = \boldsymbol{h}_{w_1} \cap ... \cap \boldsymbol{h}_{w_q} \tag{3.13}$$

With q being the number of query words. The intersection of q hyperplanes can lead to different figures, in function of the number of hyperplanes, and the order of the space. Dimension of M can be calculated as $n - q$, and the possible resulting figures are: points, lines, planes, or high-dimensional hyperplanes. These different results can be managed if, when no unique solution is found, we project point cloud center of masses on M. These projections, orthogonal projections in fact, are achieved using an algorithm called Modified Gram-Schmidt.

3.3.1 Modified Gram-Schmidt

No matter the resulting figure, we orthogonally project the center of mass of each n-dimensional point cloud, corresponding to each query word, on M. The technique used is the modified Gram-Schmidt process for orthogonalization (MGS), which performs an orthogonal projection of a vector into another one. A simple projection of a vector v into another vector u is defined as in Eq. 3.14.

$$\operatorname{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u} \tag{3.14}$$

With $\langle \cdot, \cdot \rangle$ representing 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. 3.15.

$$\mathbf{u} = \mathbf{v} - \operatorname{proj}_{\mathbf{u}}(\mathbf{v}) \tag{3.15}$$

This equation involves only two vectors, but this formula can be recursively applied to create a vector orthogonal to k several vectors (Eq. 3.16). We need this multivector projection because M can be highly dimensional (that is, polygons defined by several vectors e.g. two for a plane, three for a cube, etc.).

$$\begin{aligned} \mathbf{u}_k^{(1)} &= \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_k) \\ &\vdots \\ \mathbf{u}_k^{(k-1)} &= \mathbf{u}_k^{(k-2)} - \mathrm{proj}_{\mathbf{u}_{k-1}}(\mathbf{u}_k^{(k-2)}) \end{aligned} \tag{3.16}$$

For q query words, there are q projected points in M. As these points will be on different zones of M, a weighted mean is applied to these points, obtaining a unique solution, which is also contained in M (an example in Fig. 3.3). Currently the weights are all equal. Some advantages can be highlighted, as the solution is:

1. **Complete**: Because it is completely defined in all the feature space.
2. **Relevant**: Because it is contained in the area where the elongations of meanings intersect.
3. **Balanced**: Because all the words influence the same way in the final solution.

This method followed is valid and correct in all cases, but, in order to reduce computation time, some special cases can be defined. We consider special cases, those where the order of M lead to situation that can be simplified. It is important to notice that following cases are just simplifications to improve efficiency, but could be also solved with the previous method.

The first special case is $q = n$. In this situation, M order is null (remember that its dimensionality is $q - n = 0$), the resulting figure is a point, so it is not necessary a further computation, and the solution is this point (an example of this special situation is shown in Fig. 3.4). The interpretation of this situation is that in the description provided as query, there is one word representative for one of the features in the space, so no interpolation is required to supply undefined features.

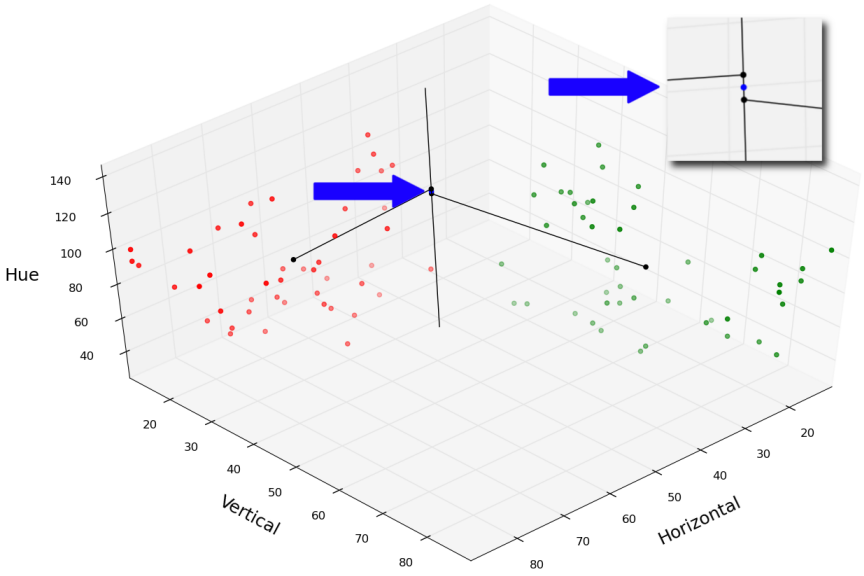

Figure 3.3: *Unique solution using MGS for two query words but three dimensions. The augmented region shows the orthogonal projection of centers of mass.*

The other special case is $q = 1$. In this case, there are no intersection, 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. Obviously this is true for clouds forming convex sets, but it remains uncertain how non-convex sets would respond to this request.

The key point of the object inference presented is that it enables the discovery of objects features from description words, even if they have never seen before together.

Let us recapitulate the steps followed in this chapter to generalize and infer object features:

1. Using PCA we obtain relevant cloud vectors, that is, its principal compo-

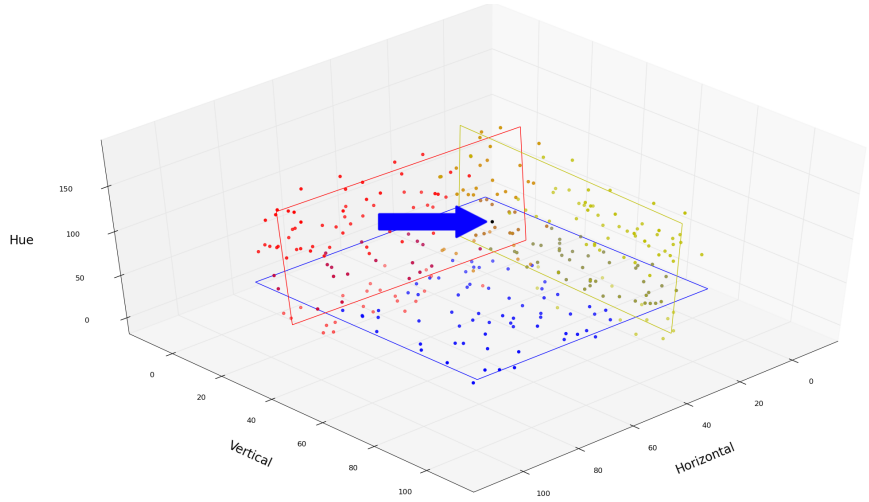

Figure 3.4: *In this special case, intersection point becomes the solution.*

nents.

2. With the PCs, we create a hyperplane for each point cloud. Up to here is the so called generalization.

3. Finally, we intersect the hyperplanes corresponding to query words. These queries are the ones asked by the human when describing an object in order to obtain its features.

This final step is called inference, because the system deduces characteristics of a described object, based on previous experiences, even if this combination of words in the description has never been introduced before.

Chapter 4

Object-Centered Actions

As stated previously, Object-Centered Actions are the ones focused on parameters of the object affected by the action. The intention behind this development is to provide robots with the capability of imagining how a set of actions affect its surrounding environment and, as previously, being able to do it even if the robot has never seen an specific combination of actions, applied to the environment, before. Environment, in the current state of development, means only objects.

While objects features are static, regarding its features, actions are, by definition, dynamic and changing, and their analysis becomes more complex and difficult. This is the reason why the author has decided to address the problem as a Goal-Oriented task. As a reminder:

- **Goal-Oriented Samples**: Every action becomes reduced to a vector of features, where each element represents the variation, in the object characteristic, between the beginning and the end. These values are fixed relative to the initial state of the object (before the action begins).

4.1 Goal-Oriented Samples

This mode does not have a generalization step detachable from the inference process. To create models, there have been selected several machine learning algorithms, each of them creating a different internal model, which serves at the same time as generalization and as parameter generator. The only ad-hoc process carried to achieve action generalization and inference is the grounding of the database in the Semantic-Feature Space.

As in objects case, we feed the Semantic-Feature Space with labeled points in a *n-dimensional* space. This process is exactly the same as performed in previous chapter, but labels, in this case, are not feature descriptions as previously, but the name of the action. After a population phase, we obtain point clouds which represent typical variations of features in an object after the application of a specific action.

Every point created reflects the change produced in the object, and not the movements performed to achieve it. Every time the action is performed, the variation of feature values are extracted from the object. This variation is formally a vector of scalars, which, for the point created, represents its coordinates. The selected algorithms are: Arithmetic Mean Model, Direct-Inverse Neural Networks, Support Vector Regression and Gaussian Mixture Model.

As reader may notice, the algorithms selected do not belong to a specific category of machine learning, and they are very diverse, ranging from classification and regression to probabilistic models. The reason is that the problem presented is not usual, and no specific algorithms exist, so we want to evaluate the strengths and weaknesses of many different algorithms in giving a coherent solution. The aim of the presented techniques is to obtain a coherent model parameter generator for the corresponding semantic input. That means, when action is queried with its name, the algorithm must return the expected variation of features in the object.

Each method has its working mechanism and peculiarities, which will be

explained below. To review our publication associated to this part of the work, see (Victores, Morante, Jardón, & Balaguer, 2013b).

4.1.1 Arithmetic Mean Model

This is the direct, and most intuitive, approach. When receiving a new sample for an action, each feature is updated by averaging with the new value. To avoid the whole average computation of all points in the cloud in each step, an incremental version has been implemented (Eq. 4.1).

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n + \frac{\boldsymbol{v}_{n+1} - \boldsymbol{A}_n}{\boldsymbol{n}+1} \tag{4.1}$$

Where A means average, v is a new value, and n and $n-1$ are previous and current state respectively. This method serves, at the same time, as generalization and as inference. For each new sample, the model is updated, simplifying the cloud to a single point. When asked with an action word as query, the average value is returned. In the case that several query words are used, the resulting value is the composition of both averages. That means, a sum, for each feature, of the values of all actions queried.

This is the simplest of the algorithms implemented, and its results will be presented in the experiments chapter.

4.1.2 Direct-Inverse Neural Networks

As a simple definition, a neural network is an interconnected group of units (called artificial neurons) with connection weights among them. It relies on a connectionist approach to acquire knowledge, which means that it connects inputs and outputs with a network and learns by modifying parameters in the network in a training phase. Neural networks model complex relationships between data using a supervised training phase, where correct outputs for determined input combinations are provided to the network, and learning is performed during this process.

In this case, a classical three-layered fully connected Feed-Forward Neural Network has been selected (Schaul & Felder, 2010). This means that our network has three consecutive layers (called input-hidden-output), where the neurons of a layer are connected with all neurons of the following layer. Actions here are represented by input neurons, with one neuron for each action.

The training algorithm is back-propagation, which for each demonstration sample, updates connection weights to minimize the error. Each neuron has, as activation function, the logistic sigmoid function (Eq. 4.2). The activation function defines the output of a single neuron for a given input value. There is also the possibility of shifting this activation function to a higher or lower value using a *bias*. The bias is directly summed to the activation value of the neuron.

$$logsig(s) = \frac{1}{1 + e^{-\boldsymbol{s}}} \tag{4.2}$$

With a network input x and an output network y, with k neurons, the final result obtained is Eq. 4.3.

$$y_k = b_k^{(3)} + \sum_{j=0}^{H} w_{kj}^{(2)} logsig \left(b_j^{(2)} + \sum_{i=0}^{U} w_{ji}^{(1)} x_i \right) \tag{4.3}$$

Where ${w^{(1)}}_{ji}$ and ${w^{(2)}}_{kj}$ are the weights for the first (input) layer and the second (hidden) layer respectively (between the neuron j-i and k-j in each case). That means that two neurons are connected with a weight between them. These weights are the ones updated in the learning process by backpropagation. H is the number of hidden neurons and U is the number of inputs neurons, while $b_k^{(3)}$ and $b_j^{(2)}$ represent the bias for the corresponding neuron. We do not use a bias in our algorithm, so the equation results as Eq. 4.4.

$$y_k = \sum_{j=0}^{H} w_{kj}^{(2)} logsig \left(\sum_{i=0}^{U} w_{ji}^{(1)} x_i \right) \tag{4.4}$$

The particularity of the network used is not in its structure, which is very common, but in that it is trained inversely when compared to its habitual use.

Usually it classifies a combination of numerical inputs into a delimited number of classes, but in our case, the class is used as input, and the numerical values belonging to the class are the network outputs. The resulting network achieved, with inputs and outputs swapped, is called *Direct-Inverse* Neural Network (Kabir, Member, & Wang, 2008). Our aim is to generate parameters located near the intermediate values of the point cloud (Fig. 4.1).

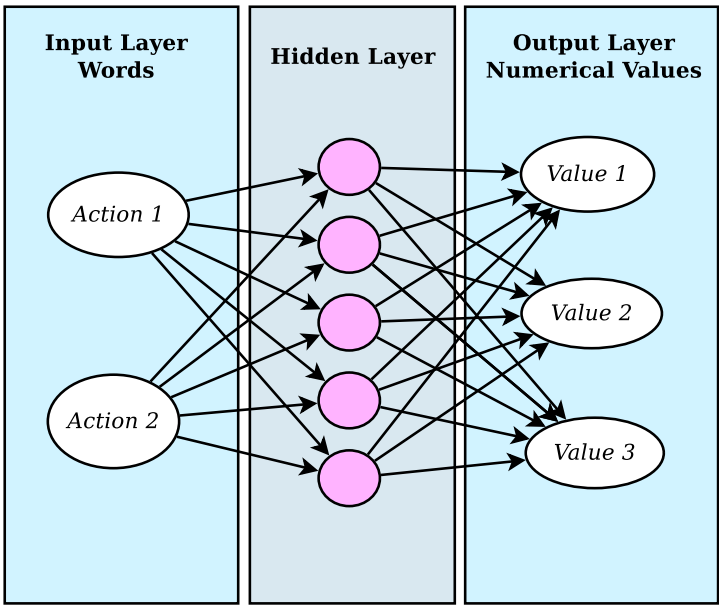

Figure 4.1: *Scheme of our Direct-Inverse Neural Network.*

Some issues arises when inverting neural networks. As stated in (Dua, 2000), the problem associated with this technique is that this mapping results in a one-to-many mapping between the output and the inputs, because in usual neural networks, different inputs can yield the same output. That is, that in inverse neural networks, same input values result in different output values.

Despite it is a problem for other projects, this is not our case. Correct, but slightly different, results, are a good simulation of how biological organism perform actions e.g. when drawing several circles in a paper, they all share a common pattern, but they are not exactly equal. This is the reason why we do not

care about this behavior. Moreover, for actions combination, every action becomes an input neuron, so simultaneous activation is allowed. In the experiments chapter, there will be shown its results, which are quite promising.

4.1.3 Support Vector Regression

Support Vector Machines (SVM), are the third of the algorithms presented, and, as neural networks, they are supervised learning models, usually used for classification, data analysis and pattern recognition. The focus in our case is regression, not classification, so we use a modified version of SVM prepared for regression. Support Vector Regression (SVR) is a modified version of SVM to perform linear and non-linear regression. Good introductions about SVR are found in (Nalbantov, Groenen, & Bioch, 2005) and (Smola & Scholkopf, 2004), although we will review the most important points in the model construction.

For pedagogical reason, there will be explained, in first place, the linear version of SVM, but as a advance, in the SVR for non-linear regression (Pedregosa, Grisel, Weiss, Passos, & Brucher, 2011) (Chang & Lin, 2011), the input data is mapped into a high-dimensional feature space using a fixed mapping (a kernel), and a linear regression is constructed in this feature space.

On it basics, linear SVR tries to reduce model error by minimizing the Euclidean Norm, while using a type of loss function called ε-insensitive loss function. That means, the goal is to find a function $f(x)$ (Eq. 4.5) with, at most, ε of deviation from the actual y_i targets.

$$\begin{aligned} f(x) &= \langle \omega, x \rangle + b \\ with\ \omega &\in X, b \in \Re \end{aligned} \tag{4.5}$$

Where $f(x)$ is a typical linear high-dimensional regression equation and X denoting the space of the input pattern. As said, to solve this regression, SVR minimizes the Euclidean Norm plus some loss values. So, the problem is an

optimization problem (Eq. 4.6), with this scheme:

$$\begin{aligned} minimize \quad & \tfrac{1}{2}\left\|\omega\right\|^2 + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ \\ & y_i - \langle\omega, x_i\rangle - b \leq \varepsilon + \xi_i^+ \\ constraints \quad & \langle\omega, x_i\rangle + b - y_i \leq \varepsilon + \xi_i^- \\ & \xi_i^+, \xi_i^- \geq 0 \end{aligned} \tag{4.6}$$

Defining C as a weight to balance the strictness of penalty (how precise the model is) with flatness of f (how "horizontal" the regression is. Understanding horizontal as lower ω values as possible). ξ variables, called *slack* and usually displayed as (ξ, ξ^*) for our (ξ^+, ξ^-), represent the penalty added by using a loss function (Eq. 4.7). There are two penalties because errors committed in the regression can be upper errors, above the regression function (ξ^+) or lower, under the function (ξ^-).

$$|\xi| = \begin{Bmatrix} 0 & if\,|y - f(x)| < \varepsilon \\ |y - f(x)| - \varepsilon & otherwise \end{Bmatrix} \tag{4.7}$$

In this summary ε is a predetermined non-negative parameter (manually set), y is the true target value and $f(x)$ is the estimated, by SVR, target value. In other words, if the absolute residual is equal to ε or less, then there is no penalty. As there is a no-penalty zone, this method is called ε-*insensitive*. However, if the point is situated outside the allowed band, a linear amount of loss $|y - f(x)| - \varepsilon$ is associated with the estimation. A graphical representation of SVR can be found on Fig. 4.2.

The process above explained represents linear regression. In case there want to be performed a non-linear regression, and previously to apply the method above explained, the data need to be converted into higher spaces, $x \rightarrow \Phi(x)$, presuming they will acquire linearity in that space. This process of mapping could be achieved by transforming all samples into transformed space, called feature space, and then applying linear SVR. But, computationally, this may be

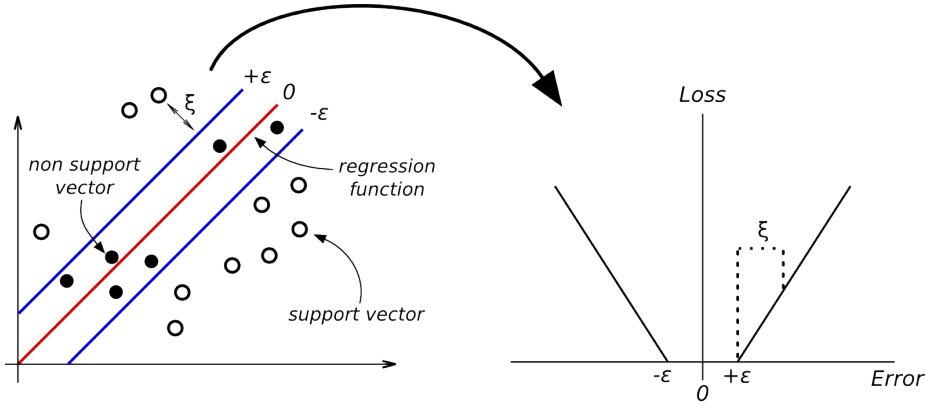

Figure 4.2: *Scheme of SVR for linear regression case.*

unfeasible for very high (or even infinite) dimensions. Luckily, checking Eq. 4.6, the only operation SVR needs to perform in the transformed space is the dot product, so there is a trick to simplify computation by using a technique called *Kernel trick*. A kernel is just a dot product of two vectors (in some space). The advantage of using determined kernels, over dot product, is that, using kernels, dot product in the transformed space can be written out as an operation with the original coordinates, as in Eq. 4.8. Using this trick, transformed coordinates not need to be calculated, and only the kenel is applied.

$$\left\langle \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') \right\rangle = K(\mathbf{x}, \mathbf{x}') \tag{4.8}$$

Where $\Phi(x)$ is the point in the transformed space. There are several kernels, each of then representing a different transformed space. The Kernel used in our case is Gaussian Radial Basis Function kernel, or RBF kernel, which induces a infinite dimensional feature space (demonstration in (Eigensatz, 2006)). RBF is a real-valued function whose value depends only on the distance between two points (x, x') (Eq. 4.9).

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right) \tag{4.9}$$

Where x is the support vector point (called center of the hypersphere), that

is, a point outside the allowed band. x' is the testing data point (actual data) and $||x - x'||$ denotes the norm (here L2 norm), a kind of distance between them. The support vector will be the RBF center and, the last parameter, σ, manually adjustable, will determine the area of influence of this support vector over the space.

Regardless the kernel used, Eq. 4.6 is a convex quadratic optimization problem (which means that local minima is also global minima), with some linear constraints. This can be solved by transforming the problem into two subproblems (a process called dual formulation) and applying and ad-hoc iterative algorithm called Sequential Minimal Optimization.

In this case, actions becomes a number (e.g. $move = 1$, $rotate = 2$, etc.), and the result are expected to be reasonable. The author believes that regression could lead to problems in massive training datasets, but further investigations are required.

4.1.4 Gaussian Mixture Model

The last algorithm implemented is Gaussian Mixture Model (GMM), which is a unsupervised probabilistic parametric model used for density estimation and clustering. Its structure relies on a weighted sum of Gaussian components (Reynolds, 2009), where each gaussian tries to model a point cloud. Formally, is a weighted sum of K multivariate Gaussian, described as Eq. 4.10.

$$p(x|\lambda) = \sum_{i=1}^{K} \omega_i \; g_i(x) \tag{4.10}$$

Where x is a D-dimensional data vector (D as the number of features), ω is the weights vector and $g(x)$ represents the Gaussian densities. M is the number of components (number of Gaussians) where the points will be clustered. That is, how many Gaussians will be used to model the point cloud.

In a basic algorithm, this is a manually tuned parameter, but there are some

score values to select the optimal number of components e.g. Bayesian Information Criterion (BIC). Each component density $g(x)$ is defined as in Eq. 4.11.

$$g_i(x) = \frac{1}{(2\pi)^{D/2}\ det(\Sigma_i)^{1/2}} exp\left\{-\frac{1}{2}(x-\mu_i)^T\ \Sigma^{-1}\ (x-\mu_i)\right\} \tag{4.11}$$

Where μ_i and Σ_i are the Gaussian parameters for the mean and the covariance matrix respectively. Weights represent, in some way, the influence of a specific component over the whole data, reason for the constraint $\sum_{i=1}^{K} \omega_i = 1$. The parametric model is completely defined by the set $\lambda = \{\omega, \mu, \Sigma\}$. An example of how the number of components affect to its performance, for the case of density estimation, is found on Fig. 4.3

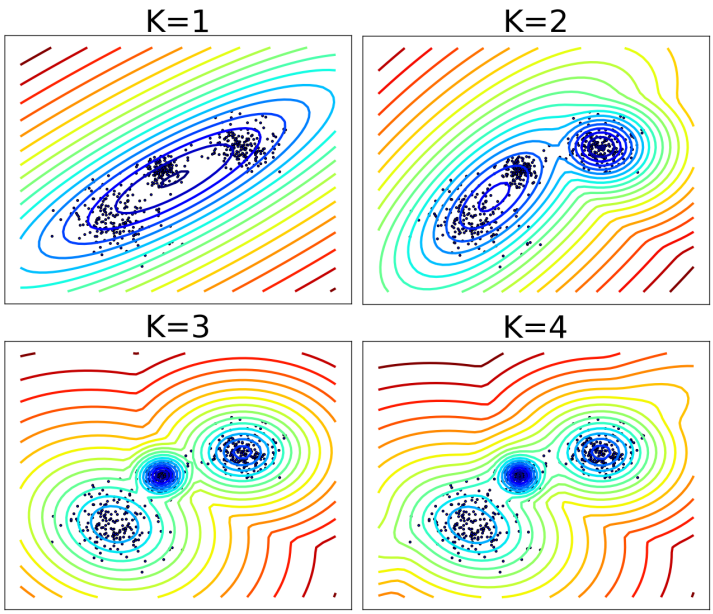

Figure 4.3: *Probability density function of a dataset, with different number of components in the mixture (K = number of components).*

For the covariance matrix, there are some configuration possibilities which modify algorithm performance (Fig. 4.4). One is the "full" mode, where the whole covariance matrix is taken into account in calculations, which implies to

use covariances between features in the final Gaussian configuration. Visually in a 2-dimensional case, Gaussian can look twisted with respect to one axis. Another mode is the “diagonal” one, where only the diagonal values of the matrix are considered, ignoring the rest, that is, using only the features variances. Visually in 2-dimensional case, Gaussian becomes parallel to one of the axis. The last mode is ”spherical”, where we force a diagonal matrix with all values being equal. Visually in 2-dimensional case, Gaussians becomes circular.

Other parameters configuration include the possibility of sharing the covariance matrix among all components. Giving all components the same covariance matrix, all Gaussians look the same in shape, but are placed in different coordinates (this is formally called “tied” parameters).

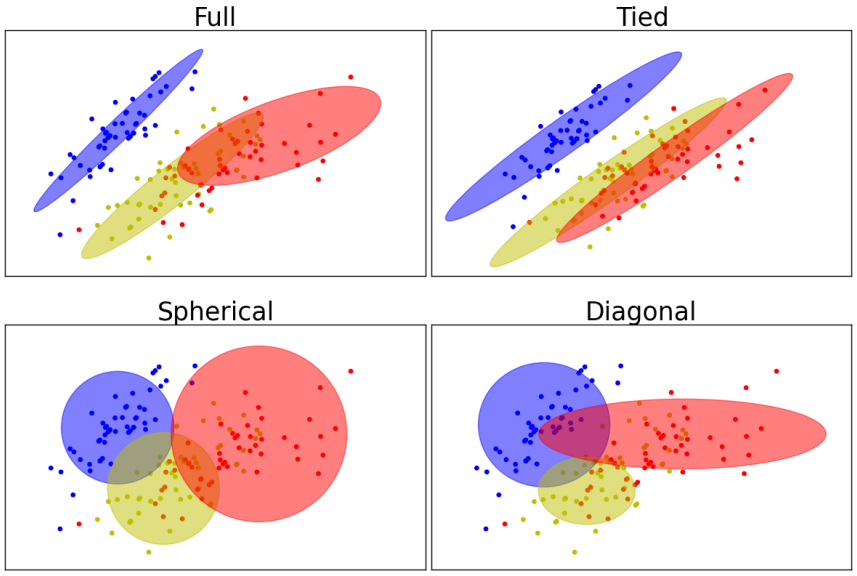

Figure 4.4: *Different mixtures in function of the covariance matrix parameters.*

The above process defined represents the model. But the parameters need to be estimated in a way that they better match the point cloud distribution. The most common used technique is called Maximum Likelihood Estimation (MLE). The MLE quality of the prediction, for T training vectors x, is written as in Eq.

4.12.

$$L(\lambda : x) = \prod_{t=1}^{T} p(x_t|\lambda) \tag{4.12}$$

These parameters may be estimated with an iterative algorithm called Expectation - Maximization. To give an idea of its working, it iteratively modifies mixture parameters to monotonically increase the likelihood of the model. Basic GMM has been tested in experiment chapter, whose results can be checked it that chapter.

As a summary, all these presented algorithms will be asked for a prediction of how an action will affect an object. In next chapter, all of them will show their results.

Chapter 5

Experiments

In this chapter, all the presented algorithms will be tested to evaluate their performance for different tasks. The experiments are divided in three blocks, each one corresponding to a different dataset to be modeled: objects, spatial references and actions.

5.1 Simulated Objects

This experiment, whose formal description can be found on (Victores et al., 2013a), involves a simulated robot, ASIBOT (Jardón, Víctores, Martinez de la casa, Gimenez, & Balaguer, 2012). This robot is an arm, created for assistance purposes. The aim is to close the algorithms to physical machines, or at least, to their limitations. For that, in this experiment, images and descriptions are provided to the system, and then, the queries are asked for drawing.

5.1.1 Datasets

The dataset is composed by synthetically generated images (Fig. 5.1). These images are colored figures, which vary in shape and position, over a black background. Linked to the images, descriptions are attached to each image (only

representative words). To accelerate the process, the initial population is generated automatically. The descriptions are composed by words automatically added when the image is generated.

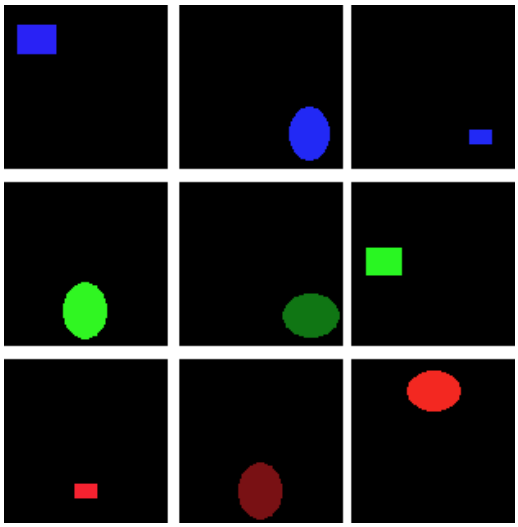

Figure 5.1: *Dataset used for populating the grounding database. For example, the first image is labeled as top-left-dark-blue-fat-straight-box*

The system is trained with 300 images with 7-word descriptions. These words are nouns and adjectives. The training population represents approximately the 13% of all the possible combinations of words. This fact is deliberate, to prove that our algorithms does not need all possibilities to be presented, and that the inference system can manage that situation. Additional Gaussian noise of 1% is added to simulate errors of real captures of data.

In total, 12 features are extracted from images. The first 2 correspond to spatial positions, x and y- Another 2 are used to define the color, hue and value, and finally, 8 features are used to characterize the contour (e.g. convexity, eccentricity, circularness, squareness, etc).

5.1.2 Testing

The final expected result is having the robot drawing an object it has never seen before. But, before that, a mental model is needed. The approach selected in this thesis is to infer features from previous object and feed this information to an Evolutionary Computation Algorithm (EC). That is, to ask semantically for a query, expects the algorithm to create a mental model and then feed EC with this model. The reason why we need EC is that mental model (a vector of scalars of features) is not directly representable graphically. If only inference were the objective, the experiment would end here.

If we consider the habitual three steps of evolutionary algorithms (selection, crossover and mutation), EC performs a Steady State Selection algorithms for the first step, which, for each generation, it replaces only a few of the resulting individuals at a time, while other methods substitute them all. That is, the idea of this selection is that a big set of the individuals will survive to next generation. Steady State steps:

- **Selection**: A tournament (comparison of fitness) is performed between random individuals. The fitnesses are compared and winners (best fitness) are selected for crossover. Fitness is a metric that represents how an individual (with its combination of features) best represents the reference provided, how close are their values.
- **Crossover**: Winners are crossed and their descendant substitute the worst values from the previous tournament.
- **Mutation**: With a certain probability, each child can be mutated.

This process occurs iteratively until a termination condition is accomplished. Termination condition is set to a certain number of generations without improvement in the fitness value. Tournament size is set to 3 individuals, mutation probability is 70% and termination condition is set to 50 generations without improvement.

For generating the mental model, EC controls the coordinated 2D positions of a set of points inside an image. Those points together forms a shape. The features, ofthis generated image, are extracted and the fitness is calculated as the sum of feature differences between the reference data and the generated one. Several generated mental models can be seen in Fig. 5.2. Let us remember that this reference is the vector of values returned from the hyperplanes intersection algorithm.

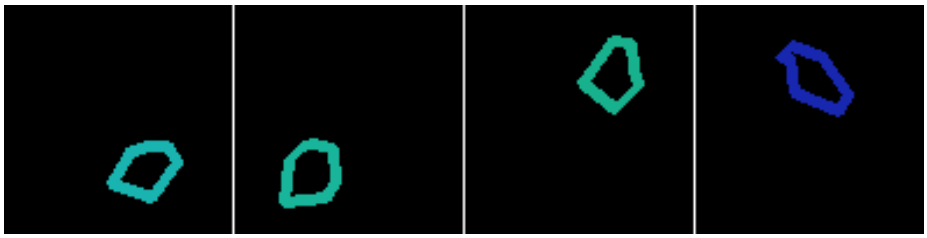

Figure 5.2: *Mental models of the robot when asked for: (a) "bottom right", (b) "bottom left", (c) "top right", (d) "top blue"*

Notice that unnamed features in the query words, such as color or shape, remain uncertain in their values. Despite they can be also modified in the mutation process, the reference values with which they are compared are undefined values from hyperplanes intersection. As we consider that hyperplanes are extended in relevant features, the rest of the features of the vector take intermediate values.

The models generated can be used directly for the drawing application, mapping the shape as a trajectory on a table. Fig. 5.3 depicts a screenshot of the simulated ASIBOT drawing a query.

Apart from the simulator, errors produced by the inference system (before EC) can be calculated. On Table 5.1 there are measured errors induced by the scalability of dimensions. That is, we force the space to have a low number of dimensions, and then we increment them one to one. The features we know are relevant, for the combination of words queried, stays always in all spaces tested.

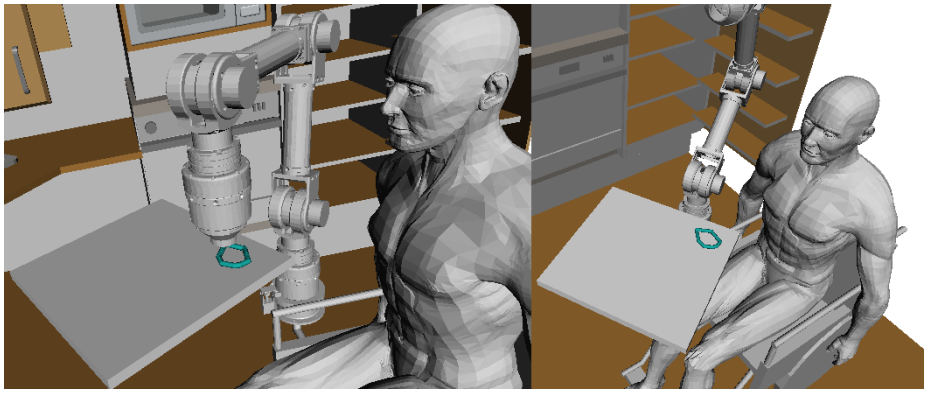

Figure 5.3: *Simulated ASIBOT robot arm drawing the model generated for "bottom left".*

Table 5.1: *Dimensionality errors for the query "top-left".*

Number of Features	2	3	4	5	10	12
Deviation of 1	**0.038**	**0.036**	**0.038**	**0.036**	16.855	16.227
Deviation of 2	**-0.308**	**-0.298**	**-0.298**	**-0.305**	-14.238	-13.541

The inference system does not know how representative a word is for a feature, but a human evaluation can determine that, for instance, "left" word is mostly associated with x coordinate. This idea is exploited by measuring the results for "top-left" query when augmenting the number of features.

Some limitations of this experiment can be listed:

- **Context dependency**: This experiment implementation is not capable of correctly managing words that change depending on the context. That is, words whose perceptions do not remains stable, or similar, for all samples sensed. A possible solution to this problem could be a previous clustering step, to guess how many meanings (clusters) a word has in our space.

- **Misrepresentation**: Some features cannot be completely defined due to color space limitations. For example, in HSV color space (Hue, Saturation, Value) we use, the hue for "red" can be represented as 0 and also as 360.

A hyperplane fitted to these values would lead to unstable, understood as without sense, directions, probably around 180 (which is not red). We have chosen HSV instead of usual RGB (Red, Green, Blue), because RGB cannot describe a color with a single word associated to one of the variables, while HSV indeed can.

- **Extendability**: The assumption is that any point cloud for a word can be extended, while conserving their meaning. This assumption could not be true when accumulating experiences for words (e.g. *mugs* can take many colors, shapes or sizes, which may affect hyperplanes creation).

5.2 Robot Spatial Knowledge

Beyond simulated or synthetic samples, the authors aim to bring embodied information closer to the proposal. The first part of this experiment is a synthetic dataset to test how the algorithms would generalize spatial information, but following it, we present experiments with a real robot.

5.2.1 Synthetic Spatial References

The aim here is to acquire a kind of spatial knowledge by providing spatial samples as reference, which is a means for a human to teach spatial language to a machine. This has more sense when talking about embodied machines, that is, robots. The synthetic tests are usually the first kind of experiments performed when testing machine learning algorithms, because of its ease and quickness in programming, and there will be presented first.

Datasets

The population phase, consists in computer-generated samples representing a white circle in a black background (Fig. 5.4). These circles are randomly dis-

tributed along the image, and they are automatically classified, with semantics, according to their coordinates.

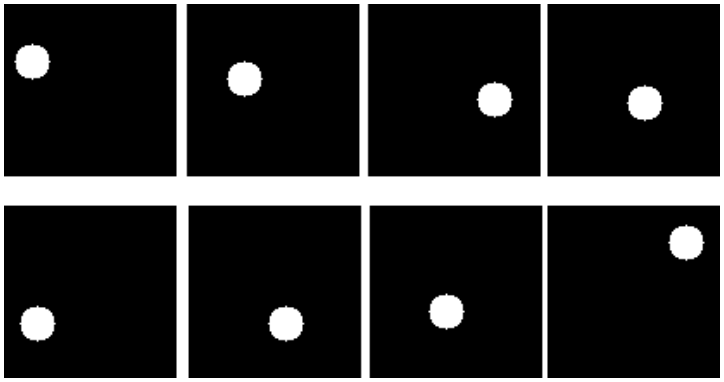

Figure 5.4: *Dataset used for training and test phase in spatial experiment.*

The samples, whose circle coordinates are their features, are labeled with their spatial characteristics. Images are 100x100 pixels size and the classification is carry 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 a double way, because it can represent vertical but also horizontal middle space. For population purposes, 70% of the dataset, composed of 300 samples, is used, and the test phase uses the remaining.

In this case, all possible combinations of words are fed to the grounding database, except those implicitly contradictory (e.g. LEFT-RIGHT).

Testing

The first result, shown on Table 5.2, represents the algorithm numerical output, the coordinates, for the semantic input combination of the words (the ones of its column and row). In this case, the algorithm used is the one previously presented as object generalization and inference, with hyperplanes. The space

Table 5.2: *Training dataset with x, y coordinates output for a 'word, word' input.*

	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

Table 5.3: *RSS result x, y when compared the test samples with algorithm output.*

	TOP	BOTTOM	LEFT	RIGHT	MIDDLE
TOP	105.8 , 37.1	*	**16.5** , **19.9**	**9.4** , **7.2**	342.8 , **31.6**
BOTTOM	*	131.7 , 39.3	**24.5** , **28.8**	**14.2** , **16.5**	190.5 , **20.3**
LEFT	**16.5** , **19.9**	**24.5** , **28.8**	42.5 , 130.2	*	**30.5** , 40.1
RIGHT	**9.4** , **7.2**	**14.2** , **16.5**	*	**23.7** , 85.8	**13.6** , 37.8
MIDDLE	342.8 , **31.6**	190.5 , **20.3**	**30.5** , 40.1	**13.6** , 37.8	134.6 , 155.7

is 2-dimensional.

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. 5.1), once for each coordinate x and y.

$$f(x) = \sqrt{\sum_{i=1}^{n}(x_i - \mu)^2} \tag{5.1}$$

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 previously on Table 5.2) for these words combination. Equation results on Table 5.3[1].

A closer look to the tables reveals some characteristics to be interpreted:

[1]Values with (*) mark correspond to undefined fields because of the absence of samples to compare with (e.g. no 'LEFT-RIGHT' sample).

- The initial automatic sample classification were performed dividing the 100 pixels in three equal parts, so any value lower than 33, or higher than 66, represents a correct parameter generation for unambiguous words (lower than 33 for BOTTOM and LEFT, and higher than 66 for TOP and RIGHT).

- When the word 'MIDDLE' is present, values are misplaced due to their ambiguity, because it is present in both vertical and horizontal.

- Best results (bold numbers) are given when both words are clearly defined and unambiguous (e.g. TOP-RIGHT, BOTTOM-LEFT, etc).

Once tested synthetically, the experiment can go a step further by trying an improved, but similar, experiment with a real robot.

5.2.2 Humanoid Robot TEO Spatial Pointing

The most real experiment has been performed on TEO, the humanoid robot (Monje et al., 2011) (Fig. 5.5). This test is an evolution of the synthetic task of learning spatial positions, but with embodied characteristics.

The objective is to teach TEO spatial 3D references (x, y, $depth$) by showing it a visual marker. After that training phase, we want to ask the robot to point to a semantic asked position. This capability is similar to "imagine" positions in space. The combinations of words may have not been heard by the robot before, like pointing to "FRONT-RIGHT" having already learnt "FRONT" and "RIGHT" separately.

Datasets

As said, this experiment is the most real-world type of the one presented, so the sensorial capabilities are here important. To create a population of samples, a human operator shows to the robot a colored visual marker, and, at the same

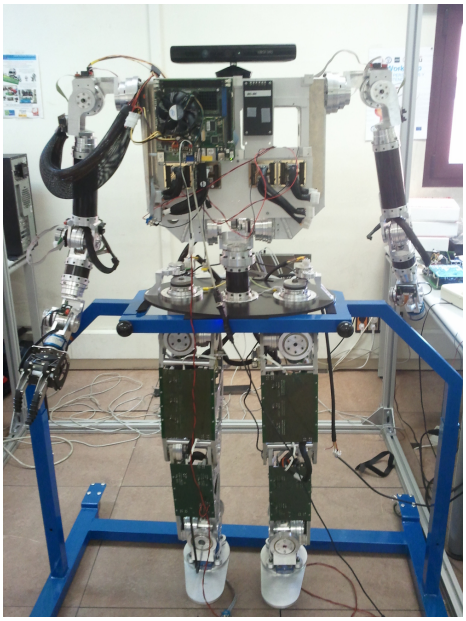

Figure 5.5: *TEO is a full humanoid robot from Robotics Lab research group, Universidad Carlos III de Madrid.*

time, describes the sample (Fig. 5.6). The words used to describe the sample, are all the used in the previous experiment, plus two additional words representing the depth variable (FRONT and BACK).

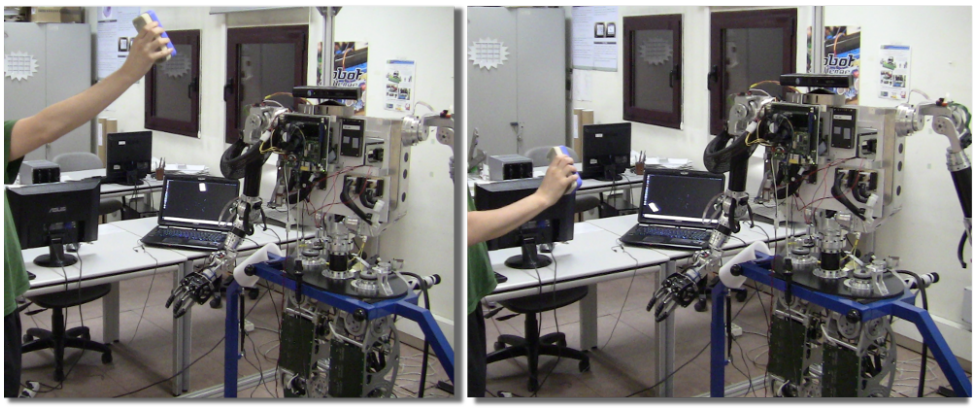

Figure 5.6: *Human operator teaching spatial positions to TEO.*

This initial phase is a semantic grounding process, where the system links the user's words, caught via automatic speech recognition (ASR), with the spatial characteristics of the marker. Robot vision is based on Kinect camera for depth measures and basic OpenCV algorithms to segment marker color (Fig. 5.7).

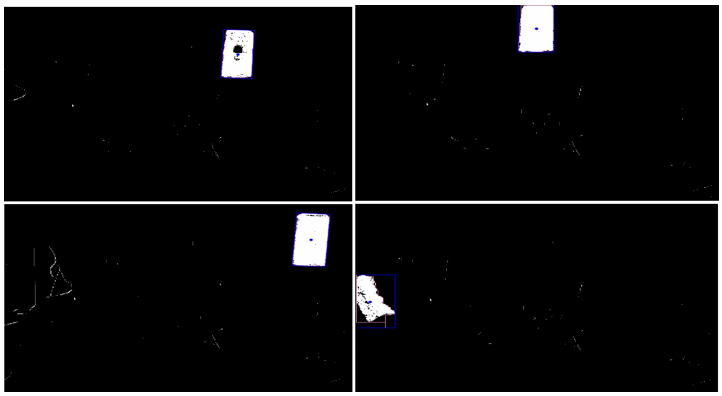

Figure 5.7: *TEO segmentation to detect colored marker.*

Testing

After the training phase, human operator asks TEO, using ASR, to point to a specific position. As a summary, the robot executes the algorithms for hyperplanes intersection, obtains a representative set of coordinates, and moves its arm to this position (Fig. 5.8). Some comments can be made about this experiment:

- The words used to describe spatial positions must be pre-accorded, in order to feed the automatic speech recognition corpus. The corpus is the bag of all words which must be recognized, ignoring the rest. This fact limitates the possibility of further spontaneous teaching.
- Luckily, spatial references must only be learned once, due to the fact that they does not change in time.

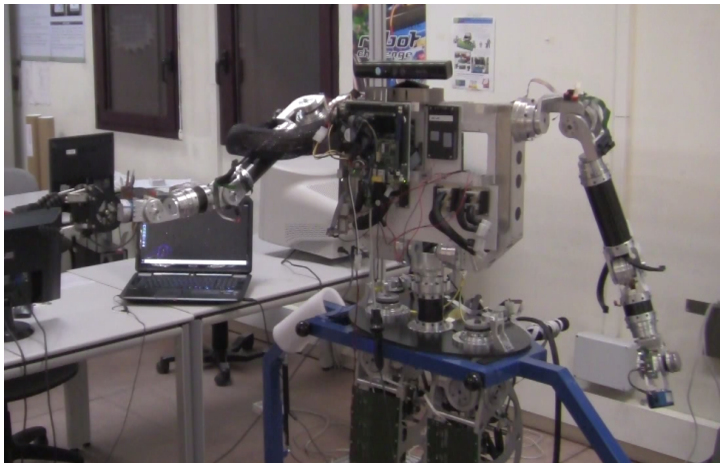

Figure 5.8: *TEO pointing to a semantic queried position.*

5.3 Goal-Oriented Samples

Instead of objects as in previous section, this experiment aims to distinguish how different actions affect different features of objects in the environment (Victores et al., 2013b). The algorithms tested are the ones already formally presented: Arithmetic Mean Model, Direct-Inverse Neural Networks, Support Vector Regression and Gaussian Mixture Model.

5.3.1 Datasets

To compare them, each model is trained with the same experimental dataset. The dataset is composed by 5 "move" frame sequences, and 5 "rotate" frame sequences. The sequences are composed by 7 frames, with a size of 100x100 pixels. White rectangles are situated over a black background (Fig. 5.9). In "move" sequence, the rectangle displaces 60 pixels along the image, with the same rotation. In "rotate" the angle varies 60 degrees, keeping the same coordinates.

An additional 1% of standard deviation noise has been incorporated to each feature x, y and $alpha$, to emulate camera perturbation effects and errors during segmentation.

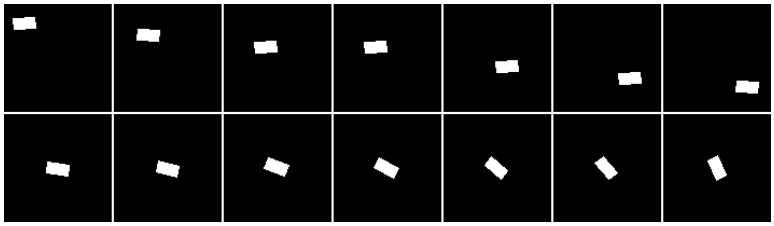

Figure 5.9: *Dataset used for training and test phase for simple actions experiment. Upper sequence is "move", lower is "rotate".*

5.3.2 Testing

As has been previously described, almost all algorithms need parameters to be set manually. Algorithm parameters are tuned as follows:

- **Arithmetic Mean Model**: No special initial tuning is needed.
- **Direct-Inverse Neural Networks**: We have chosen a network with three layers, with 50 neurons in the hidden layer, having a sigmoid function, as activation function. Epochs are fixed to 10, that is, the whole data is used for training 10 times.
- **Support Vector Regression**: Kernel chosen is Radial Basis Function. The penalty weight C is set to 1 and ε (allowed distance to the band) is set to 0.1.
- **Gaussian Mixture Model**: Components are set to 1 (only one Gaussian will fit the data), as we know there is only one point cloud in the space. The covariance matrix is "full", so all covariance matrix is considered in calculations.

On Table 5.4 and Table 5.5 can be found the outputs of the different algorithms for words "rotate" and "move", respectively.

It is important to notice the difference in meaning of the standard deviation for NN and for GMM. In NN, the standard deviation (marked as (*)) represents

Table 5.4: *"Rotate" results.*

	X	Y	ALPHA
MEAN	**0**	-0.2	65.62
D-I NN (*)	0.05 ± 0.26	-0.12 ± 0.1	64.83 ± 0.22
SVR	**0**	**-0.1**	**64.55**
GMM (**)	0 ± 1.41	-0.2 ± 0.74	65.62 ± 2.81

Table 5.5: *"Move" results.*

	X	Y	ALPHA
MEAN	59.8	**59.6**	**-0.26**
D-I NN (*)	59.75 ± 0.2	59.56	-0.74 ± 0.43
SVR	**59.9**	59.1	0.71
GMM (**)	59.8 ± 0.74	$\mathbf{59.6 \pm 1.35}$	$\mathbf{-0.26 \pm 2.8}$

the output of the algorithm, because it is stochastically variable. The reason is the initial random set of network's weights for each training phase (we trained the network once per each one of the 5 times we tested it). NN output presented is the average of running the algorithm 5 times, each with its corresponding training phase. However, in GMM, the standard deviation (marked as (**)) describes input data dispersion in the estimated model. That means, the standard deviation for each axis of the Gaussian.

For a composition of actions, a "move" and "rotate" combination, the error is calculated as an Euclidean Distance (Eq. 5.2) between the reference and the obtained values.

$$f(p, p_0) = \sqrt{(p_0^{(x)} - p^{(x)})^2 + (p_0^{(y)} - p^{(y)})^2 + (p_0^{(alpha)} - p^{(alpha)})^2} \tag{5.2}$$

Where $(p^{(x)}, p^{(y)}, p^{(alpha)} = (60, 60, 60))$ are the references and $(p_0^{(x)}, p_0^{(y)}, p_0^{(alpha)})$ are the composed real values (calculated as a direct sum of values from "move"

Table 5.6: *"Move" plus "Rotate" Euclidean Distance results.*

	EUCLIDEAN DISTANCE
MEAN	5.39
D-I NN	**4.74 ± 0.38**
SVR	5.35
GMM	5.39

and "rotate"). On Table 5.6 the results are presented. This final metric compares algorithms accuracy, obviating other advantages that will be explained later.

The following points highlight some of the benefits and drawbacks of the studied algorithms for this specific synthetic dataset:

- **Arithmetic Mean**: Fast and coherent parameter generator. Incremental version is an extra advantage. However, it lacks a certain degree of flexibility and its results are limited to point clouds which forms a convex set or a quasi-convex set. That means (in a 2-dimensional space where the reader can imagine it) that, concave shapes would lead to a point in the "middle" but empty space, which is non-acceptable.

- **Direct-Inverse Neural Network**: It has proved to work well, even being inversely trained. As a parameter generator, it is the most bio-inspired, which can be also seen in the fact that its results are correct, but not always the same, similar to a biological organism. Also, the possibility to activate simultaneously two input neurons allows a seamlessly combination of action stimulus.

- **Support Vector Regressor**: This regressor has proved to return acceptable results. However, it remains uncertain how a regression algorithm would scale to big training datasets.

- **Gaussian Mixture Model**: Results are equal to Arithmetic Mean, because

of the use of a single component, but other parameters are allowed (e.g. more components, different covariances matrix constraints, tied parameters, etc.). Another interesting capability is the return of a standard deviation model parameter, which, for action parameter generation, can make easier the generation of points, inside a coherent model, in the form of a normal distribution.

With all these experiments, we have tried to cover a wide range of possibilities in machine learning algorithms field, assuming each algorithm limitation. In the final chapter, some general conclusions will be outlined, as well as the main contributions of this thesis.

Chapter 6

Conclusions

Some conclusions have been outlined during experiments chapter, but here the most important contributions of this work will be reviewed, as a whole, and also some future works.

6.1 Contributions of this Work

As a general scope, the bases for future research on robot imagination and inference of objects and actions using machine learning algorithms have been set. More specific contributions are:

1. A system to provide robots with feature generalization abilities, first step for imagination in robots, has been created. Its main characteristics are the construction of hyperplanes with most representative components of tagged point clouds. This is achieved with PCA, to obtain principal components, and hyperplanes creation, to model the meaning of the words in the feature space.

2. A system to provide robots with inference capabilities mixed with semantic queries, improving human robot interaction, has been created. This is

achieved as a search of relevant areas in the feature space, which is obtained as hyperplanes intersection. For objects, the system is able to return the expected features of an object, simply by its description. For actions, the algorithms can generate parameters which represent how the environment is affected by only using as input the action name. For spatial language, some basics embodied references can be taught to robots.

3. All the proposals have been tested with synthetic, simulated and real experiments, with acceptable results, externally recognized by the scientific community through published papers. The algorithms have been applied to different concepts, like objects, actions and spatial language, in order to cover a wide range of possibilities.

6.2 Future Directions

The most interesting future work derived from this thesis, is to apply the knowledge acquired to different fields, maybe with real world closer applications. Obviously more robotic developments can be added to extent its functionalities: context detector using ontologies, objects and actions recognition (not only inference), external and internal robot information mixing in grounding to see how actions affect to the internal state of the robot or multi-modal information (beyond visual). Another pendant work is to generalize actions as feature trajectories of objects in a continuous tracking experiment. In other words, how the object changes through time, when an action is performed on it, beyond the final variation currently measured.

For non robotic developments the author considers that generalization and inference system, with semantic information, can served to interpret big quantities of mixed numerical and semantical data. One example would be Social Networks, where users share personal information, discuss about news and events, etc. All this data could be mixed with profile information (e.g. age, gender,

hours online, etc.) to generalize and categorize users in function of their interests, and also to infer what users group or collectives are interested in certain topics. A similar idea is developed by Deb Roy's company Bluefin Labs[1]. On the same line, this could be also used for interpreting Big Data, to simplify the presentation of information to user in charge of extract relevant information. Another company, Narrative Science[2], is exploiting this idea in conjunction with natural language generators, in order to create coherent texts from numerical data.

[1] https://bluefinlabs.com
[2] http://narrativescience.com

References

Anderson, J. r. (1995). ACT: A simple theory of complex cognition. *Cognitive modeling*, *49*.

Arbib, M., Metta, G., & Smagt, P. V. D. (2008). Neurorobotics: from vision to action. In *Springer handbook of robotics* (pp. 1453–1480). Springer.

Barsalou, L. W. (2008, January). Grounded cognition. *Annual review of psychology*, *59*, 617–45. doi: 10.1146/annurev.psych.59.103006.093639

Barsalou, L. W. (2010, October). Grounded Cognition: Past, Present, and Future. *Topics in Cognitive Science*, *2*(4), 716–724. doi: 10.1111/j.1756-8765.2010.01115.x

Bekkering, H., Wohlschla, A., & Gattis, M. (2000). Imitation of Gestures in Children is Goal-directed Harold Bekkering and Andreas Wohlschla. *The Quarterly journal of experimental psychology. A, Human experimental psychology*, *53*(December 1996), 153–164.

Bonaiuto, J., & Arbib, M. (2010, April). Extending the mirror neuron system model, II: what did I just do? A new role for mirror neurons. *Biological cybernetics*, *102*(4), 341–59. doi: 10.1007/s00422-010-0371-0

Bonaiuto, J., Rosta, E., & Arbib, M. (2007, January). Extending the mirror neuron system model, I. Audible actions and invisible grasps. *Biological cybernetics*, *96*(1), 9–38. doi: 10.1007/s00422-006-0110-8

Brooks, R. (1991, October). Intelligence with representation. *Artificial Intelligence*, *47*, 139–159.

Burgess, C., Livesay, K., & Lund, K. (1998). Explorations in context space: Words, sentences, discourse. *Discourse Processes*, *25*.

Caligiore, D. (2011). *TRoPICALS : A Computational Embodied Neuroscience Model of Compatibility Effects*. Unpublished doctoral dissertation, Universita campus bio-medico di Roma.

Calinon, S., & Billard, A. (2004). Stochastic gesture production and recognition model for a humanoid robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS*, *3*, 2769–2774. doi: 10.1109/IROS.2004.1389828

Calinon, S., & Billard, A. (2005). Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM. *International conference on Machine learning - ICML*, 105–112. doi: 10.1145/1102351.1102365

Calinon, S., & Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. *ACM/IEEE International Conference on Human-robot interaction - HRI*, 255. doi: 10.1145/1228716.1228751

Calinon, S., D'halluin, F., Sauser, E., Caldwell, D., & Billard, A. (2010). Learning and reproduction of gestures by imitation. *IEEE Robotics and Automation Magazine*, *17*(June), 44–54.

Calinon, S., Guenter, F., & Billard, A. (2005). Goal-Directed Imitation in a Humanoid Robot. *IEEE International Conference on Robotics and Automation - ICRA*(April), 0–5.

Chang, C.-c., & Lin, C.-j. (2011). LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 1–39.

Chivers, D. S. (2012). *Human Action Recognition by Principal Component Analysis of Motion Curves*. Unpublished doctoral dissertation.

Coradeschi, S., Loutfi, A., & Wrede, B. (2013, March). A Short Review of Symbol Grounding in Robotic and Intelligent Systems. *KI - Künstliche Intelligenz*, *27*(2), 129–136. doi: 10.1007/s13218-013-0247-2

Cos-aguilera, I., Cañamero, L., & Hayes, G. (2004). Using a SOFM to learn Object Affordances Using a SOFM to Learn Object Affordances. In *Workshop of physical agents.* University of Edinburgh.

Cregan, A. M. (2007). Symbol Grounding for the Semantic Web. In *The semantic web: Research and applications* (pp. 429–442). Springer.

Demiris, Y., & Dearden, A. (2005). From motor babbling to hierarchical learning by imitation : a robot developmental pathway. *Proceedings of the Fifth International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, 31–37.

Demiris, Y., & Khadhouri, B. (2006, May). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, *54*(5), 361–369. doi: 10.1016/j.robot.2006.02.003

Demiris, Y., & Khadhouri, B. (2008, May). Content-based control of goal-directed attention during human action perception. *Interaction Studies*, *9*(2), 353–376. doi: 10.1075/is.9.2.10dem

Dogar, M. R., Cakmak, M., Ugur, E., & Sahin, E. (2007, October). From primitive behaviors to goal-directed behavior using affordances. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 729–734. doi: 10.1109/IROS.2007.4399469

Dominey, P., Metta, G., Nori, F., & Natale, L. (2008, December). Anticipation and initiative in human-humanoid interaction. *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, 693–699. doi: 10.1109/ICHR.2008.4755974

Dua, A. (2000). *Inversion of Neural Networks : A Solution to the Problems Encountered by a Steel Corporation* (Tech. Rep. No. May). MIT.

Eigensatz, M. (2006). *Insights into the Geometry of the Gaussian Kernel and an Application in Geometric Modeling* (Tech. Rep.). Zurich: EPFL.

Elsner, B., & Bernhard Hommel;. (2001). Effect anticipation and action control. *Journal of experimental psychology. Human perception and performance*.

Fagg, A. H., & Arbib, M. (1998, October). Modeling parietal-premotor interactions in primate control of grasping. *Neural networks : the official journal of the International Neural Network Society*, *11*(7-8), 1277–1303.

Faillenot, I., Toni, I., Decety, J., Grégoire, M. C., & Jeannerod, M. (1997). Visual pathways for object-oriented action and object recognition: functional anatomy with PET. *Cerebral cortex*, *7*(1), 77–85.

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., & Sandini, G. (2003). Learning About Objects Through Action - Initial Steps Towards Artificial Cognition. In *Icra ieee international conference on robotics and automation* (pp. 3140–3145). IEEE.

Gallese, V., Fadiga, L., Fogassi, L., & Rizzolatti, G. (1996, April). Action recognition in the premotor cortex. *Brain : a journal of neurology*, *119 (Pt 2*, 593–609.

Gibson, J. J. (1977). The theory of affordances. In *Perceiving, acting, and knowing* (pp. 67–82). Lawrence Erlbaum Associate.

Glenberg, A. M., & Robertson, D. a. (2000, October). Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories

of Meaning. *Journal of Memory and Language*, *43*(3), 379–401. doi: 10.1006/jmla.2000.2714

Goertzel, B., Garis, H. D., Pennachin, C., & Geisweiller, N. (2010). OpenCogBot : Achieving Generally Intelligent Virtual Agent Control and Humanoid Robotics via Cognitive Synergy. In *Icai* (pp. 1–12).

Golub, G. H., & Van Loan, C. F. (1983). *Matrix Computations*. The John Hopkins University Press.

Gräve, K., & Behnke, S. (2012). Incremental Action Recognition and Generalizing Motion Generation based on Goal-Directed Features. In *Iros ieee/rsj international conference on intelligent robots and systems* (pp. 751–757). IEEE.

Greenwald, A. G. (1970). Sensory feedback mechanisms in performance control: with special reference to the ideo-motor mechanism. *Psychological review*, *77*(2), 73.

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

Hasegawa, D., Rzepka, R., & Araki, K. (2009). A Method for Acquiring Body Movement Verbs for a Humanoid Robot through Physical Interaction with Humans. In *Fifth artificial intelligence for interactive digital entertainment conference* (pp. 34–39).

Haykin, S. (2005). Cognitive Machines. In *Ieee international workshop on machine intelligence.* IEEE.

Herzog, G., & Wazinski, P. (1995). *VIsual TRAnslator: Linking Perceptions and Natural Language Descriptions* (Vol. 1). Springer.

Hommel, B., Müsseler, J., Aschersleben, G., & Prinz, W. (2001, October). The Theory of Event Coding (TEC): a framework for perception and action planning. *The Behavioral and brain sciences*, *24*(5), 849–78; discussion 878–937.

Jardón, A., Víctores, J. G., Martinez de la casa, S., Gimenez, A., & Balaguer, C. (2012). 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.

Jeannerod, M., Arbib, M., Rizzolatti, G., & Sakata, H. (1995). Grasping objects : the cortical of visuomotor transformation. *Trends in neurosciences*, *18*.

Johnson, M. (2004). Abstraction in Recognition to Solve the Correspondence Problem for Robot Imitation. In *Towards autonomous robotic systems* (pp. 63–70). Springer.

Kabir, H., Member, S., & Wang, Y. (2008). Neural Network Inverse Modeling and Applications to Microwave Filter Design. *IEEE Transactions on Microwave Theory and Techniques*, *56*(4), 867–879.

Kiesel, A., & Hoffmann, J. (2004, April). Variable action effects: response control by context-specific effect anticipations. *Psychological research*, *68*(2-3), 155–62. doi: 10.1007/s00426-003-0152-7

Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*(2), 211–240. doi: 10.1037//0033-295X.104.2.211

Lenat, D. B., & Guha, R. V. (1989). *Building large knowledge-based systems; representation and inference in the Cyc Project*. Addison-Wesley.

Marocco, D., Cangelosi, A., Fischer, K., & Belpaeme, T. (2010, January). Grounding Action Words in the Sensorimotor Interaction with the World: Experiments with a Simulated iCub Humanoid Robot. *Frontiers in neurorobotics*, *4*(May), 1–15. doi: 10.3389/fnbot.2010.00007

Mavridis, N., & Roy, D. (2003). Coupling perception and simulation: steps towards conversational robotics. In *Iros ieee/rsj international conference on intelligent robots and systems* (Vol. 1, pp. 928–933). IEEE. doi: 10.1109/IROS.2003.1250747

McGuinness, D. L., & van Harmelen, F. (2004). *OWL Web Ontology Language Overview* (Tech. Rep.). W3C.

Metta, G., Sandini, G., Natale, L., Craighero, L., & Fadiga, L. (2006, January). Understanding mirror neurons: A bio-robotic approach. *Interaction Studies*, *7*(2), 197–232. doi: 10.1075/is.7.2.06met

Monje, C. A., Martinez de la casa, S., Jardón, A., Pierro, P., Balaguer, C., & Muñoz, D. (2011). Full-Size Humanoid Robot TEO : Design Attending Mechanical Robustness and Energy Consumption. In *Icra ieee international conference on robotics and automation* (pp. 325–330).

Montesano, L., Lopes, M., Bernardino, A., & Santos-Victor, J. (2008, February). Learning Object Affordances: From Sensory-Motor Coordination to Imitation. *IEEE Transactions on Robotics*, *24*(1), 15–26. doi: 10.1109/TRO.2007.914848

Nakamura, T., Nagai, T., & Iwahashi, N. (2007, October). Multimodal object categorization by a robot. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2415–2420. doi: 10.1109/IROS.2007.4399634

Nalbantov, G., Groenen, P. J. F., & Bioch, J. C. (2005). *Support Vector Regression Basics* (No. 1).

Ngo, H., Luciw, M., Forster, A., & Schmidhuber, J. (2012). Learning Skills from Play : Artificial Curiosity on a Katana Robot Arm. In *International joint conference on neural networks (ijcnn)* (pp. 1–8). IEEE.

Oztop, E., Kawato, M., & Arbib, M. (2006, April). Mirror neurons and imitation: a computationally guided review. *Neural networks : the official journal of the*

International Neural Network Society, *19*(3), 254–71. doi: 10.1016/j.neunet.2006.02.002

Pape, L., Oddo, C. M., Controzzi, M., Cipriani, C., Förster, A., Carrozza, M. C., & Schmidhuber, J. (2012, January). Learning tactile skills through curious exploration. *Frontiers in neurorobotics*, *6*(July), 6. doi: 10.3389/fnbot.2012.00006

Pedregosa, F., Grisel, O., Weiss, R., Passos, A., & Brucher, M. (2011). Scikit-learn : Machine Learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Pfeifer, R., & Bongard, J. (2007). *How the body shapes the way we think*. MIT Press.

Pfeifer, R., & Scheier, C. (1999). *Understanding Intelligence*. MIT Press.

Poppe, R. (2010, June). A survey on vision-based human action recognition. *Image and Vision Computing*, *28*(6), 976–990. doi: 10.1016/j.imavis.2009.11.014

Reynolds, D. (2009). *Gaussian Mixture Models* (No. 2). MIT. doi: 10.1007/978-0-387-73003-5_196

Rochat, M. J., Caruana, F., Jezzini, A., Escola, L., Intskirveli, I., Grammont, F., ... Umiltà, M. A. (2010, August). Responses of mirror neurons in area F5 to hand and tool grasping observation. *Experimental brain research*, *204*(4), 605–16. doi: 10.1007/s00221-010-2329-9

Roy, D. (2002a, July). Learning visually grounded words and syntax for a scene description task. *Computer Speech & Language*, *16*(3-4), 353–385. doi: 10.1016/S0885-2308(02)00024-4

Roy, D. (2002b). A trainable visually-grounded spoken language generation system. In *International conference of spoken language processing*. Citeseer.

Roy, D. (2005, August). Grounding words in perception and action: computational insights. *Trends in cognitive sciences*, *9*(8), 389–96. doi: 10.1016/j.tics.2005.06.013

Roy, D., Hsiao, K.-Y., & Mavridis, N. (2004, June). Mental imagery for a conversational robot. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, *34*(3), 1374–83.

Sandini, G., Metta, G., & Vernon, D. (2007). The iCub Cognitive Humanoid Robot: An Open-System Research Platform for Enactive Cognition Enactive Cognition : Why Create a Cognitive Humanoid. In *50 years of artificial intelligence* (pp. 358–369). Springer.

Schaal, S. (1999, June). Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, *3*(6), 233–242.

Schaul, T., & Felder, M. (2010). PyBrain. *The Journal of Machine Learning Research*, *11*, 743–746.

Schlenoff, C., Prestes, E., Madhavan, R., Goncalves, P., Li, H., Balakirsky, S., ... Migueláñez, E. (2012). An IEEE Standard Ontology for Robotics and Automation. In *Iros ieee/rsj international conference on intelligent robots and systems* (pp. 1337–1342). IEEE.

Smith, L. I. (2002). *A tutorial on Principal Components Analysis* (Tech. Rep.). Cornell University, USA.

Smola, A. J., & Scholkopf, B. (2004). A Tutorial on Support Vector Regression. *Statistics and computing*, *14*(3), 199–222.

Steels, L. (2001). Language Games for autonomous robots. *IEEE Intelligent Systems*, *16*(5), 16–22.

Stoelen, M. F., Bonsignorio, F., Balaguer, C., Marocco, D., & Cangelosi, A. (2012, November). Online learning of sensorimotor interactions using a neural network with time-delayed inputs. *IEEE International Conference on Development*

and Learning and Epigenetic Robotics (ICDL), 1–6. doi: 10.1109/DevLrn.2012.6400857

Stramandinoli, F., Marocco, D., & Cangelosi, A. (2012, August). The grounding of higher order concepts in action and language: a cognitive robotics model. *Neural networks : the official journal of the International Neural Network Society*, *32*, 165–73. doi: 10.1016/j.neunet.2012.02.012

Subramanian, K., & Suresh, S. (2012, December). Human action recognition using meta-cognitive neuro-fuzzy inference system. *International journal of neural systems*, *22*(6), 1250028. doi: 10.1142/S0129065712500281

Tenorth, M., & Beetz, M. (2009). KNOWROB — Knowledge Processing for Autonomous Personal Robots. In *Iros ieee/rsj international conference on intelligent robots and systems* (pp. 4261–4266). IEEE.

Tenorth, M., Clifford Perzylo, A., Lafrenz, R., & Beetz, M. (2012, May). The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments. In *2012 ieee international conference on robotics and automation* (pp. 1284–1289). Ieee. doi: 10.1109/ICRA.2012.6224812

Thill, S., Caligiore, D., Borghi, A. M., Ziemke, T., & Baldassarre, G. (2013, January). Theories and computational models of affordance and mirror systems: An integrative review. *Neuroscience and biobehavioral reviews*, *37*(3), 491–521. doi: 10.1016/j.neubiorev.2013.01.012

Umiltà, M. A., Escola, L., Intskirveli, I., Grammont, F., Rochat, M. J., Caruana, F., ... Rizzolatti, G. (2008, February). When pliers become fingers in the monkey motor system. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(6), 2209–13. doi: 10.1073/pnas.0705985105

Umiltà, M. A., Kohler, E., Gallese, V., Fogassi, L., Fadiga, L., Keysers, C., & Rizzolatti, G. (2001, July). I know what you are doing. a neurophysiological study. *Neuron*, *31*(1), 155–65.

Vernon, D., Metta, G., & Sandini, G. (2007). 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.

Victores, J. G., Morante, S., Jardón, A., & Balaguer, C. (2013a). (Accepted) Towards Robot Imagination Through Object Feature Inference. In *Iros ieee/rsj international conference on intelligent robots and systems.* IEEE.

Victores, J. G., Morante, S., Jardón, A., & Balaguer, C. (2013b). Semantic Action Parameter Inference Through Machine Learning Methods. In *Robocity2030 12th workshop robotica cognitiva.*

Wang, B. Y. Y. (2010). Cognitive Robots: A Reference Model Toward Intelligent Authentication. *IEEE Robotics and Automation Magazine*, *17*(4), 54–62.