2017

Simulation Tools for the Study of the Interaction between Communication and Action in Cognitive Robots

Ferrauto, Tomassino

Simulation Tools for the Study of the Interaction between Communication and Action in Cognitive Robots

by

Tomassino Ferrauto

A thesis submitted to the Plymouth University
in partial fulfillment for the degree of

Doctor of Philosophy

School of Computing, Electronics and Mathematics
ISTC-CNR PhD Node

April 2016

Acknowledgements

The years of my PhD have been full of events. I met a lot of people that helped me, both in my doctoral studies and in everyday life.

I would like to thank my supervisor and my collegues at LARAL, they helped me grow both as a researcher and as a person. I would also like to thank all the people with whom I interacted during these years at ISTC-CNR.

I am also thankful to my Plymouth supervisor and the university staff. I did not spent much time there, but I always received help when I was in need.

I would also like to thank all the participants to the ITALK project. I have fond memories of project meetings, both for the stimulating scientific discussions and the leisure time spent together.

I am grateful to all my friends for the time we spent together and the mostly unconscious help they gave me during the difficult moments.

Finally, I would like to thank my parents Remo and Vincenzina, my sister Federica, my wife Francesca and my children Giordano and Beatrice, for all their help and support.

Abstract

In this thesis I report the development of FARSA (Framework for Autonomous Robotics Simulation and Analysis), a simulation tool for the study of the interaction between language and action in cognitive robots and more in general for experiments in embodied cognitive science. Before presenting the tools, I will describe a series of experiments that involve simulated humanoid robots that acquire their behavioural and language skills autonomously through a trial-and-error adaptive process in which random variations of the free parameters of the robots' controller are retained or discarded on the basis of their effect on the overall behaviour exhibited by the robot in interaction with the environment. More specifically the first series of experiments shows how the availability of linguistic stimuli provided by a caretaker, that indicate the elementary actions that need to be carried out in order to accomplish a certain complex action, facilitates the acquisition of the required behavioural capacity. The second series of experiments shows how a robot trained to comprehend a set of command phrases by executing the corresponding appropriate behaviour can generalize its knowledge by comprehending new, never experienced sentences, and by producing new appropriate actions.

Together with their scientific relevance, these experiments provide a series of requirements that have been taken into account during the development of FARSA. The objective of this project is that to reduce the complexity barrier that currently discourages part of the researchers interested in the study of behaviour and cognition from initiating experimental activity in this area. FARSA is the only available tools that provide an integrated framework for carrying on experiments of this type, i.e. it is the only tool that provides ready to use integrated components that enable to define the characteristics of the robots and of the environment, the characteristics of the robots' controller, and the characteristics of the adaptive process. Overall this enables users to quickly setup experiments, including complex experiments, and to quickly start collecting results.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

This study was carried out in collaboration with the Istituto di Scienze e Tecnologie della Cognizione (ISTC) - Consiglio Nazionale delle Ricerche (CNR), Rome. It was financed by the EU project ITALK from the Framework Programme 7.

This thesis contains works which were the result of collaborations with other researchers. The author contribution to the reported works over the total was about 40% for the work described in chapter 2, 50% for the work described in chapter 3 and 50% for the work described in chapter 4. During the PhD period the author also contributed to works which are not reported in the present thesis.

Relevant scientific seminars and conferences were regularly attended at which work was presented; external institutions were visited for consultation purposes and several papers prepared for publication.

Word count for the main body of this thesis: **32647**

Publications

Ferrauto T., Tuci E., Mirolli M., Massera G., Nolfi S. (2009) Two examples of active categorisation processes distributed over time, *Proceedings of the Ninth International Conference on Epigenetic Robotics (Epirob09)*

Mirolli M., Ferrauto T., Nolfi S. (2010) Categorisation through evidence accumulation in an active vision system, *Connection Science*, (22) 4:331–354.

Tuci E., Ferrauto T., Massera G., Nolfi S. (2010) Co-development of linguistic and behavioural skills: compositional semantics and behaviour generalisation, *Proceedings*

of the 11th International Conference on Simulation of Adaptive Behavior (SAB2010)

Tuci E., Ferrauto T., Massera G., Nolfi S. (2010) The Evolution of behavioural and linguistic skills to execute and generate two-word instructions in agents controlled by dynamical neural networks, *Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems (ALife XII)*

Massera G., Tuci E., Ferrauto T., Nolfi S. (2010) The facilitatory role of linguistic instructions on developing manipulation skills, *IEEE Computational Intelligence Magazine*, (5) 3: 33–42.

Tuci E., Ferrauto T., Zeschel A., Massera G., Nolfi S. (2011) An Experiment on behaviour generalisation and the emergence of linguistic compositionality in evolving robots, *IEEE Transactions on Autonomous Mental Development*, (3) 2: 176–189.

Massera G., Ferrauto T., Gigliotta O., and Nolfi S. (2013) FARSA: An open software tool for embodied cognitive science, *Proceedings of the 12th European Conference on Artificial Life*.

Massera G., Ferrauto T., Gigliotta O. and Nolfi S. (2014) Designing adaptive humanoid robots through the farsa open-source framework, *Adaptive Behavior*, 22(4):255–265.

Attended Conferences, Workshops and Summer schools

The Ninth International Conference on Epigenetic Robotics (Epirob09), Venice, Italy, November 12–14, 2009

ITALK European Project Meetings and Workshops 2010 – 2012

The RobotDoc Research Methods Workshop, University of Plymouth, Plymouth, United Kingdom, March 9–11, 2010

The iCub Summer School (Veni Vidi Vici 2010), Sestri Levante, Italy, July 19–28 2010

The 12th International Conference on the Synthesis and Simulation of Living Systems (ALife XII), Odense, Denmark, August 19–23 2010

The 12th European Conference on Artificial Life (ECAL 2013), Taormina, Italy, Septem-

ber 2–6, 2013

The Ninth International Conference on Swarm Intelligence (ANTS 2014), Brussels, Belgium, September 10–12, 2014

Signed: _____

Date: _____

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Embodied Cognitive Science

The study of cognition is undoubtedly one of the most fascinating areas of science. Historically, the most widely accepted view of cognition has depicted it as the act of manipulating a set of symbols using explicit rules. In cognitivist theories, mind is therefore modelled as an "inner arena" of symbol processing, separated from the external world of meaning and action [1, 25].

The philosopher that is generally considered the most representative of the *duality* between body and mind is René Descartes. One aspect of his thought that might have had a great impact on the cognitivist theory is the discontinuity between human and animals: in particular thought is considered a unique characteristic of the former that is completely absent in the latter. This view is justified by the assumption that sensing and acting in the world do not require thinking, which is identified with higher-order reasoning and abstraction such as those required for language, which is missing in animals [1].

With the appearance of computers, which are basically devices that manipulate symbols, mind and cognitive processes started to be modelled as computers and computer algorithms, respectively. In a conference in 1956 the term *Artificial Intelligence* was coined to indicate an interdisciplinary research field that is interested in understanding biological systems, abstract general principles underpinning intelligent behaviour and applying those principles to build intelligent artefacts (the *synthetic approach* was a distinguishing characteristic of the new discipline). At that time the cognitivist view of

intelligence was mainstream and consequently the efforts of the newly born discipline were concentrated towards studying the high-level abilities of humans, such as playing chess, solving abstract problems and proving mathematical theorems. This approach to artificial intelligence was later termed GOFAI (*Good Old Fashioned Artificial Intelligence*) by the American philosopher Haugeland [50].

The symbolic approach to artificial intelligence has been proven successful in creating many algorithms used in computer software today, like those used in search engines, natural-language interfaces, games, cameras, and many other electronic equipment. Despite these undeniable results, there are also a lot of promises that GOFAI has not been able to fulfill (at least not yet) [90]. There are many human capabilities that are still unmatched by artificial agents, such as natural language processing and image recognition. Although several algorithms have been successfully applied also to these domains, these algorithms typically work only in constrained or simplified environments. Moreover, if one takes into consideration abilities that require a physical interaction with the external environment, e.g. manipulation or locomotion, the distance between biological and artificial systems is even greater. Again, there are many robots that outperform humans or animals in very specific tasks, but none possesses the flexibility of natural systems. More in general, artificial intelligence has proven able to solve problems that we humans find difficult (e.g. playing chess or proving mathematical theorems) while having serious difficulties in tasks we are able to solve with minimal efforts (e.g. seeing, hearing, and walking).

The previous argument brings us to another major problem of GOFAI: it has failed in deepening our understanding of natural intelligence [91]. One paradigmatic example is the *CYC* project [46]. "CYC" stands for en*CYC*lopedia and was an attempt at building a database of "common sense or world knowledge, like knowing that people can read books but books can't read people, or that water flows downhill, or that things that happen later don't cause things that happened earlier."[1]. The problem with this and similar approaches is that for humans this kind of knowledge is strictly tied to the experiences

[1] `http://www.cyc.com/why-cyc/`. Retrieved on January, 30th 2016

that we have of things in the world and this in turn is a consequence of having a physical body [90]. The meaning to symbols employed by such systems needs to be provided by a human being, which acts as the intermediary between the world and "CYC" [1].

The realization of the limitations of the cognitivist approach to intelligence has led, over the past decades, to a significant shift in the study of cognition from purely abstract and symbolic models to *situated* and *embodied* ones. *Situatedness* refers to the fact that biological agents live in a physical environment and to the fact that the actions of the agents modify the environment or their relative position in the environment that in turn influence what they perceive; *embodiment* refers to the fact that agents are physical entities with a given size and shape and to the fact that their physical characteristics strongly influence the behaviour that they exhibit.

In the fields of robotics and artificial intelligence, one of the first researchers who questioned the GOFAI approach was Rodney Brooks [13, 14, 15]. In his works he advocates a radical change in what should be considered "intelligent behaviours" and in the way in which it should be studied. He starts by taking into account the evolutionary history of humans. Human beings descend from simpler life forms that have less sophisticated capabilities and have built upon such capabilities to develop their skills and to reach their current level of intelligence. In light of this, he suggests to study intelligence *bottom-up*: the study of low level abilities is considered necessary to understand high level thought in humans [15].

More generally the *embodied* approach to cognitive science has been discussed in many other fields (such as philosophy, psychology, neuroscience, linguistics, etc. [22, 23, 29, 57, 94, 119, 128]). All these diverse disciplines contributed to identify principles that can be used to model biological systems and to design artificial ones. In [18] the authors identify three key principles that characterize *embodied cognitive science*, namely *morphological computation*, *sensory-motor coordination* and *embodied cognition*.

The realization of the importance of the body and of body-world interactions, lead to

taking into account the possibility to exploit these interactions in order to simplify control policies. Take for example bipedal locomotion. In robotics this is generally considered a complex task, and one that requires a very sophisticated control policy. In has been shown, however, that a carefully designed body can lead even passive structures (i.e. articulated entities without any controller nor actuators) to display "natural" walking behaviour [28, 66]. This is a rather extreme form of *morphological computation*, in which, so to speak, all the computation is performed by the body. More recently researches have exploited this principle to build bipedal robots with minimal actuation and control [27].

Despite the possibility to exploit the body dynamics to reduce the burden on a robot controller has been demonstrated multiple times, the *design* of suitable morphologies still remains a difficult problem. In [18] the authors argue that evolutionary robotics techniques may prove successful in helping a human designer or even in autonomously designing effective robot bodies (for examples of experiments of evolved body morphologies see [3, 106]).

Another fundamental principle exploited by embodied and situated agents is *sensory-motor coordination*. An agent, in general, can only perceive a limited portion of the environment. While this is obviously a limitation, it is still possible to exploit this fact by considering that the agent can co-determine what it perceives through its actions, i.e. it can act so to perceive the appropriate information. One consequence of this is that seemingly hard tasks may be simplified by selecting the relevant information through suitable behaviours. A paradigmatic example is given in [78]. The task involves a Khepera robot that is required to stay near to big cylindrical object and stay away from small cylinders. Given the poor sensory information available to the robot (which only has infrared distance sensors and wheel speed proprioception), trying to solve the task relying only on sensory information is rather difficult, because the stimuli of the big and small cylinder are very similar. When instead sensory-motor coordination is exploited, a simple and robust solution becomes available. When the robot approaches a cylinder

it starts cycling around it and uses the difference between the speed of the left and right wheels to reliably discriminate the size of the cylinder.

The use of actions to extract useful information from the environment is termed *active perception* [4, 5, 83] and the actions whose only purpose is precisely to experience useful sensory inputs are called *epistemic actions* [55].

So far we have shown the influence that having a body has on simple behavioural capabilities, i.e. what Rodney Brooks called *Cambrian Intelligence*. Embodiment and situatedness, however, also influence high level capabilities [87], such as, for example, language acquisition and processing. In this regard neuro-scientific and psychological experiments have shown how human beings performing linguistic tasks are influenced by the action that they perform and by the posture that they assume (for a review, see [35]). Evidence of this strict relation has been found by studies in difference disciplines. For example in a series of experimental psychological studies [42] the authors measured the time it takes for human subjects to start a movement after a linguistic instruction is presented. They have shown how movements that were *congruent* with the sentence being presented (like e.g. moving the arm towards one own body in response to *open the drawer*) started in less time than movements that weren't (like e.g. moving the arm towards one own body in response to *close the drawer*). In neuroscience, several studies have showed how activations in linguistic areas in the brain correlate with activations in areas devoted to motor control. In [51], the analysis of fMRI images of the brain of subjects that were asked to read sentences containing words related to body actions shown how the perception of these words cause the activations of premotor areas that trigger actions afforded by the words (e.g. the perception of a word like pick activates the premotor area that activates the muscles of the arm).

Language acquisition is strongly influenced by the characteristics of the body as well. The study on the acquisition of the first words by 18-month old children presented in [107] has shown the effect of body posture: children are able to learn the name of novel objects also in absence of the object itself if they are looking at the same location

where the object was first presented. These results have recently been reproduced by a robotic model in [73] based on the iCub robot [99]. Finally, the importance of embodiment has also been stressed in developmental psychology theories, e.g. in *Tomasello*'s constructivist theory of language acquisition [121].

In this section we have discussed in general terms the role *embodiment* and *situatedness* in the study of behaviour and cognition. Before concentrating on the importance of tools supporting research in Cognitive Science and their characteristics, in the next section we will focus on one specific aspect of embodiment and situatedness, namely the problem of how symbols can be grounded in the non-symbolic perceptual states and in the sub-symbolic actions that are perceived and produced by embodied agents. This aspects will be investigated by the experiments presented in chapters 2 and 3 of the present thesis.

1.2 Language and the Symbol Grounding Problem

We have seen that one of the main problems of the symbolic approach to artificial intelligence is the "distance" between symbols and the real world. In [47] Steven Harnad precisely defines the terms of this problem, naming it *the symbol grounding problem*. In the paper he asks: "How can the semantic interpretation of a formal symbol system be made *intrinsic* to the system, rather than just parasitic on the meanings in our heads? How can the meanings of the meaningless symbol tokens, manipulated solely on the basis of their (arbitrary) shapes, be grounded in anything but other meaningless symbols?". We have already given an example of symbols whose meaning is "parasitic on the meanings in out heads" when we talked about the CYC project, in the previous section.

Generally, in traditional AI systems, symbols are explained in terms of other symbols, without any connection to the external world [90]. A famous challenge to this approach was set forth by the American philosopher John Searle, with his "Chinese room argument" [104]. A common assumption in GOFAI is that a system that passes the *Turing Test* (named after its proponent, Alan Turing) can be assumed to be as intelligent as a

human. The test involves a human player who communicates through a text interface to another agent (who could be a human or a machine). If the player is not able to distinguish the human from the machine, then the machine has passed the test. In this case, it is generally assumed that the machine should "understand" what is being told to it in the same way as a human does.

In his mental experiment, Searle imagines a Turing Test performed using the Chinese language and a machine that is able to pass the test. Then he supposes that he himself takes the part of the machine: he receives Chinese text and then, applying the same rules the "intelligent" machine would use, he outputs an answer in Chinese. Searle, however admits that he does not speak Chinese, thus he does not "understand" what the conversation is about – and so neither can the machine. When an external observer who knows Chinese looks at the answers of the system, they may make sense for him, but again, as in the CYC example, meaning is *extrinsic* to the symbol system and thus pure symbol manipulation cannot be a valid model for cognition.

Since the publication of the paper of Harnad in 1990, many researchers within Embodied Cognitive Science attempted to tackle the symbol grounding problem. In 2008 Luc Steels wrote a paper entitled "The symbol grounding problem has been solved, so what's next?" [112] in which he claimed that the problem has been settled by his experiments on *language games* [113, 114, 115, 116] in which a population of robots develop on the fly a language system and use such language to successfully communicate about an external world. For example, in [115] a population of robots autonomously acquires a shared lexicon that identifies different objects. The experiment is made up of several trials (games) involving two randomly selected robots. One of the agents plays the role of the *speaker*, while the other is the *hearer*.

The speaker randomly selects an object or a *topic* in the environment and tries to draw the attention of the hearer to the same object by saying a word or a sentence (which is randomly generated the first time). The hearer tries to recognize the object on the basis of previously experiences. If the game fails the speaker indicates the object and

the hearer stores associations between the perceived image and the heard word or sentence. By repeated executions of these games the whole population develops a shared lexicon and also internal categories that are connected to the objects' perception. In this sense, the resulting symbols (i.e. words) are *grounded* in the sensory experience of the agents.

In subsequent works, the same principles have been applied to show the emergence of high level linguistic structures. For example in [10] the authors have reported the results of experiments with artificial agents performing language games in which what emerged were grammatical structures, such as small markers to indicate the gender, number or the animate nature of a word associated to an object. This mimics what happens in various languages where adjectives take different forms depending on properties of the noun to which they refer. In the paper it is shown how such structures are very useful to reduce the cognitive burden on the hearer and how they could evolve from meaningful words (i.e. grounded words).

Despite these experiments constituting a significant progress in our ability to model how language evolves and how linguistic symbols are grounded on sub-symbolic states, the way in which the symbols of these robots are grounded on sub-symbolic states does not agree very well with the way in which symbols in humans are grounded in human experiences. More specifically, in those works words are associated to specific sensory patterns, but that may not be sufficient for an effective grounding. An example taken from [1] clarifies what is missing: "Grounding the symbol for 'chair', for instance, involves both the reliable detection of chairs, and also the appropriate reactions to them. [...] Thus is it possible for someone to ask, presenting a tree stump in a favourite part of the woods, "Do you like my reading chair?" and be understood. An agent who has grounded the concept 'chair' can see that the stump is a thing for sitting, [...]. Simply having stored the fact that a chair is for sitting is surely not sufficient ground for this latter capacity. The agent must know what sitting is and be able to systematically relate that knowledge to the perceived scene, and thereby see what things (even if

non-standardly) afford sitting."

What is clear from the cited example is that a key role in symbol grounding is played by *affordances* (in the sense of Gibson [40]), which are, in turn, strictly connected to the agent's *behavioural* capabilities. In this regard, the psychological and neuroscientific evidences of the strict relation between action and language that we have already discussed in the previous section (e.g. [42, 51]) are yet another indication of the fact that associating symbols to sensory patterns alone is not enough to explain the grounding capabilities of human being.

In some works the possibility to observe the emergence of deeper forms of grounding has been investigated [45, 80, 117]. In particular in [45] the authors described an experiment in which a population of robots that had the possibility to produce and detect sounds were evolved for the ability to perform a cooperative task. The agents are rewarded to stay inside different areas (i.e. one robot in the white area and one in the black area at the same time) and to switch areas as quickly as possible. A bidirection communication channel is present that allows the exchange of a single numeric value between 0.0 and 1.0 at each time step. The analysis of the evolutionary experiments indicates that the robot evolve an ability to produce and understand symbolic signals to effectively cooperate. The analysis of the way in which the signals are produced indicates that they are grounded not simply in sensory states but also on specific behaviour capabilities.

Another robotic experiment that explored the links between language grounding and behaviour is [117], in which a mobile robot is taught to respond appropriately to two-word instructions using a supervised learning technique. The experiment will be described in more details in chapter 3. What is interesting to note here is that, as in the previous example, there is no separation between a "linguistic process" and a "semantic process", both being part of a single dynamical system. In this dynamical system view of cognition, thus, the main cause of the existence of the grounding problem (i.e. the distance between symbols and their meaning) is removed.

1.3 Tools for Embodied Cognitive Science

Embodied Cognitive Science addresses the study of embodied and situated agents and, in some cases, the study of how these agents develop their capabilities autonomously while interacting with their physical and (eventually) social environment. For many years, these studies have been confined to relatively simple agents and tasks. Recent research, however, demonstrated how this method can be extended to studies that involve agents with complex morphologies and rich sensory–motor systems mastering relatively hard tasks (see for example [6, 64, 95, 97, 101, 124, 137]).

From a modelling point of view complexity does not represent a value in itself. Indeed, the Occam's razor argument claims that given two explanations of the data, all other things being equal, the simpler explanation is preferable. After all, one of the key contribution of adaptive behaviour research consists in the demonstration of how complex abilities can emerge from the interactions between relatively simple agents and the environment. On the other hand, the modelization of a given phenomenon necessarily requires the inclusion of the characteristics that constitute key aspects of the targeted objective of study. In some cases, therefore, the use of complex agents and/or tasks is necessary. For example, the modelization of the morphological characteristics and of the articulated structure of the human arm constitutes a prerequisite for modelling human object manipulation skills. Likewise, the use of agents provided with rich sensory systems constitutes a necessary prerequisite for modelling sensory integration and fusion.

From a methodological point of view the Embodied Cognitive Science approach to the study of cognition implies that models of behavioural and cognitive capacities should take into consideration the characteristics of the agent's nervous system, of the agent's body, of the environment as well as the properties that originate from the interaction between these three components. This in turn requires the formulation of models that are far more complex than their previous disembodied counterpart and that are not constituted simply by static descriptions but rather by processes that run in the physical

world or in realistic computer simulations.

This new approach to cognitive modelling brought about the necessity to validate working hypotheses on a real device. Robots are the natural candidates, as they, like living beings, have a physical body and can act in a physical environment. Nowadays there are several robotic platforms which are more or less affordable yet complex enough to be useful tools for cognitive scientists, such as the Khepera[2] and the Nao[3] robots. There are also many robots which have been developed during research projects (such as the MarXbot[4] and the iCub[5] robots) or are being developed at the time of writing (such as Roboy[6]).

The new approach also requires the usage of sophisticated software tools. Some of these tools are needed to enable the robotic model to operate. Others are required to carry out experiments in simulations. Some models, in fact, are hard or impossible to test on real robots, since the training phase of the robot would take too long or the robots might damage themselves during long lasting operations or executing actions during exploration. The cognitive models that will be presented in the chapters 2 and 3 of this thesis constitute an example. In fact they could have not been carried out entirely on hardware exactly for the two reasons discussed above.

The possibility to create computer simulations of robotic experiments has been greatly facilitated by the availability of libraries to simulate rigid body dynamics such as ODE[7] [108], Newton Dynamics[8] [53] and Bullet Physics Library[9]. However, implementing experiments through the usage of these libraries still requires a substantial amount of work. Indeed, simulating the body of the robotic agents and the environment constitutes only one of the components that need to be implemented in order to carry on an embodied experiment. Other necessary components typically include:

[2] http://www.k-team.com/mobile-robotics-products/khepera-ii
[3] http://www.aldebaran-robotics.com/en/
[4] http://mobots.epfl.ch/marxbot.html
[5] http://www.icub.org/
[6] http://roboy.devanthro.com/
[7] http://www.ode.org/
[8] http://newtondynamics.com/forum/newton.php
[9] http://bulletphysics.org/wordpress/

- the sensors and the actuators of the agent. Some of them can be implemented using low level functions from the physic simulation libraries, others require specific code (for example to simulate the communication between robots);
- the controller of the agent, e.g. the agent's neural controller;
- the learning and/or adaptive process.

Overall this implies that the knowledge barrier that Embodied Cognitive Science researchers should face to build and analyse their models is still very high.

FARSA (Framework for Autonomous Robotics Simulation and Analysis) aims at mitigating this problem. It is an open-source software tool that enables researchers and students to easily and effectively carry out research in Embodied Cognitive Science [62, 63]. FARSA combines the following features in a single framework:

- it is open-source, so it can be freely modified, used and extended by the research community;
- it is constituted by a series of integrated libraries that allow to easily design the different components of an embodied model (i.e. the agents' body and sensory-motor system, the agents' control systems, and the ecological niche in which the agents operate) and that allow to simulate accurately and efficiently the interactions between the agent and the environment;
- it comes with a rich graphical interface that facilitates the visualization and analysis of the elements forming the embodied model and of the behavioural and cognitive processes originating from the agent/environment interactions;
- it is based on a highly modular software architecture that enables a progressive expansion of the tool features and simplifies the implementation of new experiments and of new software components;
- it is multi-platform, i.e. it can be compiled and used on Linux, Windows, and Mac OS X operating systems;

- it comes with a set of illustrative experiments, that can be used as a base for running a large spectrum of new experiments, and with a synthetic but comprehensive documentation that should enable users to quickly master the tool usage;
- it allows to use both fast low-accuracy static simulation techniques and slower high-accuracy dynamic simulation techniques. It permits to improve the simulation speed by avoiding the usage of the graphic visualization when it is not needed (e.g. during training processes). It also permits to increase speed through the usage of multi-threads simulations running on multi-core computers or on computer clusters.

The tool will be extensively described in chapter 4

1.4 Thesis structure

This thesis is organized as follows. Chapters 2 and 3 describe experiments on the relation between language and action in adaptive embodied and situated agents, i.e. in robots that develop their cognitive and behavioural skills autonomously while they interact with their physical and social environment. Along with the discussion of the scientific relevance of the experiments, they also contain an analysis of the characteristics of the tools that were employed to actually perform them.

Chapter 4, describes FARSA, the open source software tool that I developed together with Gianluca Massera, Onofrio Gigliotta and Stefano Nolfi, that allows to easily set up and carry out embodied cognitive science experiments on different simulated robotic platforms. The tool was built taking into account the requirements gathered while working on the experiments described in the previous two chapters, as well as the useful feedback from other researches in the LARAL laboratory, working on different experimental setups. The chapter will also discuss how the developed tool might support research in cognitive science, its limitation and the developments that are necessary

Chapter 2

The Facilitatory Role of Labels on the Development of Manipulation Skills

This chapter presents a series of experiments that aim to investigate whether the availability of labels provided by a caretaker facilitates the acquisition of object manipulation abilities in a simulated humanoid robot [64].

The robot, which is provided with a neural network controller, is trained for the ability to manipulate spherical objects located on a table by reaching, grasping, and lifting them. The controller of the robot is trained through an adaptive process in which the connection weights of the robot's controller that regulate the fine-grained interaction between the robot and the environment are varied, and in which variations are retained or discarded on the basis of whether they lead to increased performance or not.

The experiments have been replicated in two experimental conditions. In a first experimental condition the robot only perceives the relative position of the target object through its visual system and the current position of its arm through its propriosensors. In the second experimental condition the robot also perceives an input provided by a caretaker that indicates the action that the robot should exhibit in each successive phase in order to perform the task. These *labels* for actions can be interpreted as a very primitive form of language.

The obtained results shown that the presence of such labels facilitates the development of the required behavioural skills.

The chapter is organized as follows. Section 2.1, reviews the literature that discusses

the cognitive tool function of language, i.e. the fact that language can support the development of behavioural and cognitive skills. In this section we also review some recent research that investigated this aspect through the usage of embodied simulation experiments. Section 2.2, briefly reviews the literature on the development of manipulation skills in humanoid robots. Section 2.3, describes the experimental setup and section 2.4 the obtained results. Finally section 2.5 discusses the significance of the obtained results and section 2.6 describes the influence of the experiments in the present chapter on the development of the FARSA robotic simulator..

2.1 Language as a cognitive tool

It is evident from everyday life that language plays a fundamental role for human beings. This capability is probably the most evident distinctive trait of humans when compared with the other animals. Most, if not all, animals communicate. However, no species posses a language with a complexity and a richness analogous to human language. Such a powerful tool has a great impact on social life. With language humans can give or receive commands, coordinate with peers during the execution of complex tasks, inform each other, make plans with others and more.

In all the examples we have just given, language is used as a powerful communication tool. It has been shown, however, that the presence of language has beneficial effects on cognition in general [70, 85]. It is even possible that the rich "mental world" in which human beings live would not exist at all if we didn't have language. There are many capabilities that are influenced and enhanced by the presence of language. Let me consider, for example, the ability to categorize (which is considered one of the most fundamental cognitive process [48]). In [69] the authors have presented an artificial neural network experiment to verify to which extent language influence categorical perception. What they found is that the categories acquired by the neural network are influenced by the presence or the absence of accompanying linguistic stimuli. In the presence of linguistic stimuli, the categories were more separated in the network internal space. This means that instances of the same category were perceived as "more

similar" and that instances of different categories were perceived as "more dissimilar".

The role of language as a tool that augments human capabilities has been recognized by various authors. The soviet psychologist Lev Vygotsky was one of the first who pointed out this role of language. In his works [133, 134] he investigated two aspects of child development: private speech and scaffolded actions. Often children receive help by caretakers to execute difficult tasks. The term scaffolded actions thus refer to the actions that rely on the help provided by the caretaker. This help from childhood on is often provided through linguistic instructions that guide the actions of the child. Eventually, the child then starts to perform the actions by speaking to her/himself, i.e. by substituting the language provided by the caretaker with its own self-produced language. As a result of this process, as argued by Vygotsky, language (eventually in the form of an internal language) keeps supporting action production even into adulthood.

Despite the work of Vygotsky dating back to more than half a century ago, most of the subsequent studies on language have focused on its communicative role. It is only relatively recently that the interest in language as a cognitive tool has risen again (see for example [20, 24, 30, 70]). Clark [24] discusses six domains in which language supports the execution or the acquisition of behavioural and cognitive capabilities: (i) memory augmentation, (ii) environmental simplification (e.g. the use of labels to provide perceptually simple clues), (iii) coordination and reduction of on-line deliberation, (iv) taming path dependent learning (i.e. structuring the learning process in a pedagogical effective way), (v) attention and resource allocation, (vi) data manipulation and representation. Overall this implies that language deeply affects the cognitive and computational space in which we live.

More recently in [70], the authors proposed to apply the aforementioned principles to robotics. According to the authors, a proper exploitation of the cognitive-tool role of language could enable robotics to properly address problems that are significantly behind the state of the art in robotics, which for the moment is only able to appropriately address low-level cognitive phenomena.

Section 2.3 will describe a series of experiment in which we demonstrate, for the first time, that the availability of labels, modelling a very primitive form of linguistic inputs, produced by a caretaker, facilitates the development of manipulation skills in a simulated humanoid robot that is trained through an evolutionary algorithm.

2.2 Development of manipulation skills in humans and robots

The control of arm and hand movements in human and nonhuman primates and in robots is a fascinating research topic actively investigated within several disciplines including psychology, neuroscience, and robotics. Modelling in detail the mechanisms underlying arm and hand movement control in humans and primates and building robots able to display human-like arm/hand movements still represents an extremely challenging goal [102]. Indeed, the development of robots with the dexterity and robustness of humans still constitutes a long-term goal [41], despite the progress achieved in robotics so far. These difficulties can be explained by considering the need to take into account the role of several aspects including: the morphological characteristics of the arm and of the hand, the bio-mechanics of the musculoskeletal system, the presence of redundant degrees of freedom and limits on the joints, non-linearity (e.g., the fact that small variations in some of the joints might have a strong impact on the hand position), gravity, inertia, collisions, noise, the need to rely on different sensory modalities, visual occlusion, the effects of movements on the next experienced sensory states, the need to coordinate arm and hand movements, the need to adjust actions on the basis of sensory feedback, and the need to handle the effects of the physical interactions between the robot and the environment. The attempt to design robots that develop their skills autonomously through an adaptive process permits, at least in principle, to delegate the solutions to some of these aspects to the adaptive process itself.

In work reported in the following sections we take into account most of the aspects discussed above, although often by introducing severe simplifications. More specifically, the morphological characteristics of the human arm and of the hand are taken into account by using the iCub robot, that reproduces approximately the morphological

characteristics of a 3.5 years-old in term of size, shape, articulations, degrees of freedom and relative limits [100]. Some of the properties of the musculoskeletal system have been incorporated into the model by using muscle-like actuators controlled by antagonistic motor neurons. For the sake of simplicity, the segments forming the arm, the palm, and the fingers are simulated as fully rigid bodies. However, the way in which the fingers are controlled, enable a certain level of compliance in the hand. The role of gravity, inertia, collision, and noise are taken into account by accurately simulating the physics laws of motion and the effect of collisions (see Section 2.3 for details of the model).

One of the main characteristics of the model presented in the following section is that the robot controller adjusts its output on the basis of the available sensory feedback by updating directly the forces exerted on the joints (see [103] for related approaches). The importance of the sensory feedback loop has been emphasised by other work in the literature. For example in [34] the authors describe an experiment in which a three-fingered robotic arm displays reliable grasping behaviour through a series of routines that keep modifying the relative position of the hand and of the fingers on the basis of the current sensory feedback. The movements tend to optimise a series of properties such as hand-object alignment, contact surface, finger position symmetry, etc.

In my experiment, the characteristics of the human brain that processes sensory and proprio-sensory information and control the state of the arm/hand actuators are modelled very loosely through the use of dynamical recurrent neural networks. The architecture of the artificial neural network employed is not inspired by the characteristics of the neuroanatomical pathways of the human brain. Also, many of the features of neurons and synapses are not taken into account (see [84], for an example of work that emulates some of the anatomical characteristics of the human brain). The use of artificial neural networks as robot controller provides several advantages with respect to alternative formalisms, such as robustness, graceful degradation, generalisation and the possibility to process sensory-motor information in a way that is quantitative both

in state and time. These characteristics also make neural networks particularly suitable to be used with a learning/adaptive process in which a suitable configuration of the free parameters is obtained through a process that operates by accumulating small variations.

Newborn babies display a rough ability to perform reaching. This initial capability evolves into effective reaching and grasping skills by 4/5 months, into an adult-like reaching and grasping strategies by 9 months, up to precision grasping by 12/18 months [130, 131, 132]. Concerning the role of sensory modalities, the experimental evidence collected on humans indicates that young infants rely heavily on somatosensory and tactile information to carry out reaching and grasping action and they use vision to elicit these behaviours [96]. However, the use of visual information (employed to prepare the grasping behaviour or to adjust the position of the hand by taking into account the shape and the orientation of the object) starts to play a role only after 9 months [65]. For this reason the robot is provided with proprioceptive and tactile sensors and with a vision system that provides information on the position of the object only. Moreover, visual occlusions are not simulated assuming that the information concerning the position of the object can be inferred in relatively reliable way even when the object is partially or totally occluded by the robot's arm and hand.

In accordance with the empirical evidence indicating that early manipulation skills in infants are acquired through self-learning mechanisms rather than by imitation learning [84], the robot acquires its skills through a trial and error process during which random variations of the free parameters of the robots' neural controller (which are initially assigned randomly) are retained or discarded on the basis of their effect at the level of the overall behaviour exhibited by the robot in interaction with the environment. More precisely, the effect of variations is evaluated using a set of utility functions that determine the extent to which the robot manages to reach and grasp the target object with its hand, and the extent to which the robot succeeds in lifting the object over the table. The use of this adaptive algorithm and utility functions leaves the robot free to

discover its own strategy to reach the goals set by the experimenter. This in turn allows the robot to exploit sensory-motor coordination (i.e., the possibility to act in order to later experience useful sensory states) as well as the properties arising from the physical interactions between the robot and the environment. In [123] it is shown how this approach allows the robot to distinguish objects of different shapes by self-selecting useful stimuli through action, and in [61] it is shown how this approach allows for the exploiting of properties arising from the physical interaction between the robot body and the environment for the purpose of manipulating an object.

2.3 Methods

The experiments involve a simulated humanoid robot that is trained to manipulate a spherical object located in different positions over a table placed in front of the robot by reaching, grasping, and lifting it. More specifically the robot is made up of an anthropomorphic robotic arm with 27 actuated degrees of freedom (DOF) on the arm and hand, 6 tactile sensors distributed over the inner part of the fingers and palm, 17 propriosensors encoding the current angular position of the joints of the arm and of the hand, a simplified vision system that detects the relative position of the object (but not the shape of the object) with respect to the hand and 3 sensory neurons (labels) that encode the category of the elementary behaviours that the robot is required to exhibit (i.e., reaching, grasping, or lifting the sphere). The neural controller of the robot is a recurrent neural network trained through an evolutionary algorithm for the ability: (i) to reach an area located above the object, (ii) to wrap the fingers around the object, and (iii) to lift the object over the table. The condition in which the labels are provided has been compared with the condition in which the labels are not provided. For each condition, the evolutionary process has been repeated 10 times with different random initialisations. The robot and the robot/environmental interactions have been simulated by using a very preliminary version of the FARSA tool, that will be described in Chapter 4.

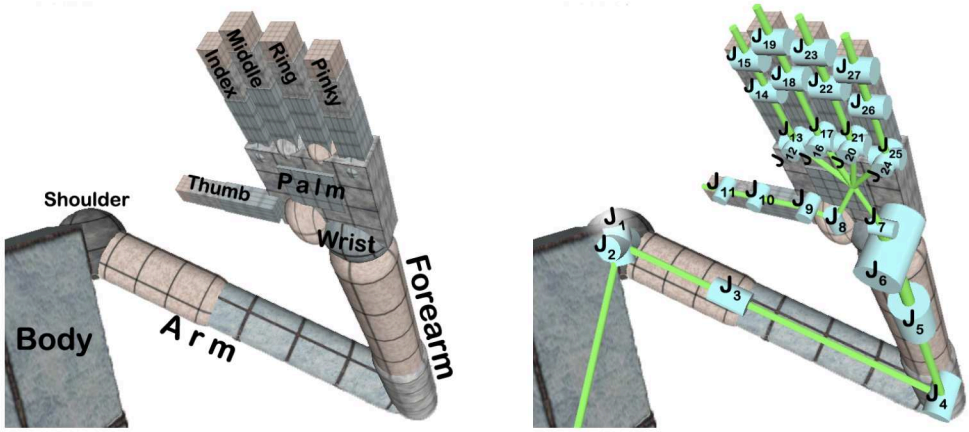


Figure 2.1: (a) The robot structure and (b) its kinematic chain. Cylinders represent rotational DOFs where its main axis indicate the corresponding axis of rotation; the links amongst cylinders represents the rigid connections that make up the arm structure.

2.3.1 The Arm

The arm consists mainly of three elements (the arm, the forearm, and the wrist) connected through articulations placed into the shoulder, the arm, the elbow, the forearm and wrist (see figure 2.1).[1]

The joints J_1, J_2 and J_3 provide *abduction/adduction*, *extension/flexion* and *supination/-pronation* of the arm in the range $[-140°, +100°]$, $[-110°, +90°]$ and $[-110°, +90°]$, respectively. These three degrees of freedom (DOFs) acts like a ball-and-socket joint moving the arm in a way analogous to the human shoulder joint. J_4, located in the elbow, is a hinge joint which provides *extension/flexion* within the $[-170°, +0°]$ range. J_5 twists forearm providing *pronation/supination* of the wrist (and the palm) within $[-100°, +100°]$. J_6 and J_7 provide *flexion/extension* and *abduction/adduction* of the hand within $[-40°, +40°]$ and $[-100°, +100°]$ respectively (see figure 2.1).

The arm joints $(J_1, \ldots, J_7)$ are actuated by two simulated antagonist muscles implemented accordingly to Hill's muscle model [98, 105]. More precisely, the total force exerted by a muscle is the sum of three forces $T_A(\alpha, x) + T_P(x) + T_V(\dot{x})$ which depend on the

[1] Details about arm and hand dimensions are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

activity of the corresponding motor neuron (α) on the current elongation of the muscle (x) and on the muscle contraction/elongation speed ($\dot{x}$) which are calculated on the basis of the following equations:

$$\begin{aligned} T_A &= \alpha\left(-T_{max}\left(\frac{x-R_L}{L_{max}-R_L}\right)^2+T_{max}\right) \\ T_P &= T_{max}\frac{exp\left\{K_{sh}\left(\frac{x-R_L}{L_{max}-R_L}\right)\right\}-1}{exp\{K_{sh}\}-1} \\ T_V &= b\cdot\dot{x} \end{aligned} \tag{2.1}$$

where L_{max} and R_L are the maximum and resting lengths of the muscle, T_{max} is the maximum force that can be generated, K_{sh} is the passive shape factor, and b is the viscosity coefficient.

The active force T_A depends on the activation of muscle α and on the current elongation/compression of the muscle. When the muscle is completely elongated/compressed the active force is zero regardless of the activation α. At the resting length R_L, the active force reaches its maximum that depends on the activation α. The red curves in figure 2.2 show how the active force T_A changes with respect to the elongation of the muscle for some possible values of α. The passive force T_P depends only on the current elongation/compression of the muscle (see the blue curve in figure 2.2). T_P tends to elongate the muscle when it is compressed less than R_L and tends to compress the muscle when it is elongated above R_L. T_P differs from a linear spring for its exponential trend that produces a large opposition to muscle elongation and little to muscle compression. T_V is the viscosity force. It produces a force proportional to the velocity of the elongation/compression of the muscle.

The parameters of the equation are identical for all 14 muscles controlling the seven DOFs of the arm and have been set to the following values: $K_{sh} = 3.0$, $R_L = 2.5$, $L_{max} = 3.7$, $b = 0.9$ with the exception of parameter T_{max} which is set to $3000N$ for joint J_2, $300N$ for joints J_1, J_3, J_4, and J_5, and $200N$ for J_6 and J_7.

Muscle elongation is computed by linearly mapping the angular position of the DOF, on which the muscle acts, into the muscle length range. For instance, in the case of

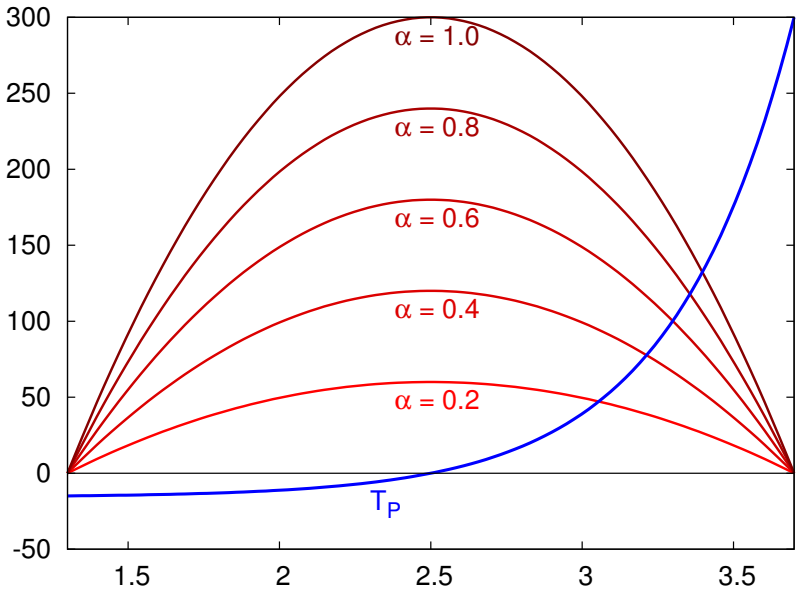


Figure 2.2: An example of the force exerted by a muscle; the graph shows how the force exerted by a muscle varies as a function of the activity of the corresponding motor neuron and of the elongation of the muscle for a joint in which T_{max} is set to $300N$.

the elbow where the limits are $[-170^o, +0^o]$, this range is mapped onto $[+1.3, +3.7]$ for the agonist muscle and $[+3.7, +1.3]$ for the antagonist muscle. Hence, when elbow is completely extended (angle 0), the agonist muscle is completely elongated (3.7) and the antagonist muscle is completely compressed (1.3), vice versa when the elbow is flexed.

The torque applied to an arm joint is the difference between the torques applied by the two antagonist muscles of that joint. The resulting simulated arm movement is then computed by the physics engine, taking into account both torque produced by muscles and forces generated by the interaction with the external environment.

2.3.2 The Hand

The hand is attached to the robotic arm just after the wrist (at joint J_7 as shown in figure 2.1). One of the most important features of the hand is its compliance. In details, the compliance has been obtained setting a maximum threshold of $300N$ to the force exerted by each joint. When an external force acting on a joint exceed this threshold

either the joint cannot move further, or the joint moves backward due to the external force.

The robotic hand is composed of a palm and 15 phalanges that make up the digits (three for each finger) connected through 20 DOFs, $J_8, \ldots, J_{27}$ (see figure 2.1).

Joint J_8 allows the opposition of the thumb with the other fingers and it varies within the range $[-120°, +0°]$, where the lower limit corresponds to thumb-pinky opposition. The knuckle joints J_{12}, J_{16}, J_{20} and J_{24} allow the *abduction/adduction* of the corresponding finger and their ranges are $[0°, +15°]$ for the index, $[-2°, +2°]$ for the middle, $[-10°, +0°]$ for the ring, and $[-15°, +0°]$ for the pinky. All others joints are for the *extension/flexion* of phalanges and vary within $[-90°, +0°]$ where the lower limit corresponds to complete flexion of the phalanx (i.e., the finger closed).

The joints are not controllable independently of each other, but they are grouped. The same grouping principle used for developing the iCub hand [100] has been used. More precisely, the two distal phalanges of the thumb move together as do the two distal phalanges of the index and the middle fingers. Also, all *extension/flexion* joints of the ring and pinky fingers are linked as are all the joints of *abduction/adduction* of the fingers. Hence, only 9 actuators move all the joints of the hand, one actuator for each of the following group of joints: $\langle J_8 \rangle$, $\langle J_9 \rangle$, $\langle J_{10}, J_{11} \rangle$, $\langle J_{13} \rangle$, $\langle J_{14}, J_{15} \rangle$, $\langle J_{17} \rangle$, $\langle J_{18}, J_{19} \rangle$, $\langle J_{12}, J_{16}, J_{20}, J_{24} \rangle$ and $\langle J_{21}, J_{22}, J_{23}, J_{25}, J_{26}, J_{27} \rangle$. These actuators are simple motors controlled by position.

2.3.3 The Neural Controller

The architecture of the neural controllers varies slightly depending on the ecological conditions in which the robot develops its skills. In the case of the development supported by labels provided by a caretaker, the robot is controlled by a neural network which includes 29 sensory neurons, 12 internal neurons with recurrent connections and 23 motor neurons. In the case without the support of labels, the neural network lacks the corresponding sensory neurons. Thus, it is composed of 26 sensory neurons instead of 29. The sensory neurons are divided into four blocks.

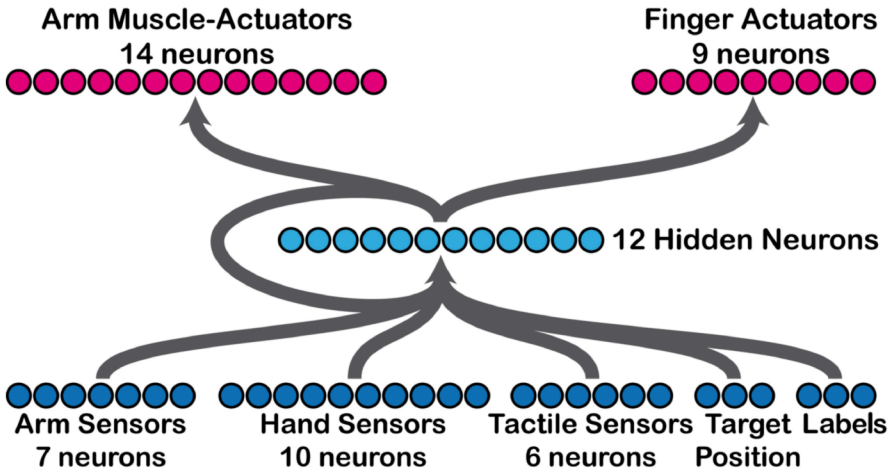


Figure 2.3: The architecture of the neural controllers. The arrows indicate blocks of fully connected neurons

The **Arm Sensors** encode the current angles of the 7 DOFs located on the arm and on the wrist normalised in the range $[0,1]$.

The **Hand Sensors** encode the current angles of hand's joints. However, instead of feeding the network with all joint angles of the hand, the following values are used:

$$\left\langle a(J_8), a(J_9), \frac{a(J_{10})+a(J_{11})}{2}, a(J_{13}), \frac{a(J_{14})+a(J_{15})}{2}, a(J_{17}), \right.$$
$$\left. \frac{a(J_{18})+a(J_{19})}{2}, a(J_{21}), \frac{a(J_{22})+a(J_{23})}{2}, a(J_{12}) \right\rangle$$

where $a(J_i)$ is the angle of the joint J_i normalised in the range $[0,1]$ with 0 meaning fully extended and 1 fully flexed. This way of representing the hand posture mirrors the way in which the hand joints are actuated.

The **Tactile Sensors** encode how many contacts occur on the hand components. The first tactile neuron corresponds to the palm and its activation is set to the number of contacts normalised in the range $[0,1]$ between the palm and another body (i.e., an object or other parts of the hand). Normalisation is performed using a ramp function that saturates to 1 when there are more than 20 contacts. The other five tactile neurons correspond to the fingers and are activated in the same way.

The **Target Position Sensors** can be seen as the output of a vision system (which has

not been simulated) that computes the relative distance in cm of the object with respect to the hand over three orthogonal axes. These values are fed into the networks as they are without any normalisation.

The **Label Sensors** is a block of three neurons each of which represents one of the commands *reach*, *grasp* and *lift*. Specifically, the vector $\langle 50,0,0\rangle$ corresponds to the instruction "*reach the object*", $\langle 0,50,0\rangle$ corresponds to the instruction "*grasp the object*" and $\langle 0,0,50\rangle$ corresponds to the instruction "*lift the object*". The way in which the state of these sensors is set is determined by equation 2.4 explained below.

Note that the state of the **Label** and **Target Position Sensors** varies on a larger interval than the other sensors in order to increase the relative impact of these neurons. Indeed, control experiments in which all sensory neurons were normalized within the $[0,1]$ interval led to significantly lower performance (result not shown).

The outputs $H_i(t)$ of the **Hidden Neurons** are calculated on the basis of following equation:

$$\begin{aligned} y_i(t) &= \sigma\left(\textstyle\sum_{j=1}^{29} w_{ji} I_j(t) + \beta_i\right) \\ H_i(t) &= \delta_i \cdot y_i(t) + (1-\delta_i)\cdot y_i(t-1) \end{aligned} \tag{2.2}$$

where $I_j(t)$ is the output of the j^{th} sensory neuron, w_{ji} is the synaptic weight from the j^{th} sensory neuron to the i^{th} hidden neuron, β_i is the bias of the i^{th} hidden neuron, δ_i is the decay-factor of the i^{th} hidden neuron, and $\sigma(x)$ is the logistic function with a slope of 0.2.

The output neurons are divided into two blocks, the **Arm Muscle Actuators** and the **Finger Actuators**. All outputs of these neurons are calculated in the same way using the following equation:

$$O_i(t) = \sigma\left(\sum_{j=1}^{12} w_{ji} H_j(t)\right) \tag{2.3}$$

where $H_j(t)$ is the output of hidden neuron j as described in 2.2, w_{ji} is the synaptic

weight from the j^{th} hidden neuron to the i^{th} output neuron and $\sigma(x)$ is the logistic function with slope 0.2. With respect to the hidden neurons, the output neurons do not have any bias or decay-factor.

The **Arm Muscle Actuators** output sets the parameter α used in equation 2.1 to update the force applied by muscles of the arm, while the **Finger Actuators** output sets the desired *extension/flexion* position of the nine hand actuators. The state of the sensors, the desired state of the actuators, and the internal neurons are updated every $10ms$.

This particular type of neural network architecture has been chosen to minimize the number of assumptions and to reduce, as much as possible, the number of free parameters. Also, this particular sensory system has been chosen in order to study situations in which the visual and tactile sensory channels need to be integrated.

2.3.4 The Adaptive Process

The free parameters of the neural controller (i.e., the connection weights, the biases of internal neurons and the time constant of leaky-integrator neurons) are set using an evolutionary algorithm [74, 138].

The initial population consists of 100 randomly generated genotypes, which encode the free parameters of 100 corresponding neural controllers. In the conditions in which **Label Sensors** are employed (hereafter, referred to as Exp. A), the neural controller has 792 free parameters. In the other condition without the **Label Sensors** (hereafter, referred to as Exp. B) there are 756 free parameters. Each parameter is encoded into a binary string (i.e., a gene) of 16 bits. In total, a genotype is composed of $792 \cdot 16 = 12672$ bits in Exp. A and $756 \cdot 16 = 12096$ bits in Exp. B. In both experiments, each gene encodes a real value in the range $[-6, +6]$, but for genes encoding the decay-factors δ_i the encoded value is mapped in the range $[0, 1]$.

The 20 best genotypes of each generation are allowed to reproduce by generating five copies each. Four out of five copies are subject to mutations and one copy is

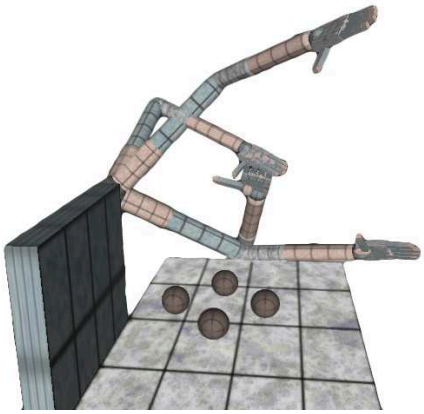


Figure 2.4: Initial positions of the arm and the sphere over imposed; the joints $J_1, \ldots J_4$ are initialised to $\langle -73, -30, -40, -56 \rangle$, $\langle -73, -30, -40, -113 \rangle$, $\langle -6, +30, -10, -56 \rangle$ and $\langle -73, -30, +45, -113 \rangle$; the initial sphere positions are $\langle -18, +10 \rangle$, $\langle -26, +18 \rangle$, $\langle -18, +26 \rangle$ and $\langle -10, +18 \rangle$.

not mutated. During mutation, each bit of the genotype has a 1.5% probability to be replaced with a new randomly selected value. The evolutionary process is repeated for 1000 generations.

The robots are rewarded for reaching, grasping and lifting a spherical object of radius $2.5\,cm$ placed on the table in exactly the same way in both Exp. A and Exp. B. Each agent of the population is tested 4 times. Each time the initial position of the arm and the sphere change. Figure 2.4 shows the four initial position of the arm and of the sphere superimposed on one another. For each initial arm/object configuration a random displacement of $\pm 1^o$ is added to each joint of the arm and a random displacement of $\pm 1.5\,cm$ is added on the x and the y coordinates of the sphere position. Each trial lasts $6\,sec$ corresponding to 600 simulation steps. The sphere can move freely and it can eventually fall off the table. In this case, the trial is stopped prematurely.

The fitness function is made up of three components: FR for reaching, FG for grasping and FL for lifting the object. Each trial is divided in 3 phases in which only a single fitness component is updated. The conditions that define the current phase at each

timestep and consequently which component has to be updated are the following:

$$\begin{aligned} r(t) &= 1 - e^{(-0.1 \cdot ds(t))} \\ g(t) &= e^{(-0.2 \cdot graspQ(t))} \\ l(t) &= 1 - e^{(-0.3 \cdot contacts(t))} \\ Phase(t) &= \begin{cases} reach & r(t) > g(t) \vee g(t) < 0.5 \\ grasp & otherwise \\ lift & g(t) > 0.7 \wedge l(t) > 0.6 \end{cases} \end{aligned} \tag{2.4}$$

where $ds(t)$ is the distance from the centre of the palm to a point located $5\,cm$ above the centre of the sphere. $graspQ(t)$ is the distance between the centroid of the fingertips-palm polygon and the centre of the sphere. $contacts(t)$ is the number of contacts between the fingers and the sphere. The shift between the three phases is irreversible (i.e. the reach phase is always followed by the reach or grasp phases and the grasp phase is always followed by the grasp or lift phases).

Essentially, the current phase is determined by the values $r(t)$, $g(t)$ and $l(t)$. When $r(t)$ is high (i.e., when the hand is far from the object) the robot should reach the object. When $r(t)$ decreases and $g(t)$ increases (i.e., when the hand approaches the object from above) the robot should grasp the object. Finally, when $l(t)$ increases (i.e., when the number of activated contact sensors are large enough) the robot should lift the object. The rules and the thresholds included in equation 2.4 have been set manually on the basis of our intuition and have not been adjusted through an automated trial and error process. In Exp. A, the phases are used to define which label the robot perceives.

The three fitness components are calculated in the following way:

$$FR = \sum_{t \in T_{Reach}} \left(\frac{0.5}{1 + \frac{ds(t)}{4}} + \frac{0.25}{1 + ds(t)} \left(fOpen(t) + pRot(t) \right) \right) \tag{2.5}$$

$$FG = \sum_{t \in T_{Wrap}} \left(\frac{0.4}{1 + graspQ(t)} + \frac{0.2}{1 + \frac{contacts(t)}{4}} \right) \tag{2.6}$$

$$FL = \sum_{t \in T_{Lift}} oLifted(t) \tag{2.7}$$

where T_{Reach}, T_{Wrap} and T_{Lift} are the time ranges determined by equation 2.4. $fOpen(t)$ correspond to the average degree of extension of the fingers, where 1 occurs when all fingers are extended and 0 when all fingers are closed. $pRot(t)$ is the dot product between the normals of the palm and the table, with 1 referring to the condition in which the palm is parallel to the table and 0 to the condition in which the palm is orthogonal to the table. $oLifted(t)$ is 1 only if the sphere is not touching the table and it is in contact with the fingers, otherwise is 0.

The total fitness is calculated at the end of four trials as: $F = min(500, FR) + min(720, FW) + min(1600, FL) + bonus$, where $bonus$ adds 300 for each trial where the agent switches from reach phase to grasp phase only, and 600 for each trial where the agent switches from reach to grasp phase and from grasp to lift phase.

During the reach phase the agent is rewarded for approaching a point located $5\,cm$ above the centre of the object with the palm parallel to the table and the hand open. Note that the rewards for the hand opening and the rotation of the palm are relevant only when the hand is near the object (due to $0.25/(1 + ds(t))$ factor); in this way the agent is free to rotate the palm when the hand is away from the sphere allowing any reaching trajectory.

During the grasp phase, the centroid of the fingertips-palm polygon can reach the centre of the sphere only when the hand wraps the sphere with the fingers, producing a potential power grasp. During the lift phase, the reward is given when the agent effectively moves the sphere upward of the table.

2.4 Results

For both Exp. A (with labels) and Exp. B (without labels), we run ten evolutionary simulations for 1000 generations, each using a different random initialisation. Figures 2.5 and 2.6 show the fitness curves of all runs of Exp. A and Exp. B respectively. Looking at the fitness curves of the best agents at each generation of each evolutionary run, we noticed that, for Exp. A, there are three distinctive evolutionary paths (see figure 2.7a). The most promising is run 7, in which the last generations agents have the highest fitness. The curve corresponding to run 2 is representative of a group of seven evolutionary paths which, after a short phase of fitness growth, reach a plateau at $F = 2000$. The curve corresponding to run 9 is representative of a group of two evolutionary paths which are characterised by a long plateau slightly above $F = 1000$. Generally speaking, these curves progressively increase by going through short evolutionary intervals in which the fitness grows quite rapidly followed by a long plateau[2]. For Exp. B, all the runs show a very similar trend, reaching and constantly remaining on a plateau at about $F = 3000$ (see figure 2.7b).

Due to the nature of the task and of the fitness function, it is quite hard to infer from these fitness curves what could be the behaviour of the agents during each evolutionary phase. However, based on what we know about the task, and by visual inspection of the behaviour exhibited by the agents, we found out how the agents behave at different generations of each evolutionary run. In Exp. A, the phases of rapid fitness growth are determined by the *bonus* factor, which substantially rewards those agents that successfully accomplish single parts of the task. The first fitness jump is due to the *bonus* factor associated to the execution of a successful reaching behaviour. This jump corresponds to the phase of fitness growth observed in run 7 in correspondence of label R figure 2.7a, and in run 2 in correspondence of label V figure 2.7a. The agents generated after these fitness jumps are able to systematically reach the object. Run 9 does not go through the first fitness jump, and the agents of this run lack the ability to

[2]The fitness curves of the runs not shown are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

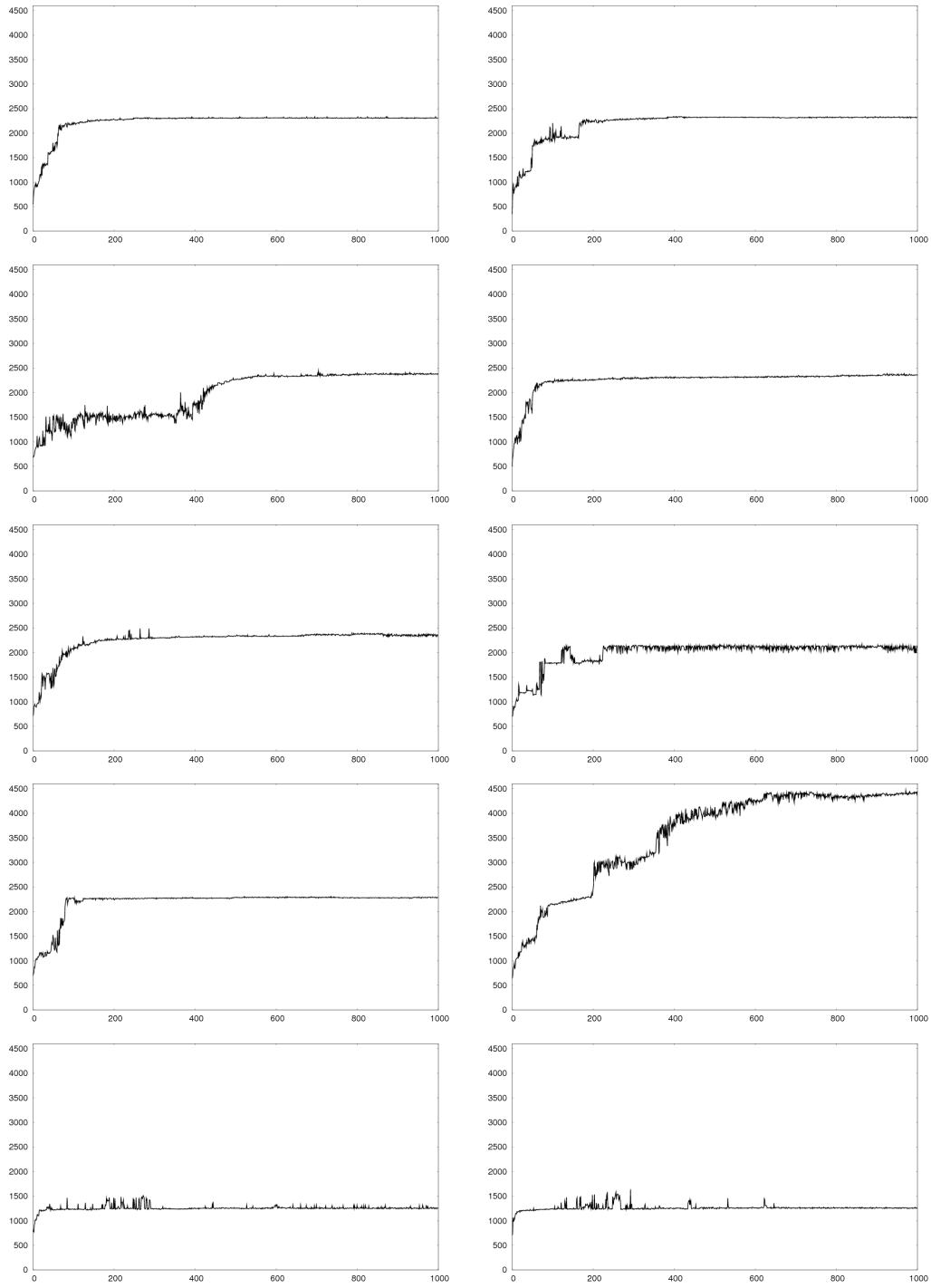


Figure 2.5: Fitness of all runs of Exp. A. The first row shows run 0 and run 1, the second row run 2 and run 3 and so on.

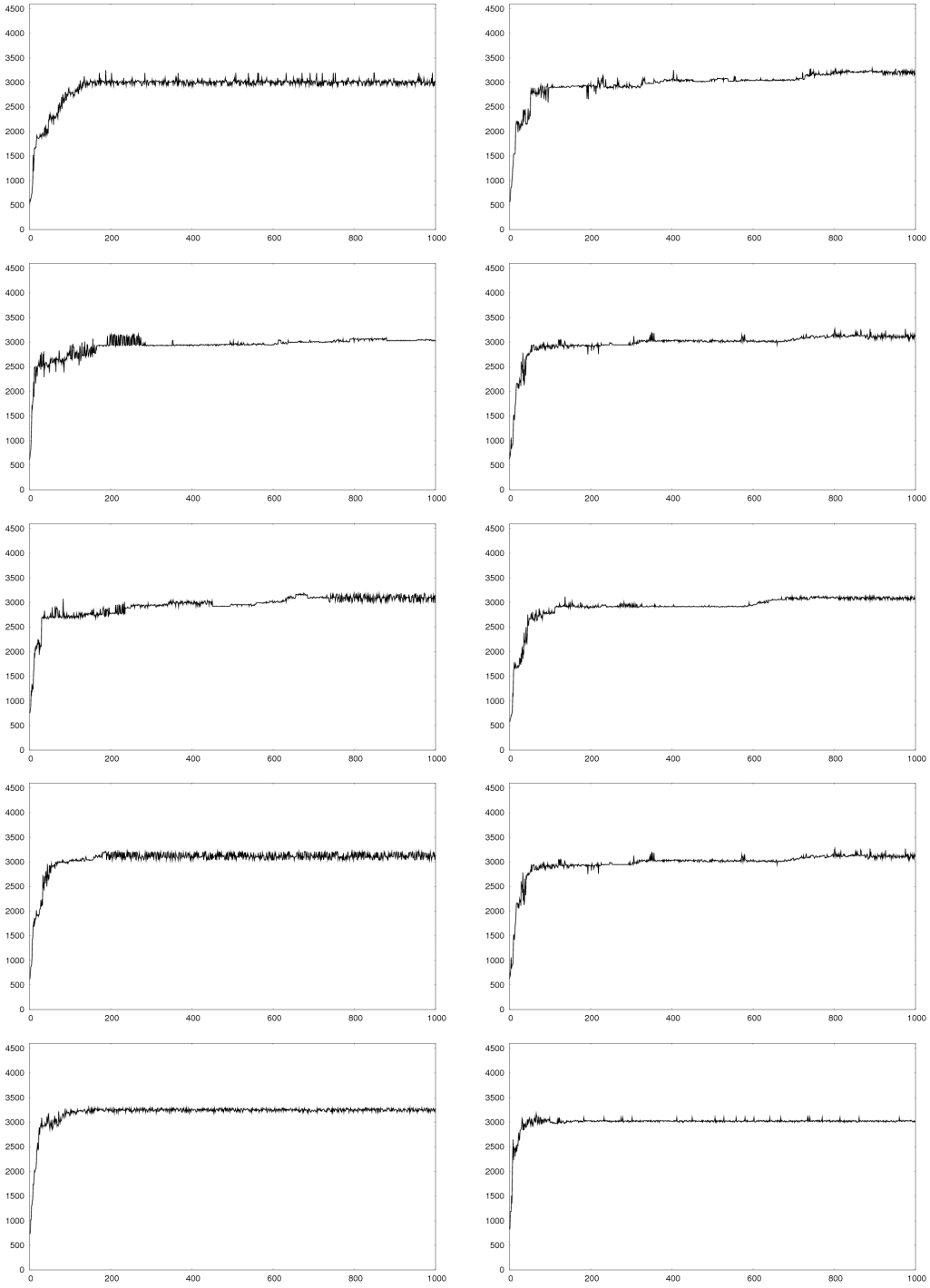


Figure 2.6: Fitness of all runs of Exp. B. The first row shows run 0 and run 1, the second row run 2 and run 3 and so on.

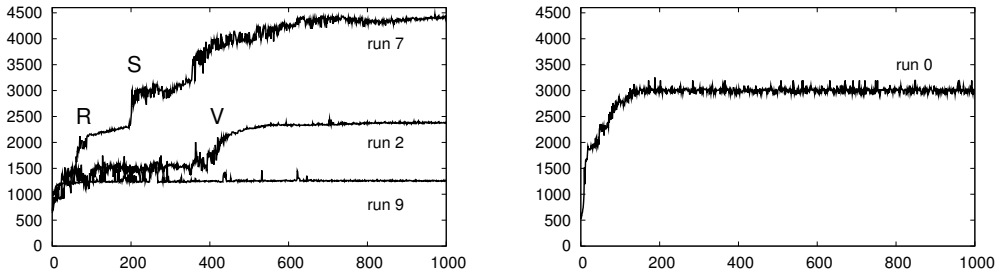


Figure 2.7: Fitness of the best agents at each generation of (a) run 2, run 7, and run 9 of Exp. A, and (b) run 0 of Exp. B.

systematically carry out a successful reaching behaviour.

The second fitness jump is due to the *bonus* factor associated with the execution of a successful grasping behaviour. Only in run 7 is it possible to observe a phase of rapid fitness growth corresponding to a second fitness jump (see label S figure 2.7a). The agents generated after this jump are able to successfully carry out reaching and grasping. Note also that, in run 7, the fitness curve keeps on growing until the end of the evolution. This growth is determined by the evolution of the capability to lift the object. Thus, in run 7, the best agents following generation 400 are capable of reaching, grasping, and lifting the object. The constant increment of the fitness is determined by the fact that the agents become progressively more effective in lifting the object. Run 2 does not go through a second fitness jump. The agents of this run lack the ability to systematically carry out a successfully grasping behaviour.

In summary, only run 7 has generated agents (i.e., those best agents generated after generation 400) capable of successfully accomplishing reaching, grasping, and lifting.[3] The best agents of run 2, and of the other six runs that show a similar evolutionary trend, are able to systematically reach but not grasp the object and completely lack the ability to lift the object. The best agents of run 9, and of the other run that show a similar evolutionary trend, are not even able to systematically reach the object. In Exp. B, they are able to successfully reach and grasp the object, but not lift it.

[3]Movies of the behaviour and corresponding trajectories are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

2.4.1 Robustness & Generalisation

In this section, we show the result of a series of post-evaluation tests aimed at establishing the effectiveness and robustness of best agents' behavioural strategies of the four runs show in figure 2.7. In these tests, the agents, from generation 900 to generation 1000 of each run, are subjected to a series of trials in which the position of the object as well as the initial position of the arm are systematically varied. For the position of the object, we define a rectangular area ($28\,cm \times 21\,cm$) divided in 11×11 cells. The agents are evaluated for reaching, grasping and lifting the object positioned in the centre of each cell of the rectangular area. For the initial position of the arm, we use the four initial positions employed during evolution as prototypical cases (see figure 2.4). For each prototypical case, we generate 100 slightly different initial positions with the addition of a $\pm 10^{\circ}$ random displacement on joints J_1, J_2, J_3, and J_4. Thus, this test is comprised of 48400 trials, given by 400 initial positions ($4 \cdot 100$) for each cell, repeated for 121 cells corresponding to the different initial positions of the object during the test. In each trial, reaching is considered successful if an agent meets the conditions to switch from the *reach* phase to the *grasp* phase (see equation 2.4). Grasping is considered successful if an agent meets the conditions to switch from the *grasp* phase to the *lift* phase (see equation 2.4). Lifting is considered successful if an agent manages to keep the object at more than $1\,cm$ from the table until the end of the trial. In this section, we show the results of a single agent for each run. However, agents belonging to the same run obtained very similar performances. Thus, the reader should consider the results of each agent as representative of all the other agents of the same evolutionary run.

All the graphs in figure 2.8, show the relative position of the rectangular area and the cells with respect to the agent/table system. Moreover, each cell of this area is coloured in shade of grey, with black indicating 0% success rate, and white indicating 100% success rate. As expected from the previous section, the agent chosen from run 7 Exp. A proved to be the only one capable of successfully accomplishing all the three phases of

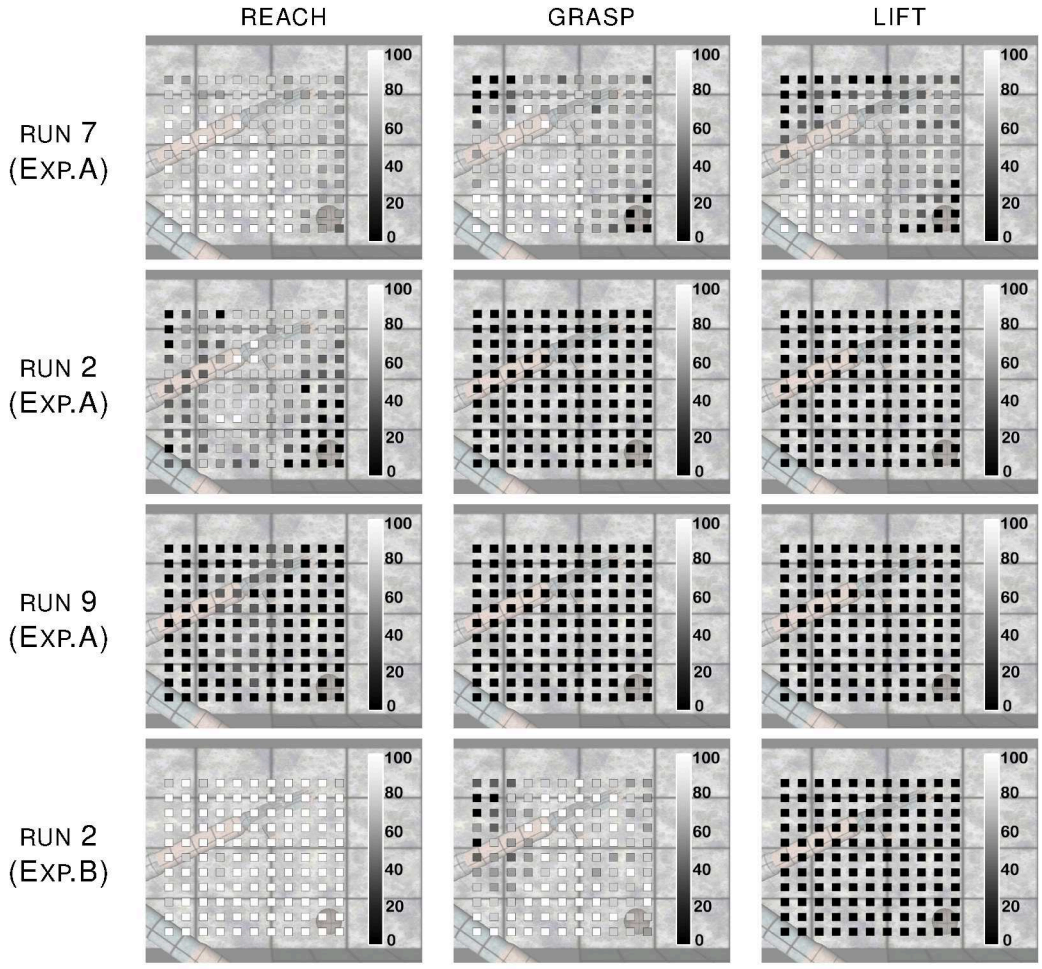


Figure 2.8: Results of post-evaluation tests on the robustness of reaching, grasping and lifting behaviour of the best agent at generation 1000 of run 7, run 2, and run 9 in Exp. A and run 0 in Exp. B. The cells in shades of grey indicate the percentage of successful trials (from 0% success rate in black, to 100% success rate in white), with the object located in the centre of each cell.

the task. This agent proved capable of successfully reaching the object placed almost anywhere within the rectangular area. Its grasping and lifting behaviour are less robust than the reaching behaviour. Indeed, the grasping and lifting performances are quite good everywhere except in two small zones located in the top left and bottom right of the rectangular area in which cells are coloured black. The agent chosen from run 2 Exp. A proved to be capable of successfully performing reaching behaviour for a broad range of object initial positions, and completely unable to perform grasping and lifting behaviour. The agent chosen from run 9 Exp. A does not even manage to systematically bring the hand close to the object regardless of the object's initial position. The agent chosen from run 0 Exp. B, proved capable of successfully performing reaching and grasping behaviour but not lifting behaviour.

2.5 Conclusion

We showed how a simulated humanoid robot controlled by an artificial neural network can acquire the ability to manipulate spherical objects located over a table by reaching, grasping and lifting them. The robot is trained through an adaptive process in which the free parameters encode the control rules that regulate the fine-grained interaction between the agent and the environment, and the variations of these free parameters are retained or discarded on the basis of their effects at the level of the behaviour exhibited by the agent. This means that the agents develop their skills autonomously in interaction with the environment. Moreover, this means that the agents are left free to determine the way in which they solve the task within the limits imposed by i) their body/control architecture, ii) the characteristics of the environment, and iii) the constraints imposed by the utility function that rewards the agents for their ability to reach an area located above the object, wrap the fingers around the object, and lift the object. The analysis of the best individuals generated by the adaptive process shows that the agents of a single evolutionary run manage to reach, grasp, and lift the object in a reliable and effective way. Moreover, when tested in new conditions with respect to those experienced during the adaptive process, these agents proved to be capable of

generalising their skills with respect to new object positions never experienced before.

By comparing the results of the experiments A and B in which the robots received or not received action labels in input from the caretaker we observed that only in the first case the robots were able to fully solve their task. This result confirms the hypothesis that the availability of labels, a primitive form of linguistic instructions, that encode the type of behaviour that should be exhibited in a given phase, promotes the development of the corresponding behavioural skills and/or the development of the appropriate behavioural sequence. More specifically, the fact that the best agents of Exp. B succeed in exhibiting the reaching and then the grasping behaviour but not the lifting behaviour suggests that labels represent a crucial pre-requisite in situations in which the agent has to develop an ability to produce different behaviours in similar sensory-motor circumstances. In fact, the transitions from reaching to grasping behaviours are marked by well-differentiated sensory-motor states. This enables the adapting robots to develop the two required differentiated behaviours and to switch from the former to the latter behaviour effectively, even without the support of labels provided by a caretaker. Instead, the transition from the grasping to the lifting behaviour is not characterised by well-differentiated sensory-motor states. Consequently, in this case the availability of an additional external cue (the labels) that support the development of differentiated behaviour and facilitate the transition from the former to the latter behaviour constitutes a necessary pre-requisite.

Overall these results extend the results reported in previous studies reviewed in Section 2.1. In particular these results demonstrate how the exposure to language-like inputs facilitates not only the development of categorization skills but also the development of behavioural skills. Moreover these results demonstrate how even a simple for of language can play a key role for the development of complex skills, such as object manipulation.

The method presented in this chapter has been later extended in the work described in [58]. In this work the authors demonstrated how, in agreement with Vygotsky's hypo-

thesis, the possibility to utilize an inner language in which the robot talk to itself promote the possibility to display complex behavioural capabilities.

2.6 The experiment and FARSA development

The experiment in the present chapter was realized using a rather complex software tool. It constitutes a prototypical example of an embodied cognition experimental setup, in which the interaction between the agent body and the environment plays a fundamental role for the acquisition of the required skills. The experiment also required the use of a computer simulation: apart from the substantial increase in the time required to run each replica, a real robot would have been easily damaged by the almost completely random movements during the initial phases of development.

To software needed to run the experiment had to integrate different components:

- a physics simulator;
- a library to build and run different kinds of neural networks;
- a library to build and run different kinds of genetic algorithms.

In section 4.3 we will review the robotic simulators that are currently available and will explain why we decided to write our own. For this experiment we partially reused pieces of software developed for previous studies. For the physics simulation we wrote an higher level library in C++, wrapping a low-level C library, which would then become *worldsim* in FARSA (details will be given in section 4.4.1). For neural networks and genetic algorithm we expanded software developed at the LARAL laboratory and connected it to the physics library.

The software developed for this experiment can be considered the first, very preliminary version of FARSA. It contained the fundamental pieces needed for an evolutionary robotics experiment, in a singular integrated tool. Yet, it still lacked most of what is needed to easily setup an experiment. In particular it was not modular and the slightest change to the experimental setup required changing to the source code, slowing down

the research process. Moreover the lack of modularity also made it difficult to share code with other people in the laboratory.

Another fundamental problem of the simulator used to run this experiment was that it was rather difficult to replicate the experiments. Replicability is one of the founding principles of the scientific method, as it ensures that the results of an experiment can be verified by other researchers. While it is indeed possible to replicate the experiments described in this chapter, it can be problematic to run them for people that did not participate in the development of the simulator. There are two main reasons: the compilation process was not documented at all and the user interface was not designed to be usable by other researchers, containing only the bare minimum needed to perform the experiment.

In chapter 4 we will further discuss the importance of replicability and of the possibility to share code. We will then show how the FARSA simulator aims at resolving these issues.

Chapter 3

Behaviour Generalisation and the Emergence of Linguistic Compositionality in Evolving Robots

This chapter presents a robotic model designed to look at aspects related to the emergence of compositional semantic structures in simulated agents [125]. The obtained results demonstrate how the agents, trained to execute several actions by responding to linguistic instructions, can generalise their linguistic and behavioural skills to never experienced instructions through the production of appropriate behaviours. The analysis of the best agents and the comparison of different experimental conditions, in which the representation of the linguistic instructions is the same but the set of actions the agents has to perform is varied, demonstrates how the emergence of compositional semantics is affected by the presence of behavioural regularities in the execution of different actions. Post-evaluation tests also unveil further details of the behavioural and linguistic strategies used by agents equipped with compositional semantics to accomplish the task.

The chapter is structured as follows. First the most relevant work in the literature is reviewed and in particular those described in [2, 117, 118], which have been particularly inspiring for the present work. Then the experimental problem, the characteristics of the agents, the architecture of the agent's control system, and the adaptive process are described. Then the obtained results are illustrated, along with the analysis that were performed in order to understand the mechanisms that are at the basis of the

capacity of the agents to comprehend new sentences and to produce new behaviours. Afterwards the relation between the results obtained and the empirical results collected in child language studies are discussed. Finally, the main implications of this work are examined, both the scientific ones and the insights into the development of the FARSA simulator.

3.1 Background

By the term "compositional semantics", we refer to a functional dependence of the meaning of an expression on the meaning of its parts. Compositional semantics in natural language refers to the human ability to understand the meaning of spoken or written sentences from the meaning of their parts, and the way in which these parts are put together. For example, the meaning of an unknown sentence like "Susan likes tulips" can be understood by learning the following three sentences: "Julie likes daisies", "Julie likes tulips", and "Susan likes daisies". In this example, the meaning of the original sentence is achieved through compositional semantics by generalising the meaning of single words from a known (already learnt) to an unknown (yet to be learnt) context.

During the cognitivist era, compositionality was supposed to be underpinned by concatenative processes in which the tokens of an expression's constituents (and the sequential relations among them) are preserved in the expression itself [37]. The difficulties shown by classic symbolic AI in accounting for general associations between semantic representations and sensory-motor profiles, and in particular in accounting for the acquisition of linguistic semantics through behavioural experiences, determined a paradigm shift in which an alternative perspective on compositionality emerged [49]. In the last decade of the previous century, the connectionist approach to cognition proposed the idea of functional compositionality; that is compositional semantics systems in which the tokens of an expression's constituents (and the sequential relations among them) are not preserved in the expression itself [127]. Various connectionist models proved that artificial neural networks can be employed to physically instantiate

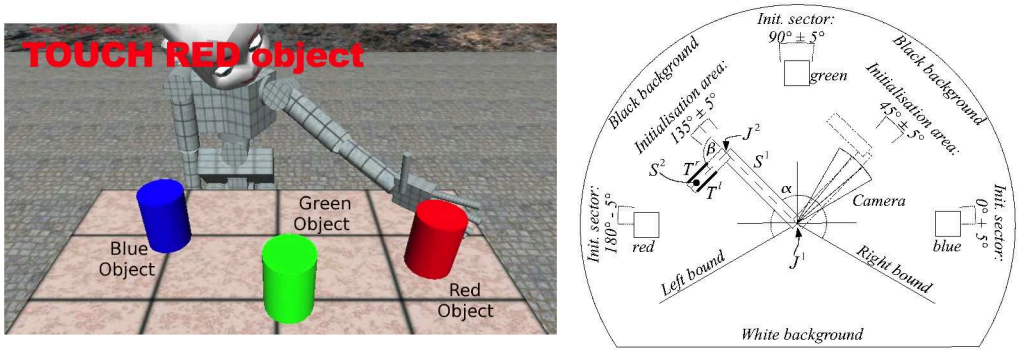
functional compositional semantic structures [32].

More recently, autonomous (real or simulated) robots have been used to investigate how a form of language can emerge and evolve in a population of robots interacting between themselves and with the physical environment [16, 75, 111, 122]. Moreover, several studies have investigated how a robot can acquire a language by interacting with a human user. For example, in [88], the authors designed robotic experiments with robots that, in addition to react to linguistic commands issued by the user are also able to acquire both the meaning of new linguistic instructions and new behavioural skills on the fly, by grounding the new commands in pre-existing motor skills. In [89] the authors designed robots able to cooperate and to share attention with a human user in a restricted experimental setting. This is achieved by allowing the robot to observe the goal-directed behaviour exhibited by the user and to adopt her plan. In [136], the author designed a developmental learning architecture that allows a robot to progressively expand its behavioural repertoire while interacting with a human trainer that shapes its behaviour. In [17], the authors studied how new, higher-order behavioural abilities can be autonomously built upon previously-grounded basic action categories, acquired through language-mediated interactions with human users.

In [2, 117, 118], the authors investigate the issue of grounding compositional semantic structures in an agent's sensory-motor skills in tasks that require the shift from rote knowledge to systematised knowledge. In particular, in [2, 117] a robot learns to execute actions in response to linguistic instructions consisting of two-words sentences. The robots neural controller comprises a behavioural and a linguistic module. The behavioural module is trained through a learning-by-demonstration method in which the sensory-motor states experienced while the robot is moved by the experimenter, through tele-operation or kinesthetic teaching, are used as a training set. The linguistic module is trained to predict the next word of a two-word linguistic instructions in which the words are provided to the agent sequentially. In [2] both the behavioural and the linguistic module are trained only on a subset of all possible linguistic instructions res-

ulting from the combination of all possible objects with all possible actions. In [117], the linguistic module is trained only on a subset of all possible linguistic instructions whereas the behavioural module is trained to execute all the possible instructions. In all three studies [2, 117, 118], the agent proves capable of performing actions associated with linguistic instructions not experienced during training. The authors claim that behavioural and/or linguistic generalisation is achieved by "conceiving something not experienced as a recombination of learnt examples" [117]. The contribution of these works is in bringing evidence for a dynamical perspective on compositional semantic systems, alternative to the one in which neural correlates of language are viewed as atomic elements semantically associated to basic units of the linguistics system. The authors show that compositional systems can be underpinned by neural structures in which the neural correlates of the linguistic instructions are dynamically self-organised topological properties of the neural substrate, induced by similarities among sensory-motor sequences. Each instruction (i.e., action plus object) is represented in a two-dimensional semantic space by a single point which lies in a grid-like geometrical structure in which one dimension refers to actions and the other to objects. The geometrical arrangement of neural correlates that emerged during the simultaneous training of the behavioural and linguistic modules, allows the agent to successfully respond to non-experienced linguistic instructions.

In this chapter, we describe a series of simulations in which a robot is required to perform a task very similar to the one described in [117]. As in [117], our goal is also to investigate the emergence and the underlying properties of a functionally compositional semantic system in a task that requires the shift from rote knowledge to systematised knowledge. However, we look at the problem with different methods that, as we will see, lead to a qualitatively different type of solution. In our case, a neural controller is trained to execute a subset of possible linguistic instructions through an evolutionary method in which the robot is rewarded for the ability to achieve a certain goal without specifying the sequence of movements through which this goal should be realised. As shown in Section 3.6, this allows the robot to co-develop linguistic skills to access the



(a) (b)

Figure 3.1: (a) An image of the simulated iCub and its word. (b) A schematic representation of the agent structure and its world in the 2D simulator. The vision system of the agent is drawn only with respect to the arm initialised on the right initialisation area. α refers to the angular position of S^1. β refers to the angular position of S^2 with respect to S^1. See text for further details.

meaning of the instructions and behavioural skills to execute them.

3.2 The agent structure and the task

The experimental scenario consists of a humanoid iCub robot [99] placed in front of a table with a red, green, and blue object as shown in Figure 3.1a. The robot is trained to execute seven actions on the object by responding to linguistic instructions formed by all the possible combinations of the three action words “INDICATE”, “TOUCH”, and “MOVE” and the three object word “Red”, “Green”, and “Blue” with the exception of the sentences “TOUCH Green object” and “MOVE Blue object”. After training, the robot is then tested on the two *non-experienced* sentences to assess whether it produces the appropriate corresponding behaviours even though it had neither experienced these sentences before nor received training on the two corresponding behaviours. To reduce the computational costs associated with the simulation of such a complex robot, we carried out our experiments on a simpler experimental 2D scenario involving a two-segments arm described below. We then port the obtained results to a simulated iCub by controlling the robot’s hand position on the basis of the current position of the end-effector of the simplified arm through the inverse kinematic software described in [86].

The best evolved controllers have been successfully ported on the iCub simulator[1].

In the simple two-dimensional simulated world, an agent is composed of an arm with two segments referred to as S^1 (100 cm) and S^2 (50 cm), and two degrees of freedom (DOF). Each DOF comprises a rotational joint that acts as the fulcrum and an actuator. The first actuator causes S^1 to rotate clockwise or anticlockwise around joint J^1, with the movement restricted in the right ($-30°$) and the left ($210°$) bound. The other actuator causes S^2 to rotate clockwise or anticlockwise around joint J^2 within the range [$90°$, $0°$] with respect to S^1 (see Figure 3.1b). Friction and momentum are not considered.

In the environment there are three objects of different colours (i.e., a blue, a green, and a red object). The objects are placed 150 cm from J^1 with their centre placed anywhere on the chord delimiting their corresponding Init. sector (see Figure 3.1b). The objects do not move unless pushed by the arm. The agent is equipped with a linear camera with a receptive field of $30°$, divided in three sectors, each of which has three binary sensors (C_i^B for blue, C_i^G for green, and C_i^R for red, with $i \in [1,2,3]$ sectors). Each sensor returns 1 if the blue/green/red object falls within the corresponding sector. If no coloured object is detected, the readings of the sensors are set to 0 (i.e., the camera perceives a black background). The camera and S^1 move together. The experimental set up is built in a way that at each time step there can be only one object in the camera view.

The agent has means to perceive whenever S^1 reaches the right or the left bound through the activation of the camera sensors. That is, when S^1 reaches the right bound C_1^B, C_1^G, and C_1^R are set to 1 (i.e., the first camera sector perceives a white background). When S^1 reaches the right bound C_3^B, C_3^G, and C_3^R are set to 1 (i.e., the third camera sector perceives a white background). Finally, two binary touch sensors (i.e., T^r, T^l) are placed on the right, and left side of S^2. Collisions between the agent and an object are handled by a simple model in which whenever S^2 pushes the object the relative contact points remain fixed.

[1] Movies and further methodological details concerning the porting can be found at `http://laral.istc.cnr.it/esm/tuci-etal-IEEE_TAMD2010/`.

To assess whether the composition of the behavioural set affects the developmental process and the generalisation capabilities of the agents, we run two sets of evolutionary experiments. In the **With-Indicate** experimental condition, the task consists in the execution of the following instructions: TOUCH Blue object ($Inst^{T}_{blue}$), TOUCH Red object ($Inst^{T}_{red}$), MOVE Green object ($Inst^{M}_{green}$), MOVE Red object ($Inst^{M}_{red}$), INDICATE Blue object ($Inst^{IN}_{blue}$), INDICATE Green object ($Inst^{IN}_{green}$), and INDICATE Red object ($Inst^{IN}_{red}$). In the **With-Ignore** experimental condition, the action INDICATE is substituted with the action IGNORE. Thus, $Inst^{IG}_{blue}$ refers to IGNORE Blue object, $Inst^{IG}_{green}$ refers to IGNORE Green object, and $Inst^{IG}_{red}$ refers to IGNORE Red object. For both evolutionary conditions, the linguistic instructions experienced during training are referred to as *experienced* instructions, while the instructions TOUCH Green object ($Inst^{T}_{green}$) and MOVE Blue object ($Inst^{M}_{blue}$), never experienced during training, are referred to as *non-experienced* instructions (see also Table 3.1). The object-label and the action-label are

Table 3.1: The linguistic instructions. In grey the *non-experienced* instructions, that is, those not experienced during training. The table also shows the notation used in equation 3.1 to refer to each bit of the linguistic instructions.

	MOVE $Inst^{M}_{o}$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	0	1	1
Green	1	0	1	0	1	1
Red	0	1	1	0	1	1

	TOUCH $Inst^{T}_{o}$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	1	0	1
Green	1	0	1	1	0	1
Red	0	1	1	1	0	1

	INDICATE $Inst^{IN}_{o}$ - IGNORE $Inst^{IG}_{o}$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	1	1	0
Green	1	0	1	1	1	0
Red	0	1	1	1	1	0

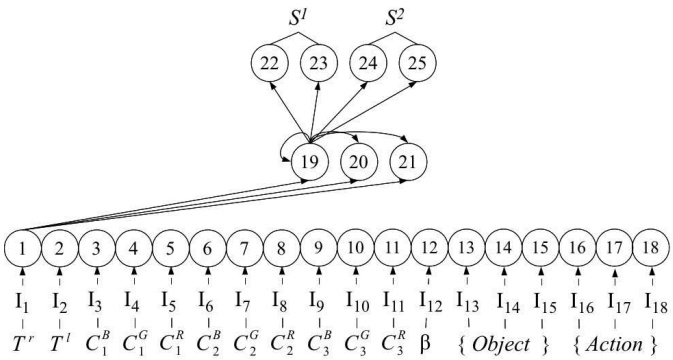


Figure 3.2: The neural network. Continuous line arrows indicate the efferent connections for the first neuron of each layer. Underneath the input layer, it is shown the correspondences between sensors/linguistic instructions, the notation used in equation 3.1 to refer to them, and the sensory neurons.

given to the agent concurrently and for the entire duration of a trial.

TOUCH and MOVE require the agent to rotate S^1 and S^2 until S^2 collides with the target object. TOUCH requires an agent to remain in contact with the target object with the right side of S^2 (that is, by activating the touch sensor T^r) for an uninterrupted interval of 100 time steps. During this interval, S^1 must not rotate. MOVE requires an agent to rotate S^1 more than $35°$ while S^2 is touching the object with its right side. The rotation of S^1 while S^2 is touching the object determines the movement of the object. INDICATE requires an agent to rotate S^1 until the angular distance between S^1 and the object is less than $30°$. INDICATE is correctly executed only if S^1 remains at less than $30°$ from the target object for more than 100 time steps. IGNORE requires the agent to look at anything except the target object. The agent has to move away from positions in which the target object falls within its visual field. During the execution of INDICATE and IGNORE, an agent must not collide with any object. During the execution of TOUCH and MOVE, an agent must not collide with the non-target objects (i.e., the objects not mentioned in the current linguistic instruction).

After training, all the agents are evaluated for their capability to access *experienced* and *non-experienced* linguistic instructions and to execute the corresponding behaviours.

3.3 The agent controller

The agent controller is composed of a continuous time recurrent neural network (CTRNN) of 18 sensor neurons, 3 inter-neurons, and 4 motor neurons [8]. At each time step sensor neurons are activated using an input vector I_i with $i \in [1,..,18]$ corresponding to the sensors readings. In particular, I_1 and I_2 are the readings of touch sensors T^r and T^l, respectively; I_3 to I_{11} are the readings of the camera sensors; I_{12} is refers to the normalised angular position of S^2 with respect to S^1 (i.e., β); I_{13} to I_{18} are the linguistic input and their value depend on the current linguistic instruction. I_{13}, I_{14}, and I_{15} identify the object, I_{16}, I_{17}, and I_{18} identify the action to execute (see Fig. 3.2).

The inter-neuron network is fully connected. Additionally, each inter-neuron receives one incoming synapse from each sensory neuron. Each motor neuron receives one incoming synapse from each inter-neuron. There are no direct connections between sensory and motor neurons. The states of the motor neurons are used to control the movement of S^1 and S^2 as explained later. The states of the neurons are updated using the following equation:

$$\Delta y_i = -y_i + gI_i; \text{ for } i \in \{1,..,18\}; \tag{3.1}$$

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{21} \omega_{ji}\sigma(y_j + \beta_j); \text{ for } i \in \{19,..,21\}; \tag{3.2}$$

$$\Delta y_i = -y_i + \sum_{j=19}^{21} \omega_{ji}\sigma(y_j + \beta_j); \text{ for } i \in \{22,..,25\}; \tag{3.3}$$

with $\sigma(x) = (1+e^{-x})^{-1}$. In these equations, using terms derived from an analogy with real neurons, y_i represents the cell potential, τ_i the decay constant, g is a gain factor, I_i the intensity of the perturbation on sensory neuron i, ω_{ji} the strength of the synaptic connection from neuron j to neuron i, β_j the bias term, $\sigma(y_j + \beta_j)$ the firing rate (hereafter, f_i). All sensory neurons share the same bias (β^I), and the same holds for all motor neurons (β^O). τ_i and β_i with $i \in \{19,..,21\}$, β^I, β^O, all the network connection weights ω_{ij}, and g are genetically specified networks' parameters. At each time step the angular movement of S^1 is $2.9H(f_{22} - 0.5)sgn(0.5 - f_{23})$ degrees and of S^2 is

$2.9H(f_{24}-0.5)sgn(0.5-f_{25})$ degrees, where H is the Heaviside step function and sgn is the sign function. Cell potentials are set to 0 when the network is initialised or reset, and equation 3.2 is integrated using the forward Euler method with an integration time step $\Delta T = 0.1$.

3.4 The evolutionary algorithm

A simple generational genetic algorithm is employed to set the parameters of the networks [43]. At generation 0, a random population of 100 vectors is generated by initialising each component of each vector to a value chosen uniformly random in the range $[0,1]$. Each vector comprises 84 real values (i.e., 75 connection weights ω_{ji}, 3 decay constants τ_i, 5 bias term β and 1 gain factor g shared by all the sensory neurons). Hereafter, using terms derived from an analogy with biological systems, a vector is referred to as genotype and its components as genes.

Generations following the first one are produced by a combination of selection with elitism and mutation. For each new generation, the three highest scoring genotypes ("the elite") from the previous generation are retained unchanged. The remainder of the new population is generated by fitness-proportional selection from the 50 best genotypes of the old population. New genotypes, except "the elite", are produced by applying mutation. Mutation entails that a random Gaussian offset is applied to each gene, with a probability of 0.4. The mean of the Gaussian is 0, and its standard deviation is 0.1. During evolution, all genes are constrained to remain within the range $[0,1]$. That is, if due to mutations a gene falls below zero, its value is fixed to 0; if it rises above 1, its value is fixed to 1.

Genotype parameters are linearly mapped to produce network parameters with the following ranges: $\beta^I \in [-4,-4]$, β^O in$[-5,-5]$, β_i in $[-5,-5]$ with $i \in \{19,..,21\}$, $\omega_{ij} \in [-8,8]$, with $i \in \{1,..,18\}$, and $j \in \{19,..,21\}$, $\omega_{ij} \in [-10,10]$, with $i \in \{19,..,21\}$, and $j \in \{19,..,25\}$, gain factor $g \in [1,13]$. Decay constants τ_i with $i \in \{19,..,21\}$, are firstly linearly mapped into the range $[-1.0,2.0]$ and then exponentially mapped into $\tau_i \in [10^{-1.0},10^{2.0}]$. The lower bound of τ_i corresponds to the integration step-size used to

update the controller; the upper bound, arbitrarily chosen, corresponds to about 4% of the maximum length of a trial.

3.5 The fitness function

During evolution, each genotype is translated into an arm controller and evaluated more than once for all the object-action *experienced* instructions by varying the starting positions. The agents perceive *experienced* instructions and they are required to execute the corresponding behaviours. Agents are evaluated 14 times initialised in the left and 14 times in the right initialisation area, for a total of 28 trials. For each initialisation area, an agent experiences all the *experienced* linguistic instructions twice. The *non-experienced* linguistic instructions $Inst^{M}_{blue}$ and $Inst^{T}_{green}$ are never experienced during the training phase. At the beginning of each trial, the agent is randomly initialised in one of the two initialisation area, and the state of the neural controller is reset. A trial lasts 25 simulated seconds ($T = 250$ time steps). A trial is terminated earlier in case the arm collides with a non target object. In each trial k, an agent is rewarded by an evaluation function which seeks to assess its ability to execute the desired action on the target object.

3.5.1 With-Indicate

In **With-Indicate**, the fitness F^{tot}_k attributed to an agent in trial k is the sum of three fitness components F^1_k, F^2_k, and F^3_k, averaged over all trials. F^1_k rewards the agent for reducing the angular distance between S^1 and the target object. F^2_k rewards the agent for performing the required action on the target object. F^3_k rewards the agent for extending S^2 when it is perceiving the target object and it is required to touch or to move it.

$$\begin{aligned} F^{tot} &= \tfrac{1}{K}\sum_{k=1}^{K} F^{tot}_k; \\ \text{with } K = 28;\ & F^{tot}_k = F^1_k + F^2_k + F^3_k; \end{aligned} \tag{3.4}$$

F_k^1, F_k^2, and F_k^3 are computed as follows:

$$F_k^1 = \max\left(0, \frac{d^i - d^f}{d^i} \cdot P_k^1, \mathbb{1}_{d^f<4.6^\circ}\right); \tag{3.5}$$

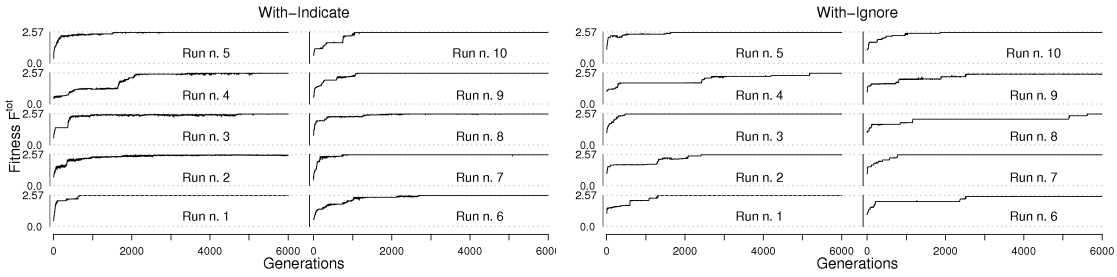
where d^i and d^f are respectively the initial (i.e., at $t = 0$) and final (i.e., at the end of the trail k) angular distances between S^1 and the target object and $\mathbb{1}_{d^f<4.6^\circ}$ is 1 if $d^f < 4.6^\circ$, 0 otherwise. P_k^1 is the penalty factor. It is set to 0.6 if the agent collides with a non target object, otherwise to 1.0. The angle between S^1 and the target object o can be measured *clockwise* (α_o^{clock}) or *anticlockwise* (α_o^{anti}). In equation 3.5, d^i and d^f are the minimum between the clockwise and anticlockwise distance, that is $d = \min\left(\alpha_o^{clock}, \alpha_o^{anti}\right)$.

$$F_k^2 = \begin{cases} \dfrac{\textit{steps-on-target}}{\textit{max-steps-on-target}} \cdot P_k^2; & \text{for TOUCH or INDICATE} \quad (3.6a) \\ \dfrac{\Delta\theta}{\textit{max-angular-offset}} \cdot P_k^2; & \text{MOVE} \quad (3.6b) \end{cases}$$

where *max-steps-on-target* $= 100$, $P_k^2 = 0$ if $F_k^1 < 1$ otherwise $P_k^2 = 1$, *max-angular-offset* $= 34.4^\circ$. For the action INDICATE, *steps-on-target* refers to the number of time steps during which $F_k^1 = 1$, and S^2 does not touch the target object. For the action TOUCH, *steps-on-target* refers to the number of time steps during which $F_k^1 = 1$, S^2 touches the target object by activating the touch sensor T^r, and S^1 does not change its angular position. $\Delta\theta$ is the angular displacement of the orientation of S^1 recorded while $F_k^1 = 1$, and S^2 is touching the target object by activating the touch sensor T^r.

$$F_k^3 = 1.0 - \frac{\beta}{0.5\pi}; \tag{3.7}$$

with β corresponding to the angular position of S^2 with respect to S^1. F_k^3 is computed only when the target object is falling within the visual field of the agent and in those trials in which the agent is required to touch or to move the target object. If the current linguistic instruction requires the agent to indicate an object and during the time of a trial in which the agent is not perceiving the target object $F_k^3 = 0$. A trial is terminated



(a) (b)

Figure 3.3: Graphs showing the fitness curves of the best agent at each generation of ten evolutionary Runs in condition (a) **With-Indicate**; (b) **With-Ignore**.

earlier if *steps-on-target* $=$ *max-steps-on-target* during the execution of INDICATE or TOUCH and when $\Delta\theta =$ *max-angular-offset* during the execution of MOVE.

3.5.2 With-Ignore

With-Ignore differs from **With-Indicate** only in the computation of F_k^1 and F_k^2 during the execution of the linguistic instructions IGNORE Blue object $Inst_{blue}^{IG}$, IGNORE Green object $Inst_{green}^{IG}$, and IGNORE Red object $Inst_{red}^{IG}$. During the trials in which an agent is required to IGNORE an object $F_k^1 = 1$ if at the end of the trial the target object does not fall within the visual field of the agent, otherwise $F_k^1 = 0$.

$$F_k^2 = \frac{\textit{steps-out-of-target}}{\textit{max-steps-out-of-target}} \cdot P_k^2; \text{ for IGNORE} \tag{3.8}$$

where *max-steps-out-of-target* $= 100$, and *steps-out-of-target* refers to the number of time steps during which $F_k^1 = 1$, and S^2 does not touch the target object.

3.6 Results

For each experimental condition (**With-Indicate**, and **With-Ignore**), we run ten evolutionary simulations for 6000 generations, each using a different random initialisation. Recall that our objective is to generate agents that are capable of successfully accessing and executing *experienced* linguistic instructions. Moreover, we are interested in investigating whether agents develop semantic structures that are functionally com-

positional. Agents endowed with functionally compositional semantics should be able to successfully access and execute *experienced* linguistic instructions and to generalise their linguistic and behavioural skills to *non-experienced* instructions (i.e., linguistic instructions never experienced during training). We run two different series of simulations to test whether a different training bears upon the development of the required mechanisms for compositional semantics.

Figure 3.3 shows the fitness of the best agent at each generation of ten evolutionary Runs per condition. All the curves reach a stable plateau with fitness either firmly fixed or progressing with small oscillation around the maximum score (i.e., $F^{tot} \simeq 2.57$). There are Runs in which the agents reach the maximum fitness very quickly (e.g., Run n° 1 condition **With-Indicate**, or in Run n° 2 condition **With-Ignore**) other in which it takes longer (e.g., Run n° 4 condition **With-Indicate**, or in Run n° 3 condition **With-Ignore**). For all the Runs, before reaching the last fitness plateau, we have periods of very rapid fitness growth induced by the acquisition of new skills to access and execute either entire linguistic instructions or just single linguistic labels. These periods are always followed by either long or short fitness plateaus characterised by rather small oscillations. Just by looking at the fitness curves, we can say that, at the end of the simulation, most of the best agents in both conditions looked capable of correctly solving the linguistic task. However, to estimate the effectiveness and robustness of some of the best evolved agents, with respect to the initial position of the arm, we post-evaluated them for a larger number of trials.

3.6.1 First post-evaluation test: Performances on *experienced* and *non-experienced* linguistic instructions

In the first post-evaluation test, the best 5 agents of each generation, from generation 4000 to generation 6000, of each evolutionary Run in both conditions, have been repeatedly post-evaluated in each *experienced* and *non-experienced* linguistic instruction. We decided to test the best 5 agents instead of the best one of each generation, because, during evolution, the agents have been ranked according to their fit-

ness, which does not take into account the agent capability to access and execute *non-experienced* linguistic instructions. Recall that *non-experienced* linguistic instructions have not been presented during evolution. Thus, with respect to the capability to access and execute *non-experienced* linguistic instructions, the best agent of each generation may not represent the most effective solution that appeared at each evolutionary time. Overall, 100000 agents per condition have been post-evaluated (i.e., 5 agents, times 2000 generations, times 10 Runs).

During this post-evaluation test, each agent is required to execute 80 times each of the nine instructions (40 trials with the agents randomly initialised in the right initialisation area and, 40 trials in the left one, see also Figure 3.1b). The position of the objects is also randomly varied as explained in Section 3.2. In each trial k, an agent can be either successful or unsuccessful. It is successful if $F_k^{tot} = 1$, otherwise it is unsuccessful (see equation 3.4, Section 3.5 for details on F_k^{tot}). At the end of the post-evaluation test, an agent capability to solve the linguistic and behavioural task is represented by nine scores, one for each linguistic instruction. Recall that each score ranges from 0 to 80, and it represents the number of times an agent is successful at the execution of the corresponding linguistic instruction.

It is worth noting that, the results of this test gave us a rather heterogeneous picture, with performances that, even for a single agent, vary remarkably from one linguistic instruction to the other. We felt that readings and interpreting these data by only concentrating on general trends, it would have significantly impoverished the message or this research work. Therefore, we chose a way of representing the results which gives the reader a coherent and exhaustive, although a bit articulated, synthesis of what the post-evaluated agents are capable of doing at the linguistic task. In particular, for each condition, the performances of the agents are compared with respect to four different sub-tasks. For each sub-task, the comparison were accomplished by grouping the 100000 agents in eleven different categories. We first describe what the sub-tasks are and then we explain the meaning of each category.

Sub-task I takes into account only the seven scores recorded during the execution of the *experienced* linguistic instructions.

Sub-task II takes into account the seven scores recorded during the execution of the *experienced* linguistic instructions plus the score recorded during the execution of the *non-experienced* linguistic instruction MOVE Blue object.

Sub-task III takes into account the seven scores recorded during the execution of the *experienced* linguistic instructions plus the score recorded during the execution of the *non-experienced* linguistic instruction TOUCH Green object.

Sub-task IV takes into account all the nine scores (i.e, seven of them for the *experienced* instructions plus two for the *non-experienced* instructions).

For each sub-task, the agents are allocated to one of eleven possible *performance categories* (from Cat^0 to Cat^{10}). For a given sub-task, an agent is assigned to Cat^n with $n \in [0,..,10]$, if its lowest score among those considered for that particular sub-task, is within the interval $(80\frac{n-1}{10},..,80\frac{n}{10}]$. Cat^0 comprises all agents that failed to complete a single trial out of 80 attempts on at least one of the instructions. The higher the category, the better the overall performance of the agent. For example, Cat^6 subsumes those agents for whom the lowest score among those considered in a given sub-task is within the interval $(40,48]$. Cat^7 subsumes those agents for whom the lowest score among those considered in a given sub-task is within the interval $(48,56]$, etc. Let's consider an agent whose performances at the post-evaluation test are represented by the following nine scores vector (80, 80, 80, 80, 80, 80, 80, 52, 67), in which the first seven scores refer to the performances while executing *experienced* instructions, the eighth score refers to the performance while executing the *non-experienced* instruction TOUCH Green, and the ninth score refers to the performance while executing the *non-experienced* instruction MOVE Blue object. This agent would be assigned to the following categories: i) category Cat^{10} as far as it concerns sub-task I; ii) category Cat^9 as far as it concerns sub-task II; iii) category Cat^7 as far as it concerns sub-task III, and sub-task IV.

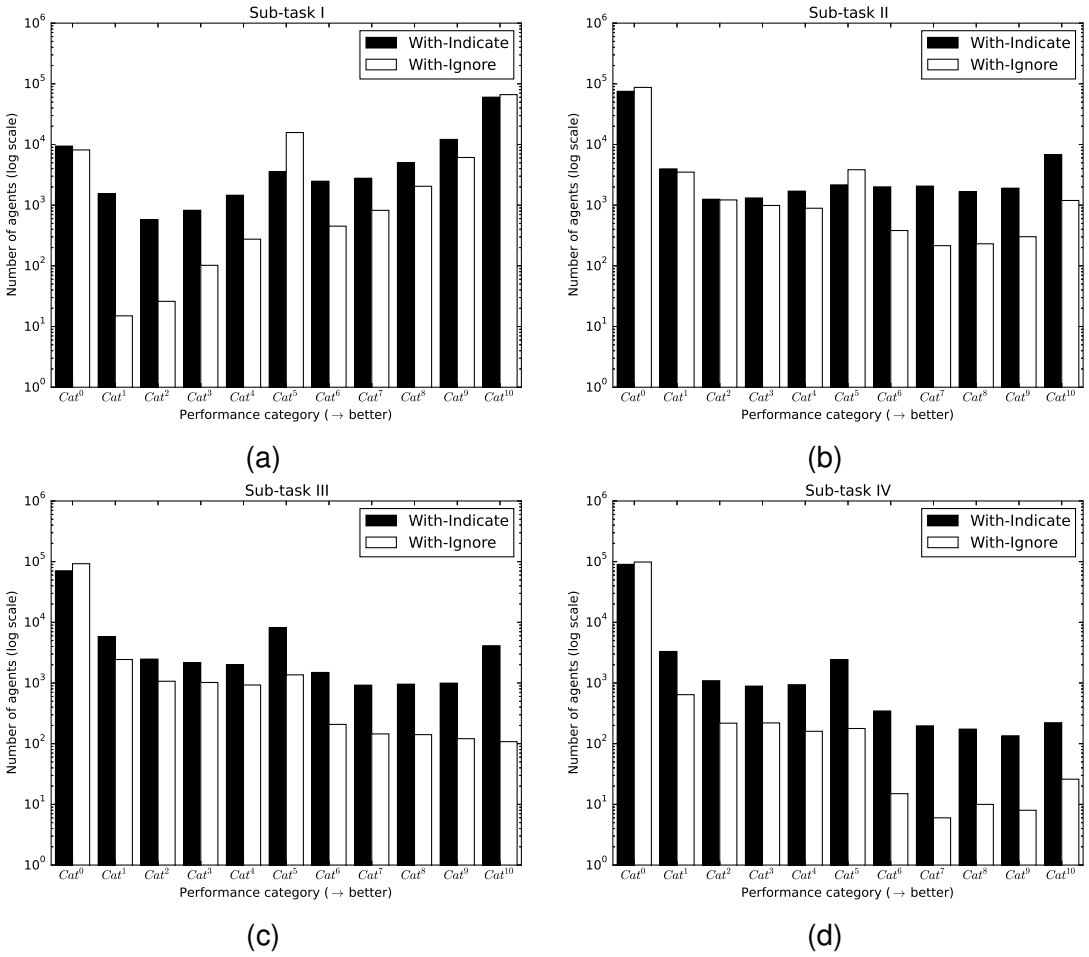


Figure 3.4: The results of post-evaluation tests. Graphs (a) to (d) refer to the four different sub-tasks. Bars represent the number of agents in each performance category Cat^n, with higher n corresponding to better performance. Note that the y axis is in logarithmic scale. See text for details.

Table 3.2 shows the number of post-evaluated agents for each category and for each sub-task, and Figure 3.4 shows the results divided by subtask, to be able to easily compare the **With-Indicate** and **With-Ignore** conditions (black and white bars, respectively). Please note that the y axis (i.e. the number of agents in each category) is in logarithmic scale, to make it evident when the number of agents in a certain category differ by one or more orders of magnitude in the two experimental conditions while also allowing to plot on the same graph values that range from few units to thousands.

The results can be summarised in the following:

- for both conditions, more than half of the post-evaluated agents (about 60% of the agents in **With-Indicate**, and about 66% of them in **With-Ignore**), are perfectly capable of accessing and executing the seven linguistic instruction *experienced* during evolution (see sub-task I, Cat^{10}, condition **With-Indicate**, and **With-Ignore**). This is expected from what was previously observed in the fitness curves shown in Figure 3.3.

- for both conditions, only a very small number of post-evaluated agents is perfectly capable of accessing and executing all the *experienced* plus one single *non-experienced* linguistic instruction, no matter which one of the two we consider (see Table 3.2 and Figure 3.4, sub-task II, and III, Cat^{10}, condition **With-Indicate**, and **With-Ignore**). The great majority of the agents in sub-task II and III completely fails to access and execute exactly the single *non-experienced* linguistic instruction included in the corresponding sub-task. This has been confirmed by further checks on the data. However, it can also be inferred from the fact that the same agents that are in Cat^{10} for sub-task I tend to be in Cat^{0} for sub-tasks II and III.

- for both conditions, only a tiny fraction of the post-evaluated agents is perfectly capable of accessing and executing both the *experienced* and *non-experienced* linguistic instructions (see Table 3.2 and Figure 3.4, sub-task IV, Cat^{10}, **With-Indicate**, and **With-Ignore**).

From these results, it clearly emerges that only a tiny fraction of the post-evaluated agents is capable of accessing and executing all the linguistic instructions, independently from the initial position of the arm. However, since the number of agents in condition **With-Indicate**, Cat^{10}, sub-task II, III and IV, is significantly different from the number of agents in condition **With-Ignore**, Cat^{10}, sub-task II, III and IV (pairwise Wilcoxon test with 99% confidence interval, compare also the black and white bars for Cat^{10} in Figure 3.4b, 3.4c and 3.4d), we conclude that condition **With-Indicate** facilitates the evolution of agents capable of accessing and executing both *experienced*

Table 3.2: Results of post-evaluation tests showing, for each evolutionary condition, the number of agents for each performance category and for each sub-task. The total number of post-evaluated agents per condition is 100000 (i.e., 5 agents, times 2000 generations, times 10 Runs).

	With-Indicate			
	Sub-task I	Sub-task II	Sub-task III	Sub-task IV
Cat^0	9408	75200	70787	90263
Cat^1	1545	3962	5840	3313
Cat^2	578	1252	2477	1092
Cat^3	823	1314	2174	889
Cat^4	1458	1703	2016	939
Cat^5	3558	2161	8217	2430
Cat^6	2483	2004	1493	346
Cat^7	2780	2061	922	197
Cat^8	5020	1668	957	174
Cat^9	12116	1906	995	135
Cat^{10}	60231	6769	4122	222
Total	100000	100000	100000	100000
	With-Ignore			
Cat^0	8127	87238	92457	98516
Cat^1	15	3502	2439	643
Cat^2	26	1220	1069	218
Cat^3	102	989	1021	220
Cat^4	275	890	928	160
Cat^5	15733	3836	1363	178
Cat^6	451	382	208	15
Cat^7	822	215	145	6
Cat^8	2049	231	141	10
Cat^9	6107	302	121	8
Cat^{10}	66293	1195	108	26
Total	100000	100000	100000	100000

and *non-experienced* linguistic instructions. In other words, evolutionary pressures to evolve a behavioural repertoire to execute the INDICATE behaviour seem to facilitate the development of compositional semantics. In the next Section, we will further investigate this issue and present a closer look at what makes condition **With-Indicate** more suitable to the evolution of compositional semantic structures.

Obviously, it is important to emphasise the fact that the evolutionary conditions detailed in previous Sections, and in particular those in condition **With-Indicate**, generate the neural mechanisms required by the agents to go beyond what was experienced during evolution. Nevertheless, the fact remains that in either condition, the agents capable of generalising their skills are only a tiny fraction of the agents capable of successfully accomplishing the evolutionary task. This can be explained by the fact that: (i) evolution only seldom produced agents fully capable of generalising their skills; (ii) the selective process does not differentiate between compositional and non-compositional agents since they tend to produce equally good performance with respect to the conditions in which they are evaluated. We noticed that agents capable of generalising appear only in six Runs out of ten, and they are never more than one or two per generation[2]. When they appear, they generally have the highest fitness recorded at that particular generation, which almost always is the highest possible fitness. However, they tend to appear when there are already many more agents with the same fitness in the population that are nevertheless not capable of generalising their linguistic and behavioural skills to *non-experienced* linguistic instructions. The selection mechanism, which can not distinguish on the basis of the fitness alone, agents capable of generalising from those not capable of generalising, tends to favour the latter, to the detriment of the former, simply because the latter are more frequent in the population.

A final point of minor significance is that generalisation capabilities with respect to the MOVE Blue object instruction are more frequent than that with respect to the TOUCH Green object instruction. That is, for both conditions, the number of agents in Cat^{10} sub-

[2]Data not shown in this chapter can be found at `http://laral.istc.cnr.it/esm/tuci-etal-IEEE_TAMD2010/`.

task II is significantly different from the number of agents in Cat^{10} sub-task III (pairwise Wilcoxon test with 99% confidence interval). Although we have no empirical explanation for this finding, we know that the action MOVE, which requires the agents to rotate both arms around their joints, is an action that, in evolutionary terms, appears earlier than the capability to TOUCH an object, which requires the agents to stop rotating both arms. At the beginning of the evolution, when the agents' linguistic and behavioural skills are rather simple, it pays more to be able to rotate the arms in order to approach the target objects, rather than to be able to stop a not existing yet rotation of the arms. This evolutionary progression of the behavioural skills may explain why the *non-experienced* instruction which requires to MOVE a target object turns out to be more easily accessible and executable than the *non-experienced* instruction that requires to TOUCH a target object.

3.6.2 Compositionality: Operational principles

What kind of operational principles do agents employ to be able to access and execute both *experienced* and *non-experienced* instructions? What are the mechanisms underpinning compositional semantics? By visually inspecting the behaviour of some of the agents, we notice that, contrary to the behaviour of the agents evolved in condition **With-Ignore**, the behaviour of compositional agents evolved in condition **With-Indicate** is the result of the combination of two types of elementary behaviour: an "INDICATE Red object" or "INDICATE Green object", or "INDICATE Blue object" behaviour produced during the first phase of the trial, eventually followed by a "TOUCH" or "MOVE" behaviour, in the second phase of the trial. During the first phase of the trial, regardless of the action to be performed on the object, the agents search the target object by rotating S^1 in order to reduce the angular distance between the target object and S^1, keeping S^2 bent as at start until the target object falls into the agent visual field. During the second phase of the trial, regardless of the target object, the agents rotate S^2 without moving S^1 if TOUCH is required. They rotate S^2 until this segment collides with the target object and then they start rotating S^1 again if MOVE is required. They

keep S^1 pointing to the object and S^2 fully bent as at start if INDICATE is required. This qualitative analysis of the behaviour of compositional agents suggests that the agents have developed behavioural skills that, being independent from the particular nature of the linguistic instructions in which they are employed, can be used in contexts already experienced as well as in context not experienced during training.

From this observation, we hypothesised that compositional semantics is underpinned by simple mechanisms by which, during the first part of the trial, the agents regulate their actions on the basis of the object-label and not on the basis of the action-label, and viceversa, during the second part of the trial. This quite intuitive hypothesis suggests that, in any given trial, there may be a first temporal phase, which starts right at the beginning of the trial, in which agents access the part of the linguistic instruction that defines the target object (i.e., the object-label) and act in order to execute the appropriate search behaviour. During this phase, the other part of the linguistic instruction (i.e., the action-label) should not influence the agent's behaviour. The first phase would be followed by a second one, which begins roughly when the target object is visually found. In the second phase, the agents regulate their behaviour on the basis of the action-label only (i.e., the object-label does not have any influence) in case the instruction is TOUCH or MOVE. In the case of INDICATE, instead, the agents keep producing the same behaviour during the entire trial. On this account of compositionality, linguistic instructions not experienced during training (i.e., MOVE Blue object, TOUCH Green object), would be:

- accessed by exploiting the capability to extract from a *non-experienced* instruction already experienced linguistic labels (i.e., TOUCH, MOVE, Blue object, and Green object).

- executed by calling upon known elementary behaviours associated to or triggered by one or the other linguistic label.

In what remains of this Section, we show the results of two post-evaluation tests de-

signed in order to verify whether the agents temporally and functionally decompose the linguistic and behavioural task into two sequential phases as suggested by our hypothesis. These tests are referred to as the *action-transition test* and the *object-transition test*. Both tests follow a similar logic. In the *action-transition test*, the action-label is changed during the course of a trial, while in the *object-transition test*, the object-label is changed during the course of a trial. In both tests, the change takes place in a single time step randomly chosen within a 10 time steps interval which starts at the time when the target object falls within an agent visual field. Based on our hypothesis, agents equipped with compositional semantics are expected to execute the second given action-label and neglect[3] the first given one, at the *action-transition test*. This is because the first given action-label is experienced during the first phase of a trial, when the attention of the agents should be focused on the object-label. At the *object-transition test*, these agents are expected to neglect the second given object-label. This is because this object-label is experienced during a time in which the agents already see the first given target. Consequently, they should pay attention only to the action-label.

The *action-transition test* and the *object-transition test* have been run on a pool of agents selected on their results at the first post-evaluation test (see Section 3.6.1). In particular, for each evolutionary condition (i.e., **With-Indicate**, and **With-Ignore**), we chose the agents that proved to be more than 75% successful at executing each *experienced* instruction. For the purposes of these tests, these agents have been further selected, and the following three categories have been created: i) *non-compositional* agents, referring to those agents that, at the first post-evaluation test, proved to be less than 10% successful at executing each of the *non-experienced* instructions; ii) *partially-compositional* agents, referring to those agents that, at the first post-evaluation test, proved to be more than 75% successful at executing only one of the two *non-experienced* instructions, and less than 10% successful at executing the other *non-*

[3] In this Section, we often take an anthropomorphic stance, by talking about agents that attend or neglect linguistic labels. This is purely for ease of exposition. It is not our intention to claim that the agents are cognitively rich enough to intentionally attend or neglect sensory stimuli.

experienced instructions; iii) *fully-compositional* agents, referring to those agents that, at the first post-evaluation test, proved to be more than 75% successful at executing each of the *non-experienced* instructions.

For both tests, the agents are evaluated 80 times (i.e., 80 trials) on each transition. In half of the trials, the agents are randomly initialised in the right, and in half of the trials, in the left initialisation area. In each trial k, an agent can either succeed, if at the end of the trial $F_k^{tot} = 1$, or fail, if $F_k^{tot} < 1$. Following the logic of each test, the fitness components F_k^1, F_k^2, and F_k^3 are updated with respect to the execution of the second given action-label on the current target object, in the *action-transition test*, and with respect to the execution of the current action-label on the first given target object, in the *object-transition test*. For both tests, an agent's overall performance on each specific transition is considered a success if the agent successfully executes the transition in more than 60 out of 80 trials (i.e., 75% success rate). Since both tests are indiscriminately done on *non-compositional*, *partially-compositional*, and *fully-compositional* agents, we removed from the two sets of possible transitions, those which, assuming our hypothesis holds, require a response that *non-compositional*, and *partially-compositional* agents are not capable of performing. That is, we remove those transitions which require a MOVE Blue object, or a TOUCH Green object response[4].

Figure 3.5a and 3.5b show the results of the *action-transition test* and of the *object-transition test*, respectively. In both graphs, each bar indicates the percentage of agents that managed to obtain a success rate higher than 75% in all possible transitions of the corresponding test. Black bars refer to the agents evolved in condition **With-Indicate**, white bars refer to the agents evolved in condition **With-Ignore**. Before commenting the results, the reader should be aware of the following. These are quite severe tests since they demands a high success rate on part of the agents on each experienced transition. If our hypothesis on the mechanisms underpinning compositionality holds,

[4] In particular, in the *action-transition test*, the transitions experienced by the agents are those in which the second given action-label in combination with the object-label does not produce a *non-experienced* instruction. Similarly, in the *object-transition test*, the transitions experienced by the agents are those in which the first given object-label in combination with the action-label does not produce a *non-experienced* instruction

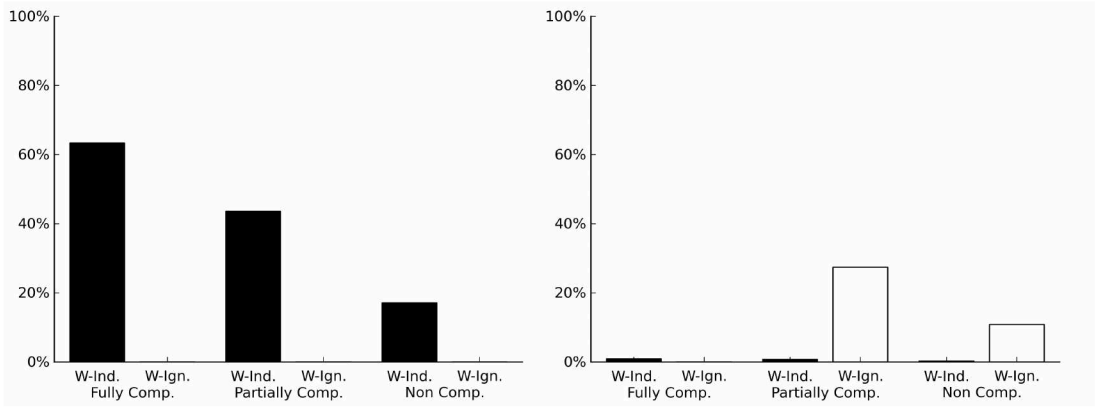


Figure 3.5: Graphs showing the results of the (a) *action-transition test*; (b) *object-transition test*. In both graphs, each bar indicates the percentage of agents that managed to obtain a success rate higher than 75% in all possible transitions of the corresponding test. Black bars refer to the agents evolved in condition **With-Indicate**, white bars refer to the agents evolved in condition **With-Ignore**. See the text for the definition of *fully-compositional*, *partially-compositional*, and *non-compositional* agents.

we expect *non-compositional* and *partially-compositional* agents to be very bad at least in one of the experienced transitions. This is because we assume that the test can be successfully performed only by agents possessing the capability to functionally and temporally decompose the linguistic and behavioural task into two sequential phases, and that this capability can only be found in *fully-compositional* agents. However, the agents may not need to fully decompose every single trial into two sequential phases in order to be able to successfully access and execute *non-experienced* instructions. In this sense, the test may demand more than what is required to be capable of behavioural and linguistic generalisation. Moreover, in these tests the agents' performance is influenced by whether the label change takes place exactly at the time when the agents switch the focus of their attention from the object-label to the action-label. For methodological convenience, we treated all the agents in the same way, by arbitrarily making this switch in a single time step randomly located in a 10 time steps interval that starts when the agents see the target object. Nevertheless, this may not fully comply with each agent's own strategy, causing failure even in those agents that can functionally and temporally decompose the task.

In spite of these limitations, these graphs tell us several important things. We first concentrate on the results of the *action-transition test*. Figure 3.5a indicates that the majority of *fully-compositional* agents evolved in condition **With-Indicate**, relies on strategies in which the action-label does not influence the agents' behaviour during the first phase of the task (see Figure 3.5a, black bar on the left). This suggests that the capability to neglect the action-label while searching for the target object is associated with the presence of compositional semantic structures, since it tends to be observed in *fully-compositional* agents. However, some of the *partially-compositional* and *non-compositional* agents in condition **With-Indicate** proved also capable of accomplishing their task without failing in any transition of the *action-transition test* (see Figure 3.5a, central and right black bars). Thus, the first conclusion we draw is that neglecting the action-label while reaching the target object is not sufficient to attain compositionality, since it does not allow those *partially-compositional* and *non-compositional* agents that possess it to access and execute *non-experienced* instructions.

Figure 3.5a also shows that the capability to cope with the action-label change is completely absent in the agents evolved in condition **With-Ignore**. This result seems to suggest that the significant differences, illustrated in the previous Section, between the two evolutionary conditions in the generation of agents capable of accessing and executing *non-experienced* linguistic instructions, could be explained by the fact that solutions based on the combination of independent elementary behaviours are more rare in the **With-Ignore** condition. Thus, we further conclude that the condition **With-Indicate** seems to contain the evolutionary pressures that facilitate the emergence of compositionality by indirectly favouring those agents whose behaviour is not influenced by the action-label while they reach the target object .

Figure 3.5b, which refers to the *object-transition test*, tell us that the capability to neglect the object-label during the second phase of a trial, when the target object is already within an agent's visual field, is completely absent in agents evolved in condition **With-Indicate**, and in particular is completely absent in *fully-compositional* agents. Only

some of the *partially-compositional* and of the *non-compositional* agents evolved in condition **With-Ignore** seem to be able to cope with the object-label change (see Figure 3.5b, central and right white bars). How do we explain these results? As far as it concerns the unexpected failure of the *fully-compositional* agents evolved in condition **With-Indicate**, we found out that, contrary to what hypothesised by us, the agents use the object-label during the second phase of the task to keep the target object within their visual field. We observed that, when the object-label does not match what is visually perceived, *fully-compositional*, *partially-compositional*, and *non-compositional* agents perform a search behaviour, loosing visual contact with the object indicated by the first given object-label. Thus, the object-label influences the agents' behaviour during both the first and second phase of a trial, by triggering the agents' response of searching and orienting toward the appropriate object. As far as it concerns the performance of the agents evolved in condition **With-Ignore**, we think that their successes at the *object-transition test* can be explained by considering the evolutionary circumstances in which they evolved. In particular, the action IGNORE can be accomplished by executing a common act for all the objects. Behavioural inspections have indeed demonstrated that *partially-compositional* and *non-compositional* agents evolved in condition **With-Ignore** and capable of coping with the object-label change, once required to IGNORE an object simply don't move at all from their position. This is a strategy which can be successfully applied to execute the action IGNORE regardless of the target object. This may have facilitated the emergence of mechanisms to be able to neglect the object-label while executing the required action. However, this is speculative and further analyses are required to test it.

Overall, these tests indicate that in *fully-compositional* agents obtained in condition **With-Indicate**, the "INDICATE Red object", "INDICATE Blue object", and "INDICATE Green object" behaviours are executed during the entire trial, as demonstrated by the fact that the agents are able to search for a new object and then keep indicating it when the object-label is modified during the second phase of the trial. The execution of the "INDICATE" behaviour during the second phase of the trial is not apparent in normal

condition (i.e., when the position or the colour of the objects do not change) simply because the execution of this behaviour do not produce any movement. Thus, during the second phase of the trial, when the action label is “INDICATE”, agents keep producing the same behaviour. When the action label is “TOUCH” or “MOVE”, the agents perform the corresponding elementary behaviour that operates in parallel with the “INDICATE” behaviour. The key mechanism that enables compositionality, therefore, is the fact that the action-label does not affect the agents behaviour during the first part of the trial. In other words, “TOUCH” and “MOVE” behaviours constitute independent behavioural units realised through the execution of the same sequence of micro-actions irrespectively from the object-label. Moreover, we can now state that a different training bears upon the development of the required mechanisms for compositional semantics, and that condition **With-Indicate** facilitates the emergence of compositionality by favouring the emergence of the functional independence of the action-label from the behavioural experience of searching for the target object.

Indeed, by looking at the phylogeny of *fully-compositional* and *partially-compositional* agents in condition **With-Indicate**, we notice that in early stages of their evolutionary history, one of the first behavioural skill to appear is indeed related to the capability of these agents to systematically reduce the angular distance between S^1 and the target object regardless of what type of action the current linguistic instruction is demanding. For example, the ancestors of *fully-compositional* agents, when required to MOVE or to TOUCH an object, they successfully bring S^1 in correspondence of the target object, and they keep S^2 bent until the end of the trial, by systematically failing to execute the action MOVE and TOUCH. In other words, these agents proved to be capable of accessing the linguistic label that defines the object without being able to appropriately execute the linguistic label that defines the TOUCH and MOVE actions. The ability to handle these type of actions is developed later. This can be considered a further evidence that, since the early generation of evolution in condition **With-Indicate**, *fully-compositional* and *partially-compositional* agents learn to decompose the trial into two parts, in the first one of which their behaviour is entirely triggered by the object-label.

It is important to note that the early appearance of the capability to decompose the task into two parts is not enforced by any means by the design of the fitness function, it emerges through the dynamics of evolution, and it is facilitated in condition **With-Indicate** by the presence of the instruction INDICATE. However, in the absence of further tests, we can not exclude that these phenomena are induced by design constraints, such as the fact that the segment S^1 and the vision system are bound together. This is because, this particular constraint makes it impossible for an agent to develop a visual search strategy without concurrently acting as required by the instruction INDICATE.

3.7 Discussion: perspectives for research on child language acquisition

Computational approaches to language learning are an intensely researched topic in several disciplines (for recent reviews, see [33, 54, 60]). As yet, however, there is still a marked gap between language learning research in cognitive robotics on the one hand and language acquisition studies in computational linguistics on the other. One reason for this is the different thrust of typical research in the two disciplines: in robotics, the focus is commonly on semantic issues to do with the grounding of individual linguistic symbols in agents' sensory-motor experience [31]. In computational linguistics, the focus is usually on structural issues to do with the induction of complex grammars from unrestricted text [21, 109]. In a nutshell, roboticists tend to concentrate on words as carriers of meaning (but neglect their combinatorial properties), while linguists tend to concentrate on their grammar (but neglect their meanings).

Given this apparent opposition, it is interesting to note that a currently influential theory of child language acquisition assumes both a phenomenological continuum and a developmental connection between these two seemingly complementary learning targets (i.e., meaningful "words" and meaningless "rules" in traditional terminology). In usage-based models of language learning, children are assumed to acquire linguistic "rules" (i.e., grammatical categories and constructional patterns thereof) through piecemeal abstractions over utterance-length concrete "words" (i.e., unanalysed holo-

phrastic strings like “there+you+go” and “look+at+this” that are associated with a holistic communicative intention, see [121]). Learners’ discovery of the internal structure of these units, coupled with the realisation that the segmented building blocks can be productively recombined within the abstracted constructional patterns, marks the crucial transition from finite lexicons to open-ended grammars. From this perspective, the above experiment is therefore concerned with the emergence of a genuine breakthrough on the way to language.

Needless to say, both the learning target and the learning architecture are substantially less complex here. However, most computational models of language acquisition do not purport to provide an accurate representation of the precise learning mechanisms and processes at work in human children. Rather, the more modest aim is usually to show that it is possible to solve a given task through learning at all (i.e., without innate domain-specific biases). In this way, computational models have made an important contribution to the debate over language learnability, innateness and the purported “poverty of the stimulus” (e.g. [52, 93]). However, none of the models in these debates is grounded in the way that human children’s internal representation of language is. In other words, such research has focused on the combinatorial dimension of language alone, but has ignored the additional challenge of linking linguistic structures to the embodied conceptualisations that constitute their meanings. The present study takes steps towards closing this gap, and several of its findings can indeed be related to similar observations made in empirical studies of child language learning.

To better appreciate these connections, it will be helpful to translate aspects of the design into the terminology of usage-based models of child language acquisition. Agents’ capacity to correctly access and execute a *non-experienced* linguistic instruction corresponds to their acquisition of an “item-based construction”, for example, [move N] in the sense of [121]. As the term “item-based” implies, the generalisations that child language learners have acquired at this developmental stage do not apply across the board. For instance, they may begin to use grammatical marking on some verbs but

not on others, indicating that the more inclusive generalisation that both items belong to the same overall category has not yet been formed. Empirical evidence for such item-specific effects in early language acquisition is abundant (cf. [121]), and the theoretical vision of a transition from holophrastic units over networks of item-specific "islands" to ever more schematic grammars has also received support from different computational simulations of (non-grounded) language learning [135]. From this perspective, agents' differential performance on the two *non-experienced* instructions in the present experiment does not come as a surprise: also in child language acquisition, the transition from holophrases to compositional grammars is not instantaneous.

Similarly, also the second major finding of this study, that is the significant effect of learning condition (**With-Indicate** vs. **With-Ignore**) on agents' generalisation performance readily translates into a concept of usage-based models of child language learning: if the above assumptions about what makes the behaviour INDICATE more similar to MOVE and TOUCH than IGNORE are plausible, agents' poorer generalisation performance in condition **With-Ignore** would be said to be the outcome of a lower cue consistency (i.e., regularity of form-function mapping) of the category "Verb" in this condition. Furthermore, since such constellations of inconsistency, competition and syncretism are in fact taken to be the norm in natural language processing and learning, a look to usage-based acquisition models in linguistics could also suggest certain useful extensions of the present approach that might be worthwhile to explore in future work (e.g., studying agents' generalisation performance across more than one construction, with or without semantic similarity between actions and/or referents, with balanced or statistically skewed training input, etc.). In other words, further studies should investigate the characteristics that favour the emergence of compositional solutions (i.e., that ensure behavioural generalisation) and/or that reduce the chance to converge on non-compositional solutions. Additionally, further work could investigate the possibility to scale the model with respect to the number and the complexity of the linguistic/behavioural repertoire of the agent.

3.8 Conclusions

In this chapter, we described a robotic model that allows a simulated robot to interact with three coloured objects (a Red, a Green, and a Blue object) located in its peripersonal space by executing three actions (INDICATE, TOUCH, and MOVE) during a series of trials. In each trial, the agent receives as input a linguistic instruction and is rewarded for the ability to exhibit the corresponding behaviour. The results of this study show that dynamical neural networks designed by artificial evolution allow the robot to access and correctly execute the linguistic instructions formed by all the possible combinations of the three action-labels and the three object-labels with the exception of the instructions "TOUCH Green object" and "MOVE Red object", which are *non-experienced* during training. Post-evaluation tests showed that some of the evolved agents generalise their linguistic and behavioural skills by responding to the two *non-experienced* instructions with the production of the appropriate behaviours.

Our study shows that behavioural and linguistic competences can co-evolve in a single non-modularised neural structure in which the semantics is fully grounded in the sensory-motor capabilities of the agents and fully integrated with the neural mechanisms that underpin the agent's behavioural repertoire. Owing to the use of artificial evolution, we leave the agents free to determine how to achieve the goals associated to each linguistic instruction. This allows the agents to organise their behavioural skills in ways that facilitate the development of compositionality thus enabling the possibility to display a generalisation ability at the level of behaviours (i.e., the ability to spontaneously produce behaviours in circumstances that have not been encountered or rewarded during training).

The comparison between two experimental conditions, in one of which the action-label INDICATE is substituted with the action-label IGNORE, shows that the composition of the behavioural set significantly influences the development of solutions that generalise to *non-experienced* instructions. Only individuals evolved in condition **With-Indicate** are characterised by a particularly successful linguistic and behavioural organisation

based on the decomposition of the task into two phases, each of which can be associated with the execution of an elementary behaviour. In the first phase only the object-label bears upon the agents' behaviour by triggering the object search strategy. In the second phase, both the object-label and the action-label determine the agents' response. In particular, the object-label keeps an agent eliciting the same behaviour produced during the first phase (i.e., the agent keeps on searching/pointing the target object with the first segment of its arm). At the same time, the action-label triggers a different behaviour that consists in bending the second segment of the arm so to touch or move the object. The capability to decompose the task into two sequential phases as described above, and the use of elementary behaviours employed in different circumstances, are features that, although not sufficient *per se* to explain compositional semantics, they certainly facilitate its evolution.

The use of elementary behavioural skills to generate instructions denoting complex actions resembles the process described in [17], in which the ability to execute more complex linguistic commands, such as GRAB, is acquired by associating two or more previously acquired elementary behaviours (e.g., CLOSE-LEFT-ARM and CLOSE-RIGHT-ARM). However, in [17], the relation between complex and elementary behaviours is established through explicit teaching (i.e., through linguistic input such as: GRAB is CLOSE-LEFT-ARM and CLOSE-RIGHT-ARM). By contrast, in the experiments reported in this chapter, behavioural decomposition emerge as a side effect of the need to acquire the ability to execute several related linguistic commands. Moreover, the way in which the agents accomplished the required functionality (i.e., by combining in sequence or in parallel relatively independent behavioural units) represents an important prerequisite for the emergence of compositionality. Therefore, leaving the agents as free as possible to organise how they produce the required skills might be advantageous since it might allow them to decompose the problem in a way that maximise skills re-use. This in turn implies that methods such as the evolutionary method that rewards the agent on the basis of the ability to achieve a given functionality without specifying in details the behaviour that should be produced might be advantageous with respect

to alternative methods in that respect.

3.9 The experiment and FARSA development

The main experiment described in this chapter is, from a technical point of view, less complex than the one presented in chapter 2: the environment is a bidimensional plane and the interaction with objects is less accurately simulated.

As in the previous experiment we at first adapted code that had been used in our laboratory for other experiments. We experienced problems similar to those described in the previous chapter, such as the lack of modularity, of configurability and the difficulty for researchers not actively involved in the development of software to replicate the experiment.

This experiment, in general, reinforced the lessons drawn from the experiment in chapter 2 about the importance of having a modular software tool that is easily configurable to be able to test various hypothesis about, e.g. the effect of different sensors or effectors.

An additional requirement of this experiment was that of validating the results obtained in a bidimensional environment on a simulated iCub robot, as explained in section 3.2. To do this we use an expanded version of the simulator developed during the experiments in chapter 2, which included the model of the iCub robot. Despite the simulator was already available, however, testing the model required a substantial amount of work. This was due to the fact that the code developed to perform the bidimensional experiment was completely disconnected from the iCub simulator and, on the other hand, the simulator was not modular enough to be easily coupled with an external neural network library.

The possibility to use the same tool for both the bidimensional, kinematic setup and the tridimensional, dynamic setup would have greatly reduced the technical effort needed to complete this experiment. More in general being able to select different levels of accuracy and to consequently vary the computational cost of a simulation (and consequently the time it takes to run it) is an important property of a simulator. In fact it

may allow to perform relatively quick experiments during the initial phase of a research (when there might be a number of hypothesis to test) and then to increase the accuracy to study only few, promising hypothesis.

In chapter 4 we will describe in details how we aimed at solving there issues in FARSA. We will also give examples of experiments at different levels of accuracy, from simple kinematic setups to dynamical simulation in which different objects interact and collide with each other in a realistic way.

Chapter 4

FARSA: An Open Source Software Tool for Embodied Cognitive Science

4.1 Introduction

In this chapter we will introduce the robotic simulator developed at LARAL[1] during the past years. The FARSA (Framework for Autonomous Robotics Simulation and Analysis) software was born to address the need for a common code base shared among all people in the LARAL laboratory, which was nevertheless as customizable as possible. The tool was then released as an open source project with the hope of being useful to other researchers in the same area. Moreover the use of a tool which is multi-platform and can be easily installed, makes it possible to release the binary or source code of published experiments so that they can be easily replicated.

In addition to my own research laboratory, FARSA has already been used in the following research laboratories:

- Laboratory of Prof. Fernando M. Montes González[2], Departamento de Inteligencia Artificial, Universidad Veracruzana, Mexico;
- Laboratory of Prof. Andrea Sterbini[3], Dipartimento di Informatica, Università La Sapienza, Italy;
- Laboratory of Computational Embodied Neuroscience (LOCEN[4]), Institute of Cog-

[1] http://laral.istc.cnr.it/
[2] http://www.uv.mx/fmontes/
[3] http://twiki.dsi.uniroma1.it/twiki/view/Users/AndreaSterbini
[4] http://www.istc.cnr.it/group/locen

nitive Sciences and Technologies, National Research Council (ISTC-CNR), Italy;

- Human Evolutionary Biology group[5], Institute of Evolutionary Sciences, National Center for Scientific Research (ISEM-CNRS), France;

as well as in an undergraduate course by Prof. Marco Mirolli, at the University "Sour Orsola Benincasa" in Napoli, Italy.

FARSA is a re-engineered and extended version of a tool that has been developed since the 1995 by Stefano Nolfi and then by Onofrio Gigliotta [77, 82] which has been used for research and education purposes by more than 50 research laboratories and universities.

The chapter is structured as follows. Sections 4.2 and 4.3 describe the related tools. Section 4.4 illustrates why we decided to develop FARSA. Section 4.5 describes the tools and its features. Section 4.6, describes some of the available illustrative experiments. Section 4.7, describes how to customize and expand FARSA. Finally, conclusions are drawn in section 4.8.

4.2 Related tools: robotic middlewares

Before introducing robotic simulators available today, this section we will present several *robotic middlewares*. These software abstraction layers are often used with both real robots and simulated robots. We will briefly introduce them mainly to show what is the software architecture generally used in robotic today. We will see in section 4.4 which are the critical aspects of this architecture and why we had to use a different approach when developing FARSA.

4.2.1 Player

Player[6] [39] is a network server that runs on a robot, providing a network interface to access sensor data, configure devices and send commands to actuators. Player permits to use the same interface for accessing different devices. This enables the user

[5] `http://www.evolutionhumaine.fr`
[6] `http://playerstage.sourceforge.net/index.php?src=player`

to access to data in the same way independently from the particular type of hardware used. For example the user can access in the same way the video stream extracted from the camera regardless of the underlying hardware and low-level software.

The project aim is to create free software for research into robotics and sensor systems [38]. Today, Player, together with the Stage simulator, is one of the most popular open-source robot interface in research [26], as can be appreciated from the number of scientific papers published every year that acknowledge the use of this tool. This remarkable achievement has been reached thanks the robustness of the tool that has been reached during years of continuous and collaborative development. The source code of Player is released under the GNU Public License.

The core of the Player framework is the Player network server. The server runs on Linux, Solaris and BSD and is designed to be executed on the robot. Internally, the server runs a set of *drivers*, one for each sensor, actuator, or control algorithm. Each driver implements one or more standard *interfaces* and the same interface used by similar devices. Consequently, the user can access to sensory information or set the state of the actuators by using the high-level command provided by the interface without accessing the drivers directly.

The robot resources are typically accessed through a network connection. Consequently the controller of the robot can be implemented in different programming languages (e.g. C, C++, Python, Ruby, Java, and Tcl) and can run on computers that use different types of operating systems. The framework does not impose any constraint on the architecture of the control software, that can consist of a simple read-think-act loop, a complex multithreaded program, an interactive client, etc.

The main advantage of Player is that it potentially enables to reuse the same control software on different hardware or simulated platforms. Moreover it allows to access the same device by multiple clients, thus enabling the possibility to use different programs that, for example, control the robot and monitor the state of the robot's sensors.

4.2.2 ROS

ROS[7] (Robotic Operating System) is a software framework which provides services normally available on an operating system, on a computer or on a computer cluster. Such services include hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. The broader aim of the project is to encourage collaborative development of robotic software through the adoption of a common standards and through collaborative development.

The project started in 2007 under the name *switchyard* at the Stanford Artificial Intelligence Laboratory as part of the STAIR project [59]. From 2008 until 2013, the development of ROS was carried out primarily by Willow Garage[8], a robotics research institute. From February 2013, ROS was taken over by the Open Source Robotics Foundation[9]. The core libraries and tools of ROS are released under the BSD license, which permit to use the ROS resources both in open source and proprietary projects. Most ROS modules are released under open source licenses.

The main functionality of ROS is to enable communication of loosely coupled processes. On the basis of the ROS terminology, each process performing computations is a *node* and different nodes can run on different physical machines. Nodes communicate using *messages*, which are basically data structures with typed fields. There are basically two kind of communication mechanisms: *topics* and *services*. Topics implement a publish/subscribe protocol, in which some nodes produce data that is read by other nodes. In general nodes that publish or read data in a topic are not aware of each other. Services, instead, implement a request/reply communication scheme. A central element of the ROS network is the *ROS Master* that is responsible of keeping track of all nodes, topics and services. Nodes query the ROS Master to be able to connect to other nodes, subscribe to topics and request services. Figure 4.1 show a typical small

[7] http://www.ros.org/
[8] http://www.willowgarage.com/
[9] http://www.osrfoundation.org/

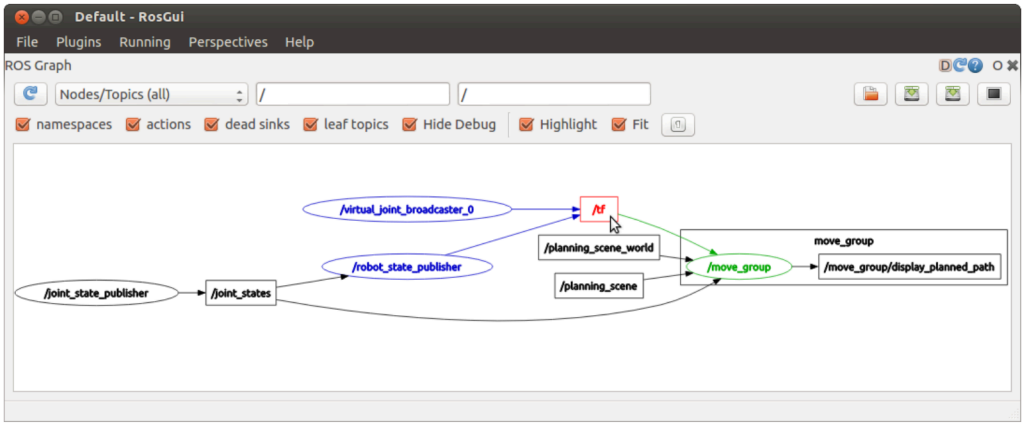


Figure 4.1: Snapshots of the rqt_graph application, a tool to visualize the ROS network. The Figure is available from http://wiki.ros.org/rqt_graph

ROS network.

The main ROS client libraries are written in C++, Python and Lisp. However, there are many other libraries, at various stage of development, that allow to use other languages such as Java, Go, and Ruby. At the moment ROS only runs on Unix-like operating systems and has been tested under Linux Ubuntu and Mac OS X.

ROS also comprises a set of associated software tools that have the purpose of facilitating its use and facilitating the development of software modules that can be shared. The basic unit for organizing software in ROS is the *package*. Packages allow to specify dependencies that are automatically retrieved and compiled during package compilation. ROS comes in distributions, released every year, that are constructed so to enable the developers to work on a sufficiently stable codebase. ROS also provide a series of resources for developers that include a public wiki, bug tracking, mailing lists and question/answer repositories.

4.2.3 YARP

YARP[10] [36, 67] (Yet Another Robot Platform) is a software middleware that has been designed to foster code sharing and re-use through software tools that enable to separate robotics software modules from hardware devices. Another goal of the YARP

[10] `http://wiki.icub.org/yarpdoc/index.html`

designers is to facilitate the usage of other middleware, such as ROS (see [36]). YARP development started in 2002. To date it has been used extensively to run experiments with the iCub humanoid robot [99] and with other humanoid platforms.

Like ROS, YARP facilitates the development of distributed robotic software components that can run on a cluster of computers and that can communicate with each other through different protocols. In the case of YARP different protocols can be used, i.e. tcp, udp, multicast, shared memory, MPI, XML/RPC, etc.

Data is exchanged through *port*s. Each process can open one or more input and output ports that can be accessed or set by multiple clients. Each connection can use a different communication protocol. Moreover, the type of the connection can be decided at runtime. A central process, called the *YARP server*, is in charge of keeping track of all the ports and connections that have been created.

4.2.4 Other Middlewares

Several other robotic middleware with similar functionalities has been developed. The RT-middleware[11] (Robotics Technology Middleware) has an architecture that is similar to ROS and YARP but consists only of a standard that has been realized into a series of different implementations using different programming languages. For example *OpenRTM-aist* is a specific implementation realized by using CORBA that has been developed by the National Institute of Advanced Industrial Science and Technology in Japan.

Microsoft Robotic Studio[12] developed by Microsoft is another middleware that permits the creation of *Decentralized Software Services* (DSS) that can eventually run on multiple computers. This software suite, which runs on the Windows operating system only, initially gained a significant popularity. However, its development and update stopped in 2012.

[11] `http://openrtm.org/`
[12] `http://www.microsoft.com/robotics/`

Finally, OpenRDK[13] is still another robotic middleware characterized by a centralized blackboard-type communication system.

4.3 Related tools: robotic simulators

This section presents, simulators and associated software tools. It will be shown that none of these tools aims to provided a complete set of libraries, fully integrated, as in the case of FARSA.

4.3.1 Webots

In this and in the following subsections we will describe a series of simulators, i.e. software tools that enable to simulate robotic agents, environments, and robots/environmental interactions. Webots[14] [68] is probably the most influential and popular simulator that has been developed. It was initially created by Olivier Michel at the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland, and then commercialized by a spin-off company led by the software creator. The simulator runs on Microsoft Windows, Linux and Mac OS X and is available in different versions, which vary with respect to the price and the features available. The tool has been used by more than 1200 universities and research centres worldwide.

Webots contains a rich library of robots that can be simulated: Aibos, Bioloids, Boe-Bot, e-puck, HOAP-2, iRobot Create, Katana, Khepera, Koala, Kondo KHRs, Nao, Pioneer, Shrimp III, Surveyor SVR-1 and others. It also contains an extensive set of objects that can be used to build a simulated world (e.g. boxes, doors, walls, lights and so on). Finally it contains an extensive set of examples and tutorials. The simulator is based on ODE[15], a well known open source library that enable to simulate the dynamics of rigid bodies.

To run an experiment the user should program at least two software components that implement the architecture of the environment and the robot controller (each robot is

[13] http://openrdk.sourceforge.net/
[14] https://www.cyberbotics.com/
[15] http://www.ode.org/

provided with an independent controller). Eventually the user can implement a supervisor software component that can be used to modify the environment during the experiment.

The functionalities of Webots can be extended by realizing software plugins, i.e. software components that are compiled independently and loaded at runtime. Plugins can be used for example to create additional display windows (that can be used, for example, to visualize the state of the robot's sensors) or to create additional types of simulated sensors and actuators.

A shortcoming of Webots is that it can only operate with the graphical interface on. This prevents the possibility to speed up the simulation by avoiding the usage of the graphic, a feature that is particular important for speeding up time-consuming experiments, e.g. experiments that require long training phases.

4.3.2 ARGoS

ARGoS[16] [92] is a 3D physic simulation tool targeted particularly at swarm robotics research. It is an open source tool that was developed during the Swarmanoid[17] European project. It was then used in the following others European projects: ASCENS[18], H2SWARM[19], E-SWARM[20] and Swarmix[21].

The simulator is written in C++ and has a highly modular architecture. Every component can be implemented as a separate plugin and loaded at runtime, so that all the relevant aspects of a simulation can be overridden. This potentially enables to extend the tool in any possible direction. A special kind of plugin, called *loop function*, can be used to implement custom-made simulation-specific extensions.

The most distinctive feature of ARGoS is the possibility to partition the simulated space

[16] http://www.argos-sim.info/
[17] http://www.swarmanoid.org/
[18] http://ascens-ist.eu/
[19] http://www.esf.org/coordinating-research/eurocores/running-programmes/eurobiosas/collaborative-research-projects-crps/h2swarm.html
[20] http://www.e-swarm.org/
[21] http://www.swarmix.org/

into sub-spaces and to assign a different physics engine to each of them. The sub-spaces must be non overlapping and it is possible to use different kinds of engines (e.g. kinematic, 2D, 3D, etc.). This allows to optimize the simulation speed by tuning the accuracy of the simulation in an appropriate manner. The simulator is also based on a multi-thread architecture that supports the utilization of computer cluster and/or multi-core CPUs.

The main drawback of ARGoS is that it does not work under Microsoft Windows. Moreover the usage of the tool requires a significant programming effort due also to the lack of an integrated graphical user interface.

4.3.3 USARSim

USARSim [19] is another open-source simulator that was initially developed by the National Science Foundation (NSF) as a research tool for *Urban Search And Rescue (USAR)* scenarios. The simulator was later extended toward a more general use. It is the official simulator of the *RoboCup Rescue Virtual Robot Competition*[22], in which teams of robots are placed in a simulated USAR scenario and are evaluated for how many people they manage to find and the portion of the environment they explore.

USARSim is based on *Unreal Engine*[23], a physics and graphic engine developed by Epic Games[24]. Despite being targeted mainly towards the realization of computer games, *Unreal Engine* can also be used to simulate robotic platforms. In fact it can be extended by using a proprietary programming language called *Unrealscript* and an interface called *Gamebots*[25]. The robots' controllers can be programmed using any programming language thanks to the possibility to use TCP sockets.

The simulator supports a wide range of robotics platforms (humanoids, wheeled, vehicles, etc.) and can be extended to support additional robotic platforms, sensors, and actuators. Extensions can be implemented through the Unrealscript programming language.

[22] http://www.robocuprescue.org/virtualsim.html
[23] https://www.unrealengine.com/
[24] http://epicgames.com/
[25] http://sourceforge.net/projects/gamebots/

The implementation of the environment can be realized through the use of a graphical tools distributed together with the Unreal engine.

USARSim is an open source project but is based on a proprietary engine (i.e. the Unreal engine). This limits the inspection and the customization of the tool and imposes the use of the Unrealscript proprietary language for the implementation of certain components.

4.3.4 Gazebo

Gazebo[26] is a general purpose open-source simulator [56] that has been developed from 2002 at the University of Sourthern California. From 2009 it become the reference simulator for the ROS community. From 2012 it became a project of The Open Source Robotics Foundation. In 2013 the simulator was used to run the Virtual Robotics Challenge, one of the DARPA Robotics Challenge. A new major version of the simulator is released every 6 months.

Gazebo has a client-server structure: the server performs the actual simulation and has no graphical user interface, while the client connects to the server and has a graphic interface (GUI) that can be used to display the world and the robot. The 3D rendering of the scene is performed by using OGRE[27], a high quality open-source graphics rendering engine. It supports different physical engines (currently ODE, Bullet[28], Simbody[29] and DART[30]). The characteristics of the world and of the robot are described in SDF files[31] by using a XML format. Plugins programmed by the user can be used to implement the robots' controllers and/or to extend the simulator.

Gazebo is often used in combination with ROS, YARP and Player to enable the possibility to test the robots' controller both in simulation and in hardware.

[26] http://gazebosim.org/
[27] http://www.ogre3d.org/
[28] http://bulletphysics.org/wordpress/
[29] https://simtk.org/home/simbody/
[30] http://dartsim.github.io/
[31] http://sdformat.org/

4.3.5 Stage

Stage[32] [129] was initially developed at the University of Southern California as part of the Player Project (see sec. 4.2.1). It is a simulator targeted toward the realization of experiments involving large number of robots. For this reason it relies on simulation techniques that have a limited accuracy but that are fast.

Stage can operate as a standalone program. In this modality the characteristics of the environment can be specified in a configuration file, and the controller of the robot can be implemented by using the C++ programming language. However, it can also be used in combination with Player. Moreover, it can be embedded directly into a program developed by the user in C++.

Stage runs on Linux and Mac OS X and it is released under the GPL open source license. However, it does not run on Microsoft Windows.

4.3.6 Others

Other available simulators include: (i) the iCub simulator developed by Tikhanoff et al. [120] that is based on YARP; (ii) a 3D physical simulator based on the NVIDIA PhysX Technology that is included in the Microsoft Robotics Developer Studio middleware described in section 4.2.4, (iii) the MORSE[33] simulator that is implemented in Python and that is targeted toward the academic research community and (iv) the V-REP[34] simulator that is multiplatform and multilanguage and is free to use for researchers and hobbists.

4.4 FARSA objectives

The primary objective of FARSA is to provide an integrated tool that has all the key components that are necessary for carrying out research in Embodied Cognitive Science and that can enable also users with limited technical expertise to set-up and to carry on embodied experiments.

[32] http://playerstage.sourceforge.net/index.php?src=stage
[33] https://www.openrobots.org/wiki/morse
[34] http://www.coppeliarobotics.com/features.html

FARSA differs significantly from the tools reviewed in sections 4.2 and 4.3. It is not a middleware and it does not natively allow accessing remote robotic resources or communicating with remote processes. However, it is not simply a simulator since it does not only consist of a tool for simulating robot/environmental interactions. It is an integrated software environment that provides also tools for building and training robot controllers, and a tool for visualizing and analysing the behaviour of the robots.

FARSA also has limitations with respect to the other tools reviewed above. In particular it currently supports a small number of robotic platforms, it does not provide tools that support the realization of highly distributed applications (as some of the middleware described above), and it only provides libraries that support the utilization of certain type of control architecture (e.g. neural network architecture) and of certain type of adaptive algorithms (e.g. evolutionary algorithm, and supervised learning algorithms).

The tool is constituted by a series of integrated software libraries that we will briefly review in the next sections.

4.4.1 The Robots/Environment Simulator

The robots/environment simulator (worldsim) is a library that allows to simulate the robot/s and the environment in which it/they operate. The library supports both individual robot simulation and collective experiments in which several robots are placed in the same environment. The physical and dynamical aspects of the robots and of the robots/environment interactions can be simulated accurately by using a 3D dynamics physics simulator or by using a faster but simplified kinematic engine. For what concerns the dynamics simulation, FARSA relies on the Newton Game Dynamics engine [53] that enables accurate and fast simulations. The underlying dynamic engine has been encapsulated so to enable the inclusion of alternative engines in future.

Currently, FARSA supports the following robotic platforms: the Khepera [72], the e-Puck [71], the marXbot [11] (see Figure 4.2, bottom) and the iCub [100] (see Figure 4.2, top). These robots have been designed by assembling a series of building blocks (physical elements, sensors, and motorized joints) that users can re-use to implement

alternative, not yet supported, robots.

In the case of the iCub, the simulator supports the same YARP interface as the real robot. This strongly facilitates the possibility to port results from simulation to reality and the possibility to integrate into FARSA projects the software modules available from the iCub software repository[35].

4.4.2 The Sensor and Motor Library

FARSA also includes a library of ready-to-use sensors and motors. In some cases, sensors and motors include software routines that pre-elaborate sensory or motor information (e.g. to reduce its dimensionality) and/or integrate different kinds of sensory-motor information (as in the case of motors that set the torque to be produced by a joint motor on the basis of the current and desired position of the controlled joint).

Wheeled robots are provided with infrared, ground, traction force, linear vision, and communication sensors, among others. Moreover, they are provided with wheels, grippers, LEDs, and communication actuators.

The iCub robot is provided with proprioceptors that measure the current angular position of the robot's joints, tactile sensors, and vision sensors among others and with actuators that control all the available DOFs.

The state of the robot's sensors and motors, as well as the state of selected variables of the robot's control system, can be graphically visualized while the robot interacts with the environment (see Figure 4.3). This provides an useful analysis and debugging tool.

4.4.3 The Controller Libraries

These libraries enable the user to design, modify and visualize the robot's control system. Currently FARSA includes two libraries that support the design of neuro-controllers. Users willing to use other architectures or formalisms can integrate into FARSA alternative libraries (see section A).

[35] `http://wiki.icub.org/iCub_documentation/`

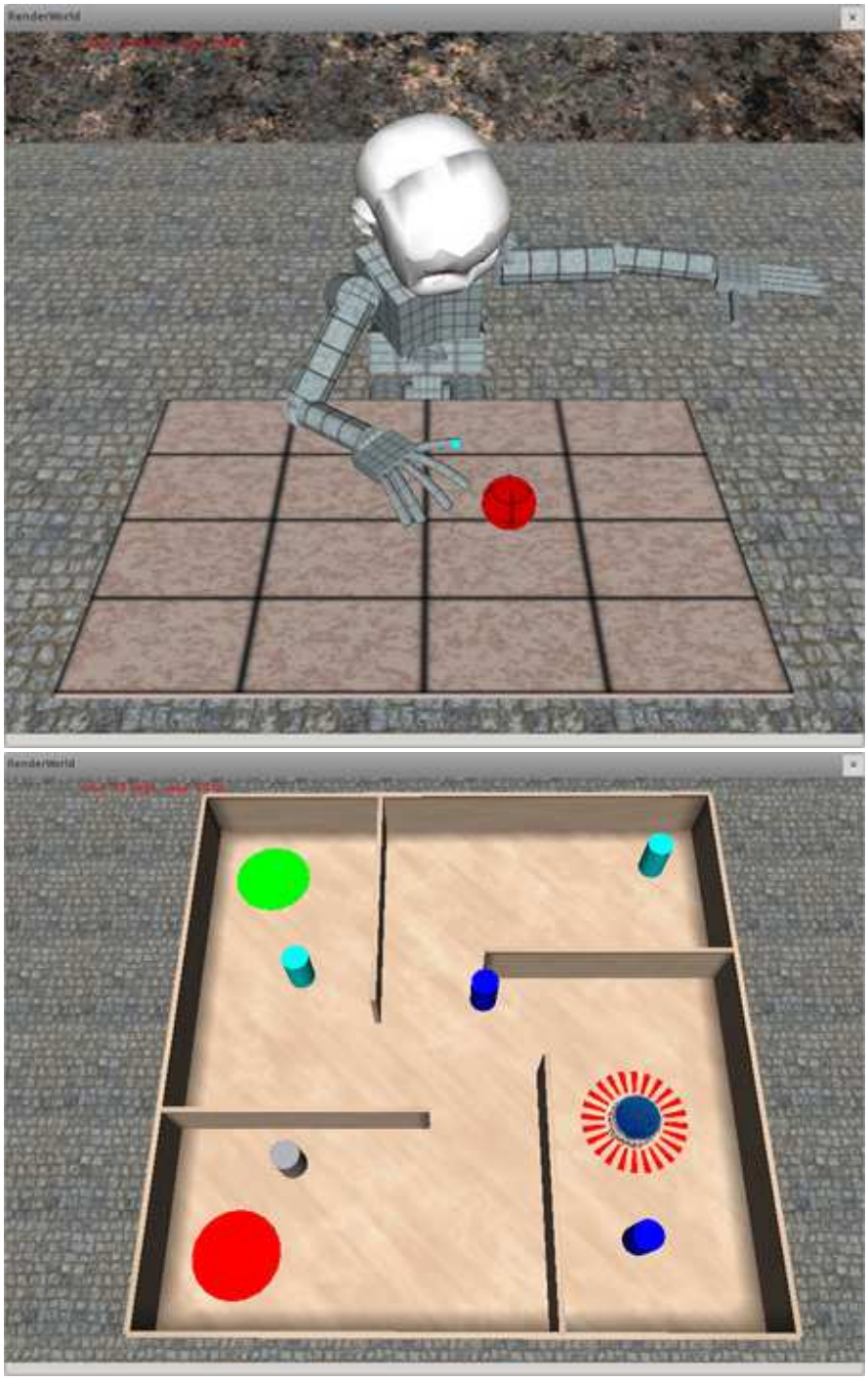


Figure 4.2: Snapshots taken from the 3D robot/environment renderer of FARSA. Top: A simulated iCub robot that reaches and grasps a spherical object located over a table. Bottom: A simulated marXbot robot that navigates in a structured environment containing walls and colored objects.

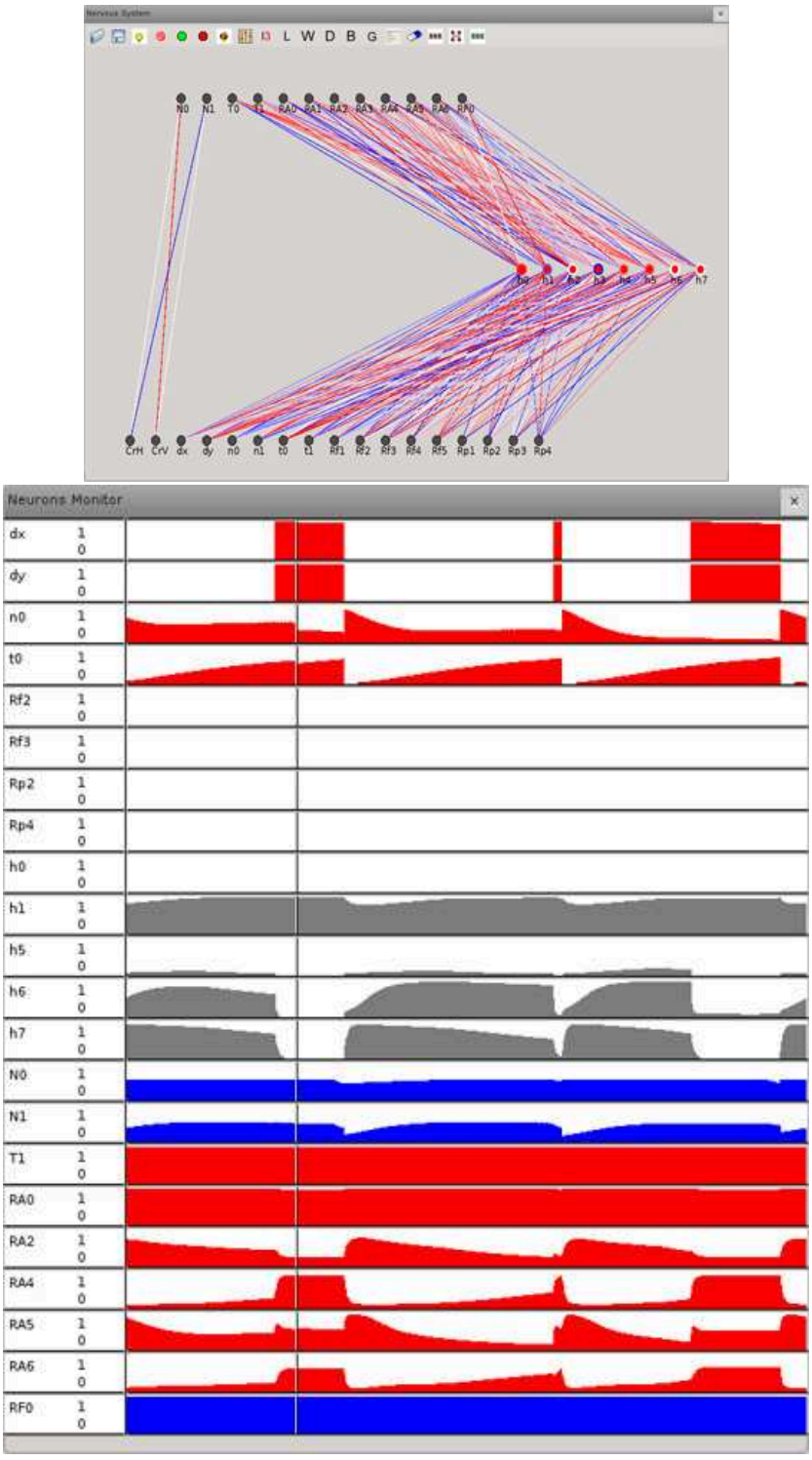


Figure 4.3: Top: the controller graphic widget that allows to visualize, modify, and analize the robot's neural architecture, the strenght of the connection weights and biases, and the properties of the neurons. Bottom: the controller monitor that displays the activation state of the sensory, internal, and motor neurons while the robot interacts with the environment.

Evonet is an easy-to-use library that enables users to graphically design, modify and visualize the architecture of the robot's neural controller as well as the properties of the neurons and of the connection weights (see Figure 4.3). The library supports logistic, leaky integrator, and threshold neurons. NNFW is an alternative object-oriented library that provides a larger variety of neuron types and output functions (Gaussian, winner-take-all, ramp, periodic, etc.) and supports the use of radial basis function neural network.

Thanks to the integration between the controller and the sensory and motor libraries, the sensory and motor layer of the neural controller is automatically generated on the basis of the selected sensors and motors. Moreover, the update of the sensory neurons and the update of the actuators on the basis of the state of the motor neurons is handled automatically.

Finally, the graphic viewer of the robot's controller (see Figure 4.3) also enables users to lesion and/or to manually manipulate the state of the sensors, internal, and motor neurons in order to analyse the relationship between the state of the controller and the behaviour that originates from the robot/environmental interaction.

4.4.4 The Adaptation Libraries

These libraries enable the user to subject a robot or a population of robots to an adapting process (i.e. to a evolutionary and/or learning process during which the characteristics of the robots are varied and variations are selected so to improve the abilities of the robots to cope with a given task/environment).

The adaptation libraries that are currently available support the use of evolutionary algorithms (including steady state, truncation selection, and Pareto-front algorithms) and supervised learning algorithms (i.e. back-propagation). The evolutionary algorithms are parallelized at the level of the individual's evaluation and can therefore run significantly faster in multi-core machines and computer clusters.

In the case of evolutionary and supervised algorithm, the variation in performance dur-

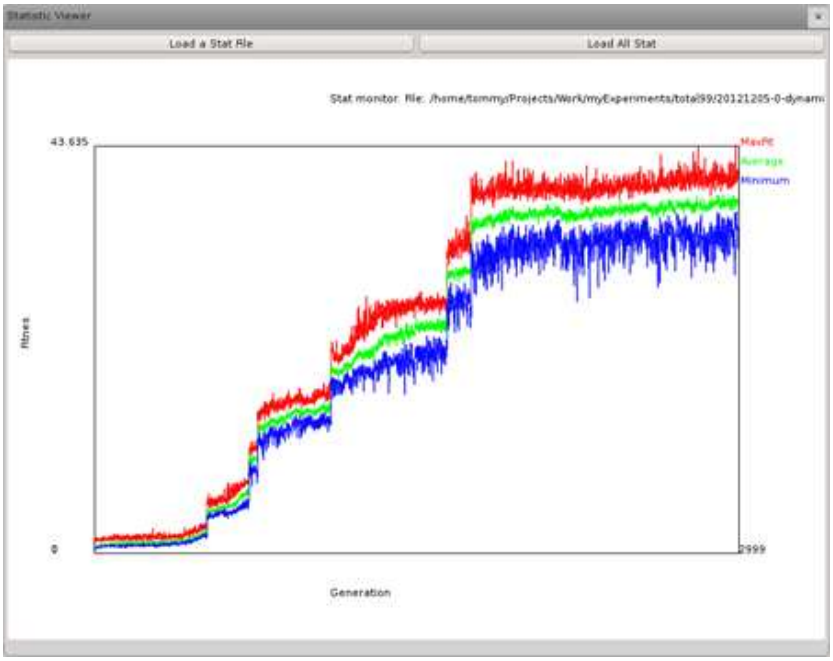


Figure 4.4: The graphic widget of the adapting process. In this example, the widget is used to show the best, average and worst fitness of an evolutionary experiment through out generations.

ing the adaptation can be monitored and analysed in the associated graphic renderer (see Figure 4.4).

4.5 Design and Working Principles

The architecture of FARSA is based on four key ideas: *components*, *configuration file*, *plugins* and *resources*.

The *components* are software modules that implement a given object or process. They can be organized in a hierarchical manner. For example, a project might include an evolutionary process component, that includes as subcomponent an experimental component, that includes as subcomponent an iCub robot component, a neural network controller component, and several sensors and motors components. The main characteristic of components is that they can be automatically instantiated and configured from the content of a *configuration file* (i.e. they have a direct relation to groups of parameters in a *configuration file*, as explained below). Components might also include associated commands (e.g. “evolve”, “stop”, “test” in the case of an evolutionary

component), and graphical widgets that can be accessed by the FARSA main graphic interface (see next section).

The *configuration file* is a text file that specifies the components (e.g. the robotic platform, the robots' sensors and the motors, the robots' controllers, and eventually the robots' adapting process) that are going to be used in a particular experiment. Moreover the configuration file include configurable parameters (e.g. the number of robots situated in the same environment, the length of the testing period etc.) that are used to configure them. The file has a hierarchical structure analogous to the hierarchical organization of components. The configuration file is a human readable text file (in .INI or .XML format) that can be edited through the Total99 graphic interface (described in the next section) or directly through a standard text editor. This enables users to configure and run experiments on remote machine (e.g. computer clusters) that do not have a graphical environment. The modular and hierarchical organization of components combined with the configuration file has several advantages:

- it allows to instantiate at runtime only the components that are needed in a particular experiment;
- it gives the possibility to re-use the same components in different projects;
- it enables a progressive expansion of the tool with the development of additional components;
- it simplifies the tool usage through the visualization of only the parameters, the commands, and the graphic widgets that are relevant for a given project/experiment.

A *plugin* contains compiled code of new components or features created by users. It might contain subclasses of existing components (e.g. a subclass of an evolutionary experiment with a new implemented fitness function or a new subclass of the sensor class implementing a new type of sensor not available in the sensor library) or of completely new components (e.g. a behaviour-based controller tool with associated para-

meters, commands and graphic widgets). The plugins, which are loaded and instantiated at run time, are totally equivalent to the other native components of FARSA for what concern the functionalities and use (e.g. they can be configured and commanded in the same manner and through the same graphic interface of the native components). Plugins provide several advantages:

- they enable users to neatly separate their new code from the main library;
- they facilitate the distribution and sharing of additional components and feature within the FARSA community;
- they enable users to get access to a number of illustrative experiments that increase over time;
- they allow authors of scientific papers to provide an easy way to replicate their work.

Overall the workflow in FARSA is as follow: the project configuration file and the required plugins are loaded, the required components are created and configured on the basis of the configuration parameters, the associated commands and graphical widgets are created and made available to the user through the graphic interface.

Resources are another useful component of FARSA that enable to share data among different components of an experiment, without the need to rely on complex interfaces or type casts. As explained above, the components that make up an experiment in FARSA are specified in a configuration file, that is loaded at runtime. This means that a component does not know which other components are part of the experiment until all components are loaded and the experiment is running. Yet it is sometimes useful to share data among different components.

The graphical interface, that is named *total99*, can be used to configure experiments, to instantiate the required software components, and to use the associated commands and graphic widgets. *total99* can also operate in batch mode without graphics if required. It can be used to create, view, or modify a configuration file (Figure 4.5). This

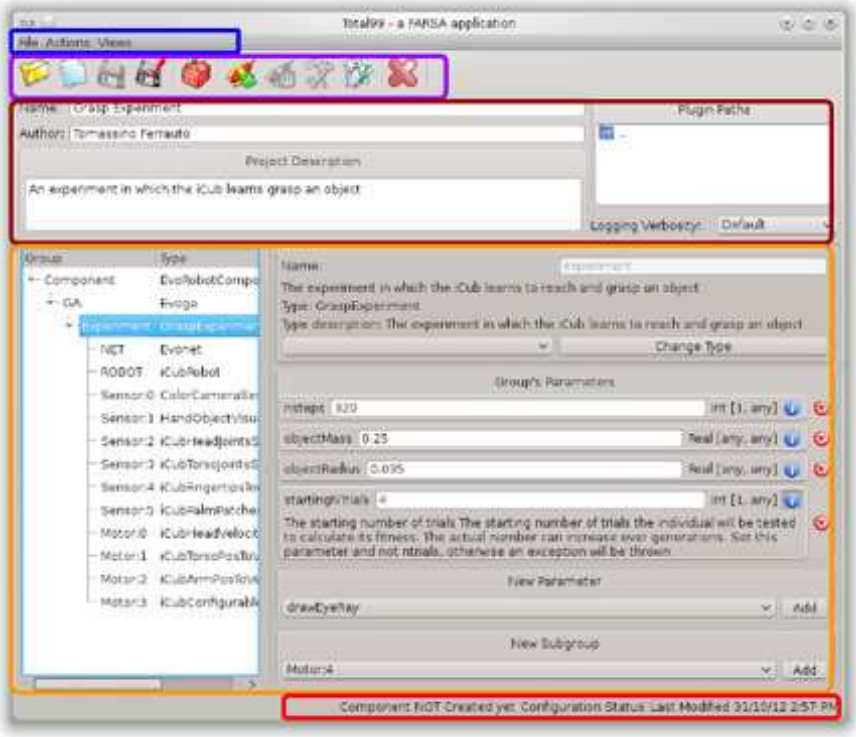


Figure 4.5: The *total99* graphical interface. The menu bar (blue), the toolbar (magenta), the project information bar (brown), the project parameters widget (orange), and the status bar (red) have been highlighted with coloured rectangles.

can be done by loading or creating a configuration file (through the use of the commands available in the File menu) and by setting the configuration components and parameters through the parameters widget (see the orange rectangle in Figure 4.5).

More specifically, the left part of the parameters widget is used to display the hierarchical organization of the components and the right part is used to display the parameters of the currently selected component and/or to add or remove sub-components and parameters (these can be selected from automatically generated lists that include only the parameters that belong to the current component and the subcomponents that can be instantiated from the current component).

Once the configuration file has been set up, the user can run the project through the menu or the tool bar. As we mentioned above, this initiates the loading of the selected plugins, the instantiation of the software components specified in the configuration file,

and the configuration of the components on the basis of the parameters specified in the configuration file. At this point, the commands associated to the components that have been instantiated and the associated graphic widgets can be executed from the Action and Views folders of the menu bar.

4.6 Illustrative experiments

In this section we briefly illustrate the available illustrative experiments. These examples and the associated documentation enable user to familiarize with the tool. Moreover they can be used as starting point for creating new experiments.

4.6.1 Braitenberg Vehicles

Braintenberg vehicles consist of a series of minimal embodied and situated agents of increasing complexity described by Valentino Braitenberg in his very influential book [12]. These vehicles are provided with few sensors and motors and with minimal brains realized by simply connecting sensors and motors through wires. Braitenberg vehicles were meant to be through experiments. However, some of them can be easily implemented in physical robots.

The *BraitenbergExperiment* plugin enables to experiment with Braitenberg vehicles 2 and 3. In particular, thanks to the integrated graphical interface, it enables the user to vary the wiring circuit, the conductivity of the wires, and to immediately observe the resulting behaviour. The documentation included in [81] (chapter 1) includes a brief overview of Braitenberg work, an explanation of vehicles 2 and 3, and directions on how to use this experiments to gain a practical knowledge on how behaviour emerges from the robot/environmental interaction.

4.6.2 The Discrimination Experiment

The *KheperaDiscriminationExperiment* plugin enables the user to replicate one of the first evolutionary robotics experiments that were carried out in the world [76]. This experiment still represents one of the most straightforward demonstrations demonstration of how adaptive robots that develop their skills autonomously in interaction with the

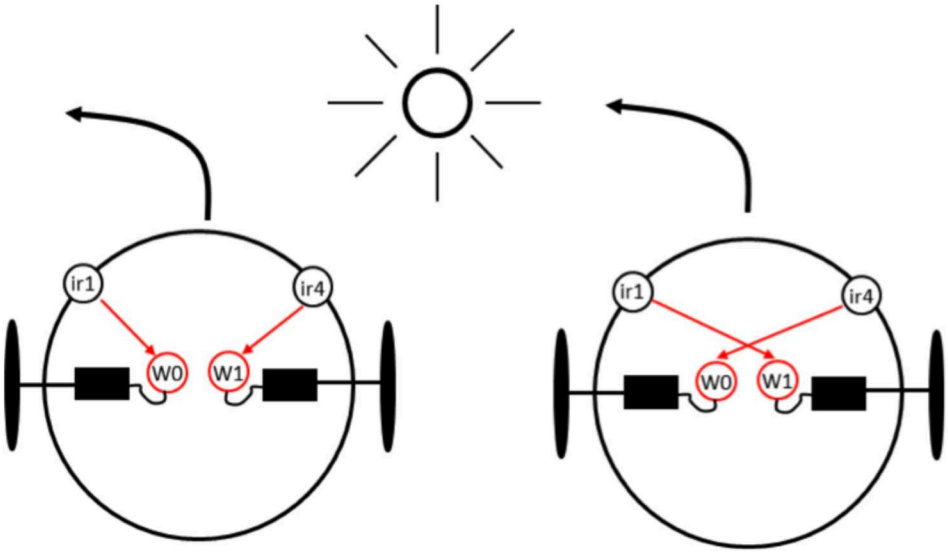


Figure 4.6: Schematic illustration of the Braitenberg vehicles used in the *BraitenbergExperiment* plugin

environment can discover simple and effective solutions that can be hard to imagine for a human designer. In conjunction with the documentation included in [81] (chapter 2-3), this experiment constitutes a perfect first example to gain practical knowledge on Evolutionary Robotics.

The experiment involves a Khepera robot provided with a neural network controller with six sensory neurons (that encode the state of the six frontal infrared sensors) directly connected to two motor neurons (that encode the desired speed of the robot's wheels) located in an arena surrounded by walls and containing a cylindrical object. The task of the robot is that to find and remain close to the cylinder. A video of the evolved behaviour is available at the following webpage: `http://laral.istc.cnr.it/res/emerg-b/`.

4.6.3 Reaching and Grasping on a iCub humanoid robot

The *GraspExperiment* plugin enables a simulated iCub robot to acquire integrated reaching and grasping capabilities that enable it to reach a ball located in varying positions over a table, grasp it, handle it, and lift it [63]. Beside the difficulties concerning the

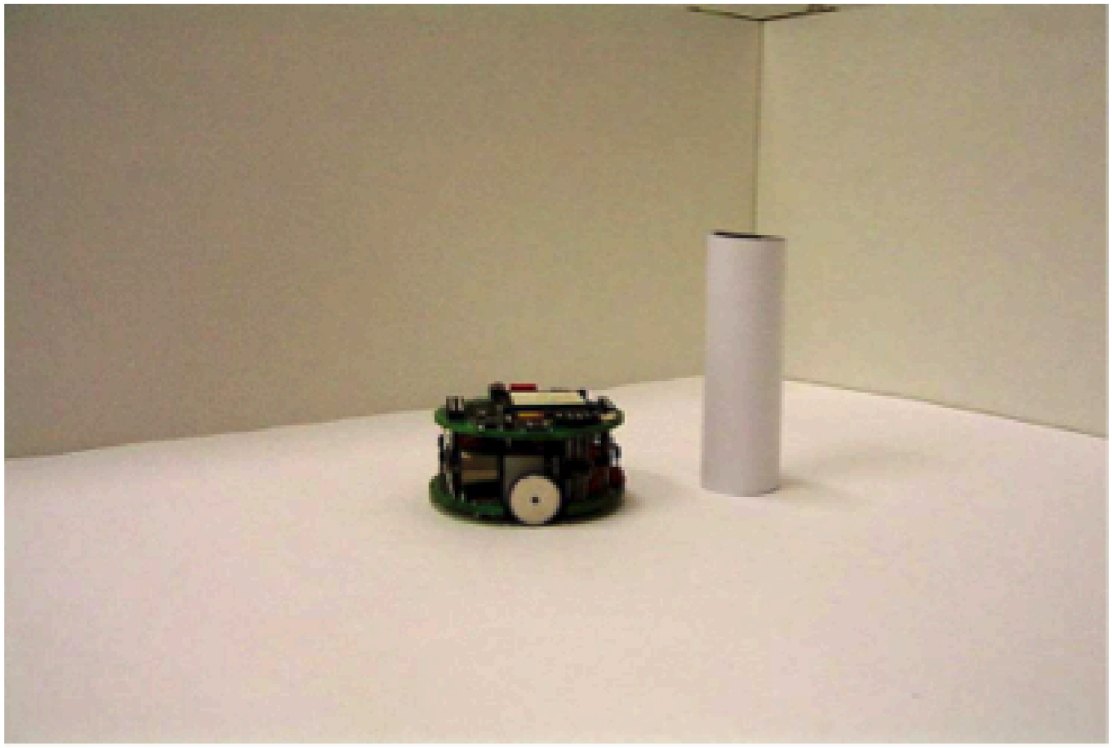


Figure 4.7: Picture of a real Khepera robot performing the same task implemented in the *KheperaDiscriminationExperiment* plugin

need to control an articulated robot with many DOFs, this represents a rather challenging task since it requires interaction with physical objects (including a sphere that can easily roll out of the robot's peripersonal space) and integration of three interdependent behaviours (reaching, grasping, and lifting). This indeed represents one of the most complex tasks that have been successfully mastered in evolutionary robotics and more generally in adaptive methods. The results obtained in simulation have not been validated in hardware. However, a video showing the behaviour of a similar experiment in which the evolved controller has been successfully ported on the physical robot is available at the following webpage: `http://laral.istc.cnr.it/res/lang-dev/`.

The *ReachExperiment* plugin, instead, enables to replicate a simpler experiment in which the robot is evolved simply for the ability to reach variable target point in the robot's peri-personal space.

The source code of these plugins can eventually be modified to replicate other related

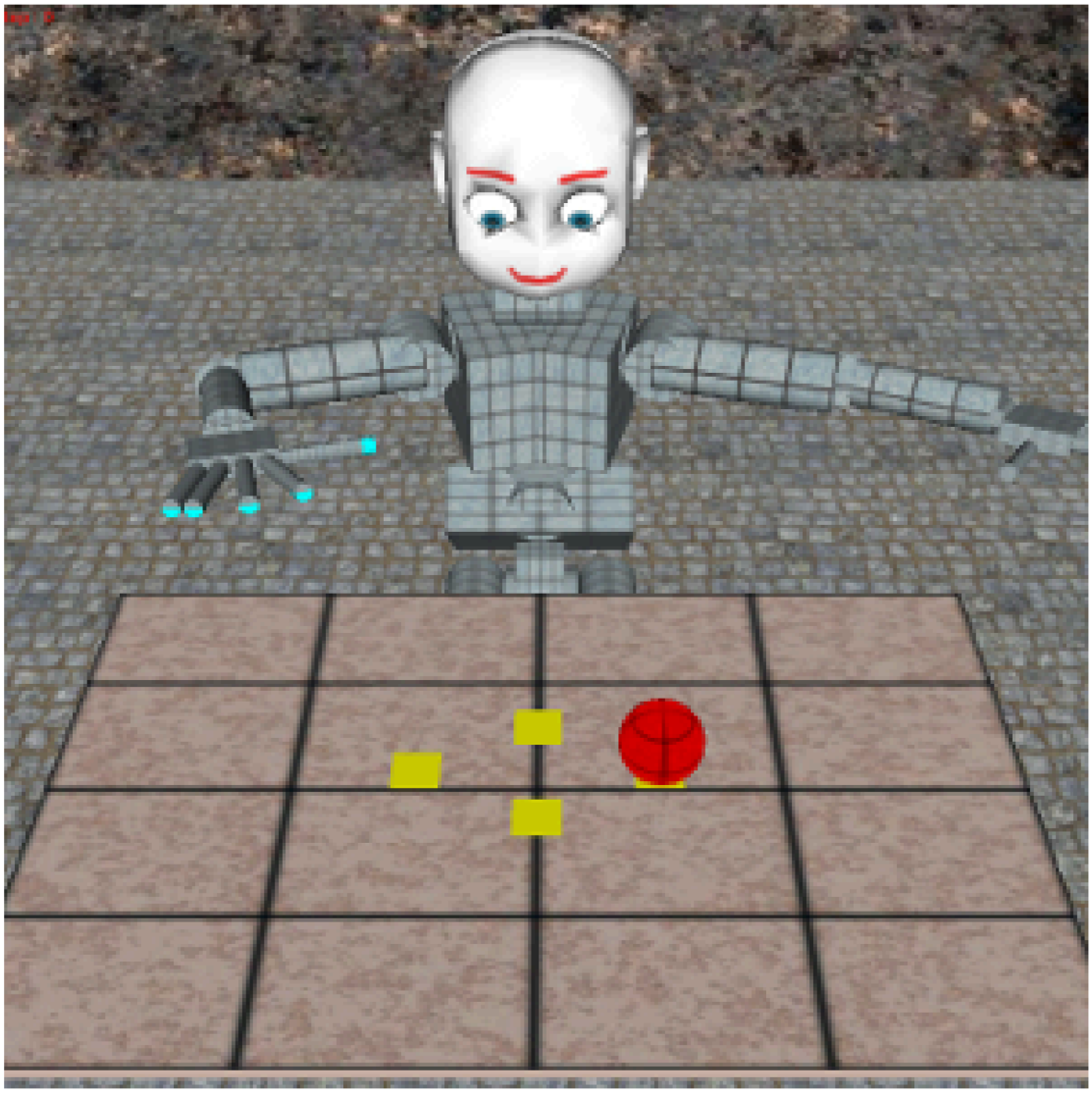


Figure 4.8: Screenshot from the *GraspExperiment* plugin

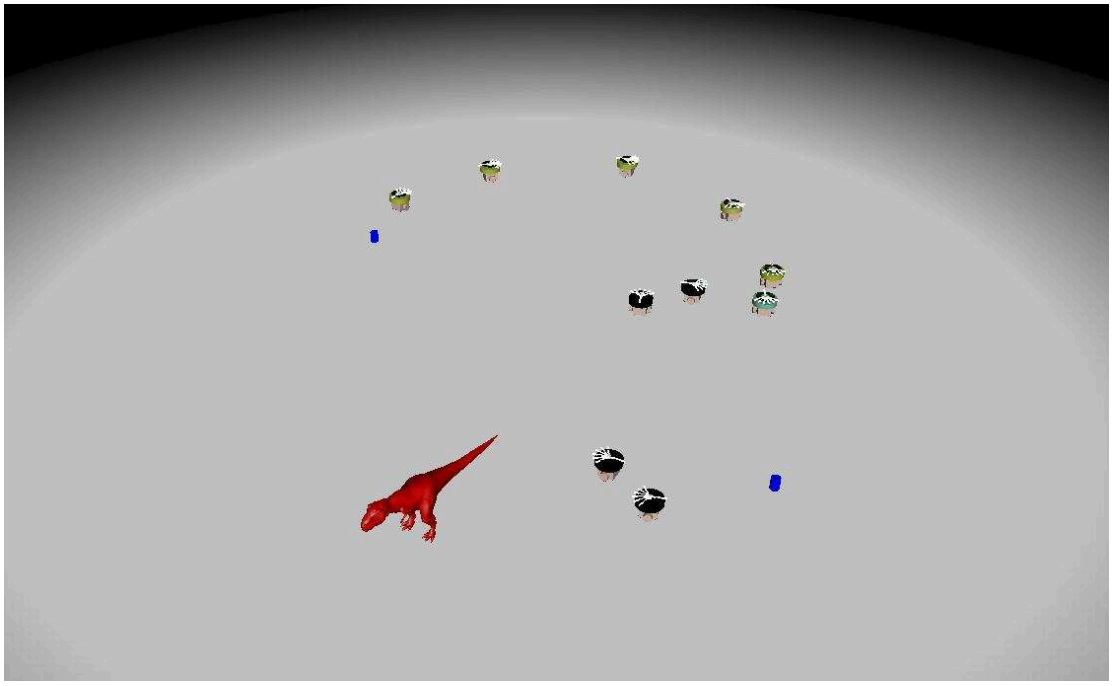


Figure 4.9: Screenshot from the *CollectiveForagingExperiment* plugin

experiments involving the iCub platform, such the experiments on active categorization described in [123], language and action described in [64], and language comprehension described in [125].

4.6.4 Collective Behaviour and Swarm Robotics

FARSA allows to carry on both individual and collective experiments, i.e. experiments in which robots are situated in an environment containing other robots. The term "swarm" is generally used to indicate experiments involving a large number of robots.

Setting up collective experiments in FARSA is extremely easy. For experiments based on the EvoRobotExperiment class, it only requires to set the parameters that specify the number of robots needed.

The *CollectiveForagingExperiment* plugin, for example, enables you to carry on experiments involving a population of 10 MarXbots that can be evolved for the ability collect "food" and to bringing it back to the "nest" (which are indicated by the two blue cylinders in Figure 4.9). The robots might cooperate to achieve better performance with

respect to robots that operate individually. In particular the robots might coordinate to collectively explore the environment and to overcome the limitation of their individual sensory systems (i.e. the fact that they are able to perceive the food area and the nest area only up to a limited range). Indeed, the robots evolved for the ability to reach the food area and to bring the collected food to the nest area tend to form a dynamic chain between the two cylinders that enables them both to preserve information concerning the relative position of the two target destinations and to travel directly back and forth toward distant locations that are often too far to be perceived (see [110]).

The plugin also allows to carry on more complex experiments in which the foraging robots also need to coordinate to help stuck robots and to escape predators (indicated in red in Figure 4.9).

In this experiment the colonies of robots are formed by fully-related genetic individuals (i.e. by 10 clones of the same individual genotype that give rise to 10 identical robots with 10 identical neural controllers).

4.6.5 Sensory-Motor Coordination

Sensory-motor coordination refers to the ability to act so to later perceive useful information. By coordinating their perceptual and action capabilities, robots can access and generate the information they need to carry on their task and can solve problems through solutions that are significantly more parsimonious with respect to solutions that do not exploit sensory-motor coordination.

The *KepheraDiscriminationExperiment* plugin described in section 4.6.2 represents a demonstration of how a problem that apparently requires to discriminate between walls and cylinders can be solved through a simple solution that does not require to categorize the two type of objects. The possibility to identify this simple solution is crucial to solve the problem, since the stimuli perceived near the two types of objects largely overlap in the robot's sensory space [78, 79].

The *AbstractDiscriminationExperiment* plugin allows to replicate a simplified version of

the experiment reviewed above in which an agent that is situated in a circular environment and that can move only clockwise or counter-clockwise needs to reach and remain in the left side of the environment. The environment is constructed so that each of the 20 different stimuli that the robot can perceive are present both on the left and on the right portion of the environment. Consequently, the perception of any stimulus by itself does not provide any information on whether the robot is located on the left or right side. Despite of that, the agent can solve the task on the basis of a reactive controller that does not have the possibility to remember previously experienced stimuli [78, 79].

Finally, the *KheperaNavigationExperiment* plugin allows to replicate an experiment that shows how a robot can coordinate its sensory-motor activity as to generate and use information that it cannot perceive from the environment. In this case, a Khepera robot placed in a rectangular environments in which the length of the top and bottom walls exceed the length of the left and right wall is asked to navigate and remain in the top-left or bottom-right corners of the environment while avoiding the other two corners. The discrimination of the target corners with respect to the other two corners can be carried out by discriminating the length of the walls which however cannot be perceived by the robot on the basis of the available infrared sensors. The experiment shows how, by coordinating the sensory and motor process, the robot can solve the problem through a simple strategy that consists in reaching a corner and then leaving it with an angle of about 45 degrees from the two walls forming the corner. This in fact ensures that by turning left and then following the wall, when the robot encounter a wall on its left side, it will always navigate toward the two right corners. In other words the strategy of abandoning a corner with a 45 degrees angle enables the robot to infers whether the walls encountered later on are long or short (walls encountered on the left side are long while walls encountered on the right side are short). Indeed, turning left and then moving forward along walls encountered on the left side, enables the robot to always travel toward the top-left or right-bottom corners [78, 79].

4.6.6 Body Evolution and Morphological Computing

The term “Morphological Computing” refers to the fact that the computations/elaborations that allow a situated robot to display a certain desired behaviour should not be performed necessarily by the robot’s control system only, but can be performed also by the robot’s body [90]. One paradigmatic demonstration of this is constituted by passive walking machines, i.e. robots that can display a bipedal walking behaviour on an inclined plane without any actuator and any control system. Whether these machines manage to display an appropriate walking behaviour or not depends on the physical characteristics of their body (e.g. the length and the mass of each body segment).

The term “Body Evolution” or “Body/Brain Evolution” refers instead to evolutionary robotics experiments in which not only the characteristics of the robots’ controllers but also the characteristics of the robots’ body are evolved.

The *PassiveWalkerExperiment* plugin provides a way to study the power of morphological computation through an evolutionary approach that is used to discover the characteristics of the robots’ body that, in interaction with the environment, enable the robot to display the desired walking behaviour.

In this experiment a biped robot constituted of only passive elements (rigid body segments, joints and springs) can evolve an ability to walk on an inclined plane. For simplicity, the plugin implements a simplified biped with a body that does not extend over the lateral axis and consequently does not need to balance over that axis. The physical characteristics of the robot body are encoded in the robots’ genotype and evolved. Evolving individuals are evaluated on the basis of the distance walked during a fixed amount of time. The implementation of the body structure is included in the plugin code and constitutes a useful exemplification of how robots made of articulated body parts can be implemented in FARSA.

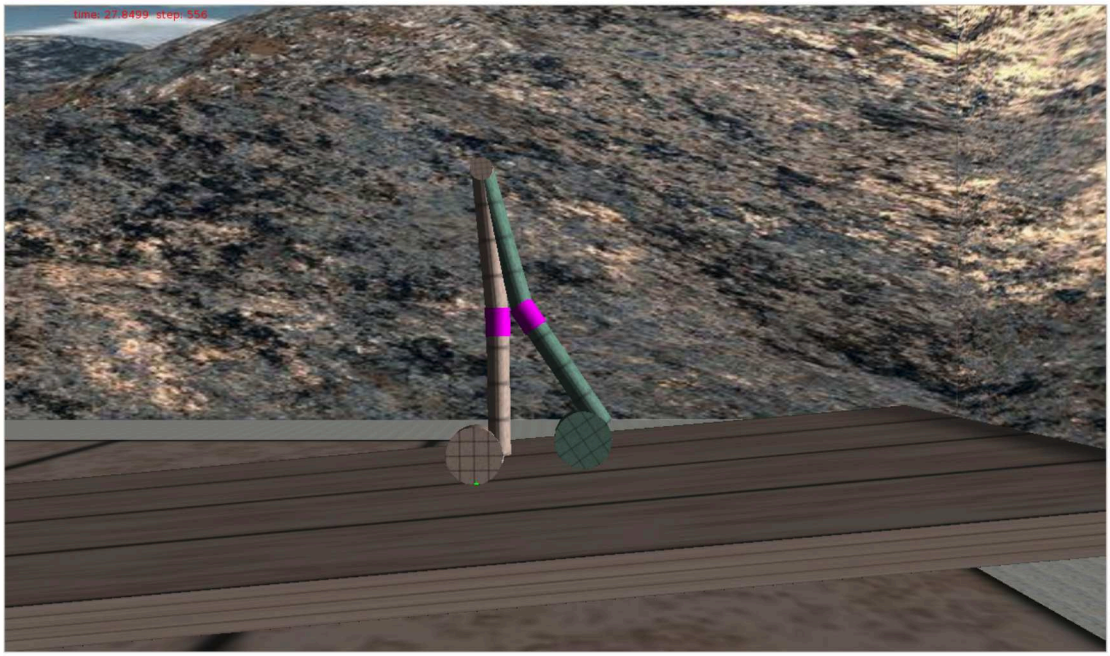


Figure 4.10: Screenshot from the *PassiveWalkerExperiment* plugin

4.6.7 Minimal Cognitive Behaviours

The term Minimal Cognitive Behaviour has been introduced by Randall Beer in [9] to describe experiments that enable the study of evolution of cognitive capabilities, such as categorization or attention, in experimental settings which are as simple as possible. The simplicity of the agent and of the environment, in fact, can enable the experimenter to carry on detailed analysis of the agent/environmental dynamics that can be important to elucidate the mechanisms that are at the basis of the studied cognitive capacity.

In the research carried out by Randall Beer and collaborators, this has been typically realized using agents which can move only left or right along a single dimension provided with a simple "eye" realized by using seven sensors that can measure the relative distance of simple objects falling down from the top with varying direction and speed (see Figure 4.11). The agents' neural controllers include seven corresponding sensory neurons, a variable number of internal neurons with recurrent connection, and two motor neurons that control the direction and the speed of the agents' movement

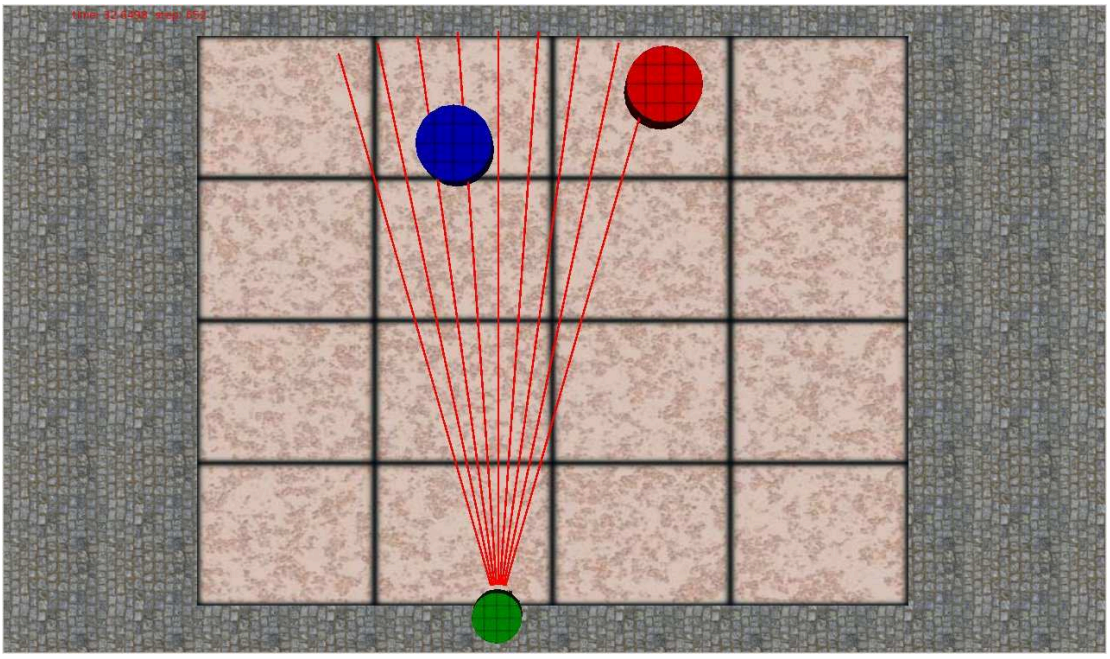


Figure 4.11: Screenshot from the *MinimalCognitiveBehaviourExperiment* plugin

along the horizontal axis. This type of simple experimental setting can be used to study different cognitive capacities. For example, experiments involving circle-shaped and diamond-shaped falling objects, in which the agent should "catch" (i.e. collide with) the former but not the latter type of objects, can be used to study active categorization [7]. Experiments involving two circular objects falling down at different time and locations, in which the agent should catch the first and the second landing object in sequence, can be used to study selective attention [44].

The *MinimalCognitiveBehaviourExperiment* plugin can be used to replicate the experiments on selective attention described in [44] and can be used as a basis for implementing other related experiments.

4.6.8 Learning by demonstration

The *KinestheticGraspExperiment* plugin implements an experiment in which an iCub robot learns to reach and grasp an object placed on a table [126]. Unlike the *GraspExperiment* plugin (described in section 4.6.3), in which the robot is trained using a genetic algorithm, in this case both a supervised learning and a genetic algorithm are

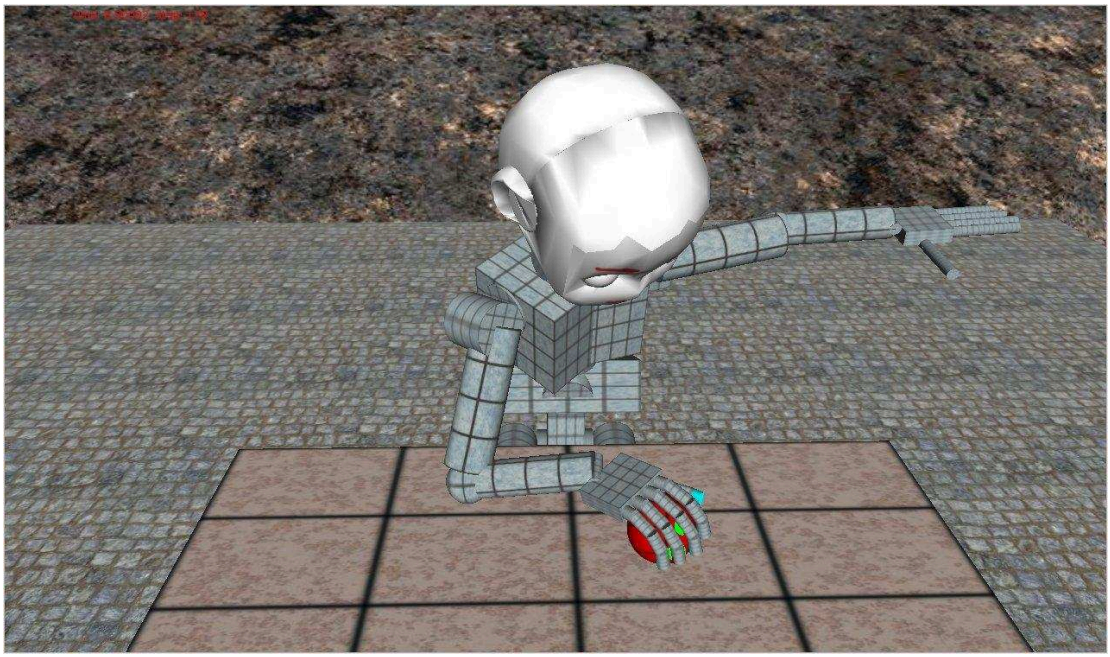


Figure 4.12: Screenshot from the *KinestheticGraspExperiment* plugin

used. More specifically, the robot first acquires the ability to perform approximated reach and grasp through a *learning by demonstration* procedure and then its skills are refined by the application of evolutionary robotics techniques.

The iCub robot is placed in front of a table with a red ball on top of it, which can assume random positions. The learning procedure is made up of two phases. In the first one the robot arm is moved by external forces to reach and grasp the ball, as if a “teacher” was guiding it. During this movement, the sequence of postures is recorded and used to train the neural controller of the robot with the *Levenberg-Marquardt* algorithm (similar to the classical *back-propagation* algorithm, but with a much faster convergence time). Following this initial training, the robot undergoes an evolutionary process to fine-tune the parameters of the controller, in order to perform effective movements.

4.7 Customizing and Expanding FARSA

FARSA can be expanded by creating new experimental plugins that can then be shared together with the existing illustrative experiments. Moreover, all the functionality of FARSA can be extended and expanded. We provide a detailed description of how this

can be done in FARSA documentation available from `https://sourceforge.net/p/farsa/wiki/Home/`. In appendix A we provide a more synthetic description that illustrates to interested readers how this can be realized.

4.8 Conclusions

In this chapter we have introduced FARSA, an open source tool targeted also toward user with limited technical capabilities that enables to carry on experiments involving embodied agents.

As far as we know FARSA is the only available tool that provides an integrated framework for carrying on experiments of this type, i.e. it is the only tool that provides ready to use integrated components that enable to define the characteristics of the robots and of the environment, the characteristics of the robots' controller and the characteristics of the adaptive process. This enables users to quickly setup complex experiments and to quickly start collecting results.

The tool still requires simple programming skills for implementing custom base rewarding functions or specific environmental structure. However, the level of technical skills required is significantly smaller with respect to alternative tools.

In its current form, the tool would allow creating experiments similar to those described in chapter 2 and 3 at a fraction of the effort. First of all there are ready-to-use robotic models and it is possible to create new ones with simple building blocks (boxes, cylinders and actuated joints). The same building blocks can also be used to build the environment in which the robots acts and it is possible to decide how accurately the interaction between objects should be simulated. This allows to trade-off accuracy for simulation speed, e.g. to quickly test multiple hypotheses first so that more time can be spent on the most promising ones. Moreover in FARSA the simulation of the robot and the environment is already integrated with the other tools that are required to perform simulated cognitive science experiments like the ones described in this thesis (i.e. a neural network library and a genetic algorithm library).

More in general FARSA contains some of the building blocks that are generally needed to create an embodied cognitive science experiment: a simulator for the body-environment interaction, a library of agent controllers and adaptation algorithms. Concerning experiments investigating the interaction between action and language, FARSA has few modules that are explicitly dedicated to communication (e.g. some sensors), but implementing them is quite straightforward using mechanisms like e.g. that of *resources*. Moreover multi-agent scenarios in which two or more robots share the same environment and communicate among them are already possible, as shown by some of the illustrative experiments described in this chapter.

FARSA, however, is far from being perfect. The main problem is that, due to use of old pieces of code that were developed in the past in the LARAL laboratory, some portions of the code are not completely modular. For example there is one C++ class that is used as the starting point to implement evolutionary robotics experiments. It puts together the simulator, neural network, sensors/motors and the genetic algorithm, but it does not allow to selectively replace one component (e.g. the genetic algorithm) with another one. In case of experiments that do not perfectly fit in this scenario, this means that the user can either use the class, accepting that there will be parts of the code that are not used actively but might consume resources (e.g. memory), or not use it at the cost of writing more code from scratch (e.g. to connect sensors and motors to the neural network).

Another issue with FARSA is that it only has limited tools to support analysis of agents' behaviour. The aim is not to be able to perform do all needed analysis directly inside FARSA, there are many tools that can be used to this end. It would however be very useful, for example, to inspect and plot some simulation variables while watching the agents' behaviour. This is possible, at the moment, only for some variables, like e.g. neurons of the neural networks controlling the agent.

It would also be beneficial to work on increasing the parallelism of simulations. At the moment, multiple agents can be simulated in parallel, but only if they do not inter-

act (e.g. when evaluating the individuals of a population of one generation of a genetic algorithm). To exploit the new highly parallel hardware architectures (like GPUs), parallelization at a finer level would be needed, like e.g. parallel activation of the controller of one agent or, in the case of swarm robotics simulations, parallel simulation of different agents of the swarm. There have been attempts at using *OpenCL*[36], a framework for parallel programming of heterogeneous systems, but they are still at a very preliminary stage.

Another limitation of the current version of the tool is that it is not possible to interface with real robots. A step that, when possible, is generally useful when working with simulations, is to validate the results on a real robot. This allows to make sure that the simulation did not introduce simplifications that fundamentally change how the agent interacts with the environment. At present, simulations performed with FARSA require a substantial amount of work to be tested on a real robot and the precise steps crucially depend on the robotic platform. As discussed in section 4.2 there are different robotic middlewares that already allow to abstract over the details of the particular robotic platform. By using one of those frameworks it would be possible to easily switch between the simulated model of the robot and the real one.

To conclude, the success of the tool will depend on whether the user community will reach a critical mass that will enable a progressive expansion of the functionalities provided by the tool and a progressive expansion of the experiments repository. For this reason in the near future we plan to disseminate the tool toward the large number of potentially interested users who include students and researchers in the embodied cognitive science community and professors of undergraduate and graduate courses.

We hope that the tool can help to reduce the complexity barrier that currently discourages part of the researchers interested in the study of behaviour and cognition from initiating experimental activity in this area.

[36] `https://www.khronos.org/opencl/`

Chapter 5

Conclusions

Embodied cognitive science addresses the study of behaviour and cognition in simulated or real agents that have a body and that are situated in an external environment with which they interact. In part of the cases, these studies also investigate how these agents can develop their skills autonomously through an evolutionary and/or learning process.

For many years, these studies have been confined to relatively simple agents and tasks, due to theoretical as well as technical limitations. However, recent research, including my own, have demonstrated how this method can be extended to studies that involve agents with complex morphologies and rich sensory-motor systems mastering relatively hard tasks ([6, 64, 95, 97, 101, 124, 137]).

In this thesis I reported two series of experiments in which we investigated the relation between language and action development. More specifically in chapter 2 I reported a series of experiments in which we demonstrated how the exposure to linguistic inputs, that indicate the action that need to be performed in a particular phase, facilitates the development of object manipulation skills. Moreover in chapter 3, I reported a series of experiments in which we demonstrated how the acquisition of an ability to comprehend simple command sentences lead to the acquisition of compositional comprehension and action skills that enable the robots to comprehend and execute appropriately also new sentences never experienced during the training process.

More specifically, the experiments reported in chapter 2 showed how the presence of linguistic labels enabled the robot to successfully developed effective reaching, grasp-

ing, and lifting behaviour and to develop an ability to correctly handle the transition among them. The presence of language helped the robot to segment the action in the three constituent parts (reach-grasp-lift) and to properly handle the transition from the second to the third behaviour.

The experiments reported in chapter 3 demonstrated how the development of an ability to respond to linguistic commands, formed by the combination of action and object "words", by producing the appropriate corresponding manipulation behaviour led to the development of solutions characterized by a modular organization that enables the robot to comprehend new sentences (never experienced before) by producing the appropriate corresponding actions. The fact that this type of compositional organization emerged only in the experimental condition in which the actions to be produced were related, demonstrates how the similarity relationships between actions influence the synthesis of compositional solutions.

The experiments presented are very simplified in certain respects. The form of language used is extremely simple and the behavioural repertoire acquired by the robots is also rather limited. The adaptive process is realized by using one of the most simple and yet effective techniques that can be used to synthesize behaviours of this type, that is a basic genetic algorithm. On the other hand, the usage of a rather sophisticated humanoid robot enables us to study adaptive tasks that have a significant complexity. The realization of experiments that are complex in that respect was functional to address our scientific objective. Indeed, the facilitation effect of language on action development can only be observed in situations in which the behaviours to be developed are sufficiently complex. Similarly, the emergence of compositional structures in action generation and language comprehension might crucially depend on the need to perform behaviours displaying a hierarchical organization (i.e. behaviour with a certain complexity).

To perform these two studies the development of a new software tool was needed. Each experiment had different requirements, that were analysed in the final sections

of chapters 2 and 3. These requirements guided the development of the tool that would then become FARSA, presented in chapter 4. In its current form FARSA allows to easily set up and carry on embodied experiments and has already been used to perform other experiments not reported in this thesis.

Today FARSA is the only available tool that provide an integrated framework for carrying out experiments with embodied and situated robots, i.e. it is the only tool that provides ready to use integrated components that enables to define the characteristics of the robots and of the environment, the characteristics of the robots' controller, and the characteristics of the adaptive process. This enables users to quickly setup complex experiments and to quickly start to collect results. We hope that this open-source tool can help to reduce the complexity barrier that currently discourages some researchers interested in the study of behaviour and cognition from initiating experimental activity in this area.

5.1 Contribution to knowledge

This thesis contains both scientific and methodological contributions. Scientific contributions to knowledge are contained in chapters 2 and 3 and are as follows:

- the experiments in chapter 2 have shown that external guidance in the form of simplified linguistic labels can help a robot to develop manipulation abilities that would otherwise be too complex to learn. This result is in line with theories about the cognitive role of language, as discussed for example in [70];
- the experiments in chapter 3 show another possible effect of language on the development of behavioural capabilities. The use of a simple compositional linguistic instruction made up of an action "word" plus an object "word", in fact, leads to the development of a modular organization of behaviours that allows the robot to respond to instructions never experienced before;
- with regard to the *symbol grounding problem* discussed in section 1.2, the cited experiments are examples of systems in which symbols are fully grounded in the

sensory-motor capabilities of the agents. This is a more profound form of grounding with respect to what has been proposed in related literature (e.g. [112]).

Chapter 4, as well as sections 2.6 and 3.9, contain methodological contributions:

- researchers in embodied cognitive science need powerful software tools. There are many available libraries, but most of them only can only be used to deal with one particular aspect of an experiment (e.g. physics simulation, control, adaptation algorithms). The FARSA framework, instead, contains an integrated set of components that can be used to quickly setup an embodied cognitive science experiment;
- such an integrated tool also facilitates the sharing of source code among researchers and the possibility to replicate experiments.

Appendix A

Customizing and Expanding FARSA

A.1 Plugins, components and resources

FARSA can be extended by creating new components, i.e. software structures belonging to a certain type (e.g. robots, sensors, motors, experiments, controllers etc.) that can be instantiated at runtime when needed. The components that are required can be specified in a configuration file and can be configured in the same configuration files through appropriate parameters. New components can be conveniently be encapsulated in plugins, that can be compiled independently from FARSA e can be loaded at runtine.

A.1.1 Creating a plugin and registering components

To create a component in a FARSA plugin the user might use the following class:

```
#include "farsaplugin.h"
...
class FARSA_PLUGIN_API ExampleClass : public Component
{
    FARSA_REGISTER_CLASS(Component)
    ...
};
```

The example contains everything that is needed to add a new component. In particular the FARSA_REGISTER_CLASS macro is necessary to instantiate the component from a configuration file. The argument of the macro is the name of the parent component.

The project required to compile the plugin can be generated with CMake[1], a cross-platform, open-source build system. The build steps needed to compile a CMake project are specified in a script file called CMakeLists.txt. FARSA provides ready-to-use CMake files that make the compilation of plugins as easy as possible[2].

A.1.2 Configuring components

Below we show an example of a configuration file that can be used to instantiate and configure a series of components:

```
[Component]
type = EvoRobotComponent
[Component/GA]
type = Evoga
ngenerations = 500
[Component/GA/Experiment]
type = KheperaDiscriminationExperiment
nsteps = 600
[Component/GA/Experiment/ROBOT]
type = Khepera
kinematicRobot = true
[Component/GA/Experiment/NET]
type = Evonet
[Component/GA/Experiment/Sensor:0]
type = KheperaSampledProximityIRSensor
activeSensors = 11111100
[Component/GA/Experiment/Motor:0]
type = KheperaWheelVelocityMotor
```

The text between square brackets is the name of a group. All parameters belong to the group immediately preceding them. Groups are organized in a hierarchical way and the "/" character is used to separate groups and subgroups. So, for example, *Component/GA/Experiment* means that *Experiment* is a subgroup of *GA* that, in turn, is a subgroup of *Component*. The parameter *type* is used to specify the class of the

[1] http://www.cmake.org/

[2] See page https://sourceforge.net/p/farsa/wiki/PluginsAndRegistration/

component.

The following example shows a component implemented inside a plugin:

```
class FARSA_PLUGIN_API KheperaDiscriminationExperiment : public
    EvoRobotExperiment
{
  FARSA_REGISTER_CLASS(EvoRobotExperiment)
public:
  ...
  static void describe(QString type);
  virtual void configure(ConfigurationParameters& params, QString prefix);
  virtual void postConfigureInitialization();
  ...
};
void KheperaDiscriminationExperiment::describe(QString type)
{
  EvoRobotExperiment::describe(type);
  Descriptor d = addTypeDescription(type, "The experiment in which a khepera
     robot has to discriminate between an object in the arena and the arena
    walls");
  d.describeReal("playgroundWidth").def(0.5).limits(0.0, +Infinity).help("
    The width of the playground");
  d.describeReal("playgroundHeight").def(0.5).limits(0.0, +Infinity).help("
    The height of the playground");
  ...
}
void KheperaDiscriminationExperiment::configure(ConfigurationParameters&
    params, QString prefix)
{
  EvoRobotExperiment::configure(params, prefix);
  m_playgroundWidth = ConfigurationHelper::getDouble(params, prefix + "
    playgroundWidth", 0.5);
  m_playgroundHeight = ConfigurationHelper::getDouble(params, prefix + "
    playgroundHeight", 0.5);
  ...
}
```

```
void KheperaDiscriminationExperiment::postConfigureInitialization()
{
  EvoRobotExperiment::postConfigureInitialization();
  ...
}
```

Every component should include a *describe()* and a *configure()* function. The *describe()* function, that is executed before the component is created, is used to describe of all the parameters and subgroups that are needed by the component. The *configure()* function, that is executed after the component is created, is used to configure the component on the basis of the parameters specified in the configuration file. Components might also include a *postConfigureInitialization()* function, that is executed after the previous function, that can be used to initialize the component, when necessary. Components can have associated graphical user interfaces (GUIs) and menu items that are added to the default menu included in total99[3].

A.1.3 Declaring and accessing resources

The resource mechanism allows components to access the data of other components in a simple way. Standard components (e.g. sensors, motors, experiments, etc.) are already enabled to create and access resources. The user has the possibility to create new resources through the *declareResource()* function and to access resources through the *getResource()* function as shown in the example below.

```
MyComponent::MyComponent() : ...
{
    ...
    addUsableResource("arena");
    ...
}
MyComponent::f()
{
    ResourcesLocker locker(this);
```

[3]See page https://sourceforge.net/p/farsa/wiki/ComponentsConfig/

```
    Arena* arena = getResource<Arena>("arena");
    ...
}
```

Notice that before accessing a resource through the *getResource()* function the object should declare that it is going to use it through the *addUsableResource()* function. Moreover, the user should lock the resource before accessing it to avoid problems that can be caused by multiple threads attempting to access to the same resources at the same time. The *resourceChanged()* function is called every time a resource is created, modified, or deleted.

A.2 Creating a new experiment

FARSA provides different classes that can be used to create an experiment, tailored to different kinds of experiments. Here we will show how to create experiments on the basis of the *EvoRobotExperiment* component, which is particularly suitable for evolutionary robotics experiments. This class enables the user to define the large majority of the characteristics of the experiment (i.e. the type of robot, the number of robots, the robot's sensors and motors, the architecture of the robots' neural controller, the characteristics of the adaptive process, the number of trials used for evaluating robots, etc.) through parameters specified in the configuration file. The characteristics of the environment, the initial positions and orientations of the robots, and the fitness function, instead, should be defined inside the experimental plugin.

As a basic example, we include below an extract of the source code from the *BraitenbergExperiment* plugin. The full source code and a sample configuration file are available in the FARSA illustrative experiments package. Moreover a detailed description can be found in the on-line documentation[4]. This example does not include the definition of a fitness function since the experiments do not require to subject the robots to an adaptation process.

```
class FARSA_PLUGIN_API BraitenbergExperiment : public EvoRobotExperiment
```

[4]See page https://sourceforge.net/p/farsa/wiki/CreatingNewExperiment/

```
{
    Q_OBJECT
    FARSA_REGISTER_CLASS(EvoRobotExperiment)
public:
    BraitenbergExperiment();
    virtual void configure(ConfigurationParameters& params, QString prefix);
    static void describe(QString type);
    virtual void postConfigureInitialization();
    virtual void initTrial(int trial);
    virtual void endStep(int step);
};
...
void BraitenbergExperiment::postConfigureInitialization()
{
    EvoRobotExperiment::postConfigureInitialization();
    ResourcesLocker locker(this);
    Arena* arena = getResource<Arena>("arena");
    arena->getPlane()->setColor(Qt::white);
    ...
    Box2DWrapper* e;
    e = arena->createRectangularTargetArea(tickness, playgroundHeight + (
    tickness * 2), Qt::black);
    e->setPosition((playgroundWidth + tickness) / 2, 0.0);
    ...
    RobotOnPlane* robot = getResource<RobotOnPlane>("agent[0]:robot");
}
void BraitenbergExperiment::initTrial(int)
{
    ResourcesLocker locker(this);
    Arena* arena = getResource<Arena>("arena");
    RobotOnPlane* robot = getResource<RobotOnPlane>("agent[0]:robot");
    ...
    robot->setPosition(arena->getPlane(), rx, ry);
    robot->setOrientation(arena->getPlane(), globalRNG->getDouble(-PI_GRECO,
     PI_GRECO));
}
```

```
// This is needed because endStep is pure virtual in EvoRobotExperiment
void BraitenbergExperiment::endStep(int) {}
```

For a description of other experimental classes provided by FARSA see `https://sourceforge.net/p/farsa/wiki/ComponentBaseExperiment/`.

A.3 Customizing the environment

The characteristics of the environment in which the robots are situated and eventually the way in which the environment is re-initialized during different periods of the robots' lifetime are usually specified within the source code of the experimental plugins. In this section we describe the *Arena* component and the *Worldsim* library that can be used to set-up different types of environments.

A.3.1 The Arena Component

The *Arena* component can be used to set-up the environment for experiments involving wheeled robots. It allows the user to easily create a flat planar surface containing objects like walls, cylinders, boxes, target areas (i.e. portion of the floor painted with a specific colour) and light bulbs. See Figure A.1 for an example

The following code briefly illustrates how to create, modify and delete objects in the *Arena*. For more information see `https://sourceforge.net/p/farsa/wiki/ArenaComponent/`.

```
void MyExperiment::setupArena()
{
  ResourcesLocker locker(this);
  Arena* arena = getResource<Arena>("arena");
  // Set the colour of the arena plane
  arena->getPlane()->setColor(Qt::white);
  ...
  // Create a light grey circular target area
  Cylinder2DWrapper* circulararea;
  circulararea = arena->createCircularTargetArea(0.09, QColor
    (200,200,200,255));
```

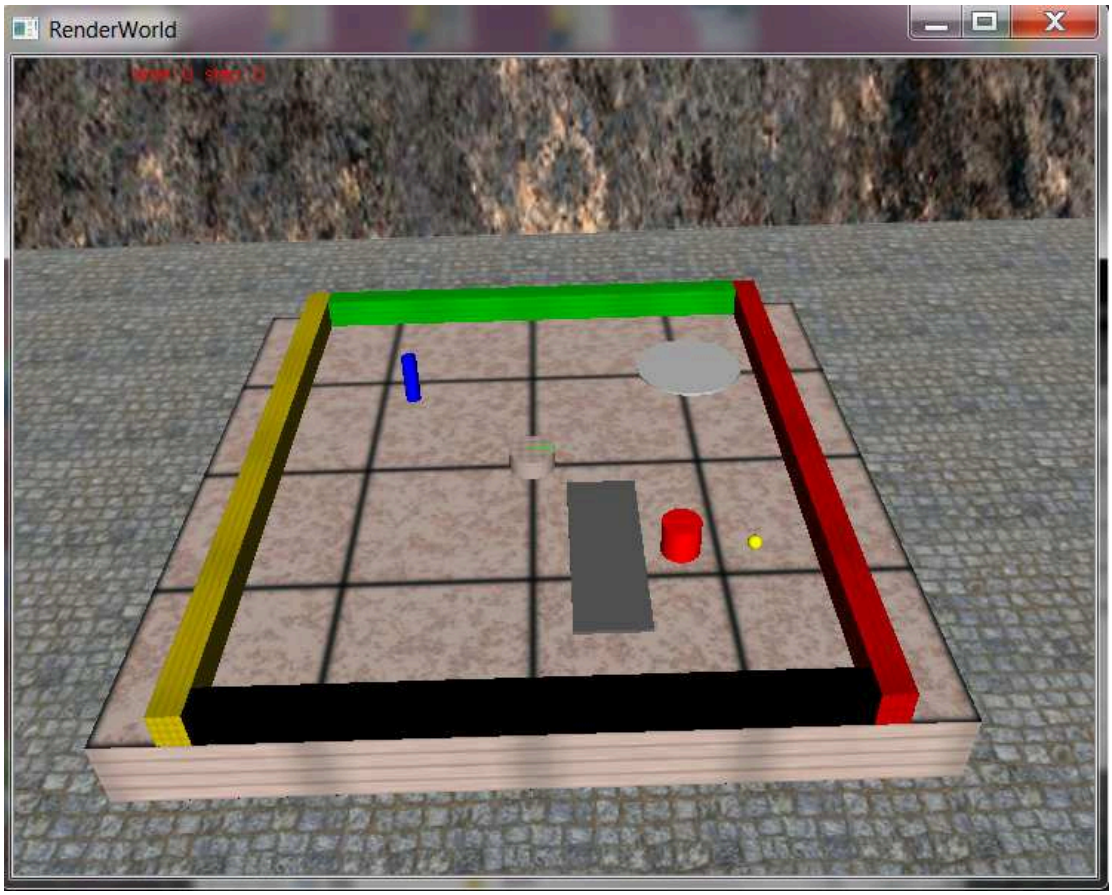


Figure A.1: A simulated *Arena*, with various objects. The cylinder in the centre represents a simulated Khepera robot.

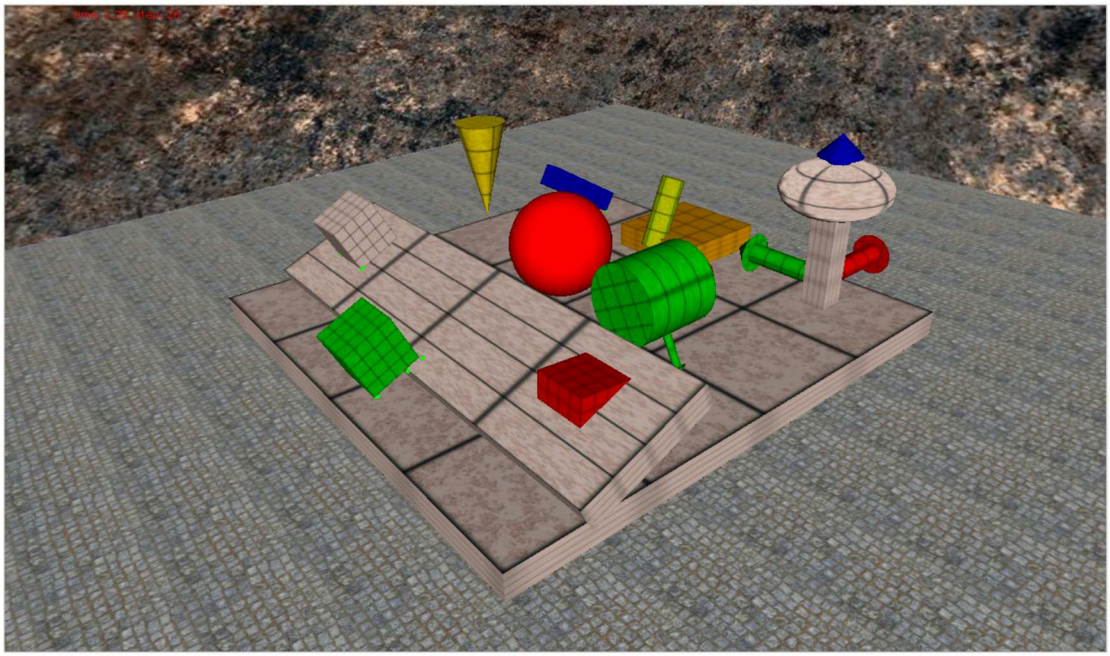


Figure A.2: A simulated world with various objects.

```
  circulararea->setPosition(0.27, -0.27);
  ...
  // Retrieve the position of the circular target area
  wVector pos = circulararea->position();
  ...
  // Delete the circular target area
  arena->delete2DObject(circulararea);
}
```

A.3.2 The Worldsim Library

The *Worldsim* library allows to create objects with different basic shapes (e.g. boxes, spheres, cylinders etc.) that can be combined to create more complex and articulated objects. Below we provide a simple example that shows how to create and modify the objects of the environment. Notice that *wVector* is the class used to represent tridimensional vectors and points, and *wMatrix* is the class used to represent 4x4 rototranslational matrices. For more information see `https://sourceforge.net/p/farsa/wiki/Worldsim/`.

```
// In the following function, m_world is the instance of the World class
void WorldsimShowcase::createShowcase()
{
  ...
  // An hinge joint between a box and a cylinder. The hinge has a lower
  // and upper limit to its movement.
  PhyBox* hBox = new PhyBox(0.4, 0.4, 0.1, m_world);
  hBox->setMass(1.0);
  hBox->setPosition(0.7, 0.0, 0.5);
  hBox->setColor(QColor(255, 190, 0));
  PhyCylinder* hCylinder = new PhyCylinder(0.05, 0.3, m_world);
  hCylinder->setMass(0.1);
  mtr = wMatrix::yaw(M_PI_2);
  mtr.w_pos = wVector(0.5, 0.0, 0.65);
  hCylinder->setMatrix(mtr);
  // Setting a slow initial velocity so that the cylinder does not remain
  // vertical
  hCylinder->setOmega(wVector(0.0, 1.0, 0.0));
  hCylinder->setColor(QColor(190, 255, 0));
  PhyHinge* hinge = new PhyHinge(wVector(0.0, 1.0, 0.0), wVector(-0.2, 0.0,
    0.0), hBox, hCylinder);
  hinge->dofs()[0]->setLimits(-M_PI / 3.0, M_PI / 3.0);
  // Allow free movement (otherwise a motor tries to keep the objects in
    their starting position)
  hinge->dofs()[0]->switchOff();
  hinge->dofs()[0]->enableLimits();
  ...
}
```

A.4 Robotic platforms

FARSA provides a series of ready-to-use robotic platforms (i.e. Khepera, ePuck, MarXbot, and iCub). The robots to be used can be specified through parameters. However the user might need to access to the robots' component, e.g. to set the initial position of the robots and to compute the fitness while the robots move.

For each robot there are two different classes, one with the same name of the robot (e.g. *Khepera*, *Epuck*, etc.) and another with the same name preceded by "*Phy*" (e.g. *PhyKhepera*, *PhyEpuck*, etc.). The *Phy* classes contain the physical model of the robot and all associated physical objects (e.g. joints, motors, sensors). Most of the properties of the robots can be specified through parameters. FARSA also enables to specify through a parameter whether the robots/environmental interaction should be simulated through the kinematic or the dynamic simulation engine.

The source code below shows an example of how a robot resource can be accessed and used to set the position and the orientation of a wheeled robot. *RobotOnPlane* is the base class that can be used to access wheeled robots located over a planar surface (*Arena*). In this situation, in fact, the position of the robot can be set by using simple bi-dimensional coordinates. The example assumes that *MyExperiment* is a subclass of *EvoRobotExperiment*.

```
void MyExperiment::initTrial(int)
{
    ResourcesLocker locker(this);
    Arena* arena = getResource<Arena>("arena");
    RobotOnPlane* robot = getResource<RobotOnPlane>("agent[0]:robot");
    // Setting the position of the robot by selecting a random location
    robot->setPosition(arena->getPlane(), globalRNG->getDouble(-0.2f, 0.2f),
     globalRNG->getDouble(-0.2f, 0.2f));
    // Setting the orientation of the robot to a random value
    robot->setOrientation(arena->getPlane(), globalRNG->getDouble(-PI_GRECO,
     PI_GRECO));
    // Getting the robot position, orientation, radius, and colour
    wVector position = robot->position();
    real orientation = robot->orientation(arena->getPlane());
    real radius = robot->robotRadius();
    QColor color = robot->robotColor();
}
```

The source code shown below instead shows how the user can create a simulated

iCub robot, initializes the posture of the robot, enables the motors of the torso, and access the velocity of one of the *iCub* joint.

```
void iCubShowcase::createiCub()
{
    ResourcesLocker resourceLocker(this);
    // Creating the iCub. Here we set the initial position and do not create
     the controlboards
    // (that are needed only if communication with YARP is required)
    farsa::wMatrix mtr = farsa::wMatrix::identity();
    mtr.w_pos.z = 0.1;
    m_icub = new farsa::PhyiCub(m_world, "icub", mtr, false);
    // Blocking a piece of the iCub torso so that it always remains in the
    same position in the
    // world
    m_icub->blockTorso0(true);
    // Setting a posture. This simply moves all joints to the desired
    positions. It should
    // not be used while iCub joints are moved by the motor controller
    QMap<int, real> jointSetup;
    jointSetup[farsa::PhyiCub::right_shoulder_pitch] = 0.0;
    jointSetup[farsa::PhyiCub::right_shoulder_roll] = 90.0;
    jointSetup[farsa::PhyiCub::right_shoulder_yaw] = 0.0;
    jointSetup[farsa::PhyiCub::right_elbow] = 90.0;
    ...
    m_icub->configurePosture(jointSetup);
    // Enabling torso motors
    m_icub->torsoController()->setEnabled(true);
}
void iCubShowcase::simulationStep(int step)
{
    ResourcesLocker resourceLocker(this);
    // Moving torso. We invert velocity every 200 steps
    if ((step % 200) == 0) {
        m_torso0Vel *= -1.0;
    }
```

```
    m_icub->torsoController()->velocityMove(0, m_torso0Vel);
    m_world->advance();
    // Reading torso position
    double pos;
    m_icub->torsoController()->getEncoder(0, &pos);
    Logger::info(QString("Torso position in degrees: %1").arg(pos));
}
```

A.5 Programming a fitness function

The fitness function of the robot is stored in the *totalFitnessValue* variable. Consequently, to update the fitness function the user needs to update the value of this variable. This can be done after each step, each trial, or at the end of all trials within the *endStep()*, *endTrial()* or *endIndividual()* functions, respectively. To calculate the fitness, typically the user needs to access the robot's component and/or the environment component through the functions that we briefly illustrated above.

Below we include an example taken from the *KheperaDiscriminationExperiment* plugin. In this case, the fitness of the robot is increased by one point, at the end of each step, when the robot is sufficiently near to a certain object in the arena. The fitness is normalized at the end of each trial by the number of steps and then at the end of the evaluation by the number of trials.

```
void KheperaDiscriminationExperiment::initTrial(int trial)
{
  ...
  // Resetting fitness for the current trial
  trialFitnessValue = 0;
}
void KheperaDiscriminationExperiment::endStep(int step)
{
  farsa::ResourcesLocker locker(this);
  farsa::RobotOnPlane* robot = getResource<farsa::RobotOnPlane>("agent[0]:
    robot");
  const farsa::Arena* arena = getResource<farsa::Arena>("arena");
```

```
  // If robot collided with something, stopping the trial
  if (arena->getKinematicRobotCollisionsSet("agent[0]:robot").size() != 0) {
    stopTrial();
    return;
  }
  // Computing the distance of the robot with the object
  const farsa::real distance = robotObjectDistance(robot);
  if (distance < m_distanceThreshold) {
    trialFitnessValue += 1.0;
  }
}
void KheperaDiscriminationExperiment::endTrial(int trial)
{
  totalFitnessValue += trialFitnessValue / getNSteps();
}
void KheperaDiscriminationExperiment::endIndividual(int individual)
{
  totalFitnessValue = totalFitnessValue / getNTrials();
}
```

A.6 Creating custom sensors or motors

FARSA provides many ready-to-use sensors and motors for each supported robotic platform. When necessary, however, the user can create new sensors or motors by defining new subclasses of the *Sensor* and *Motor* classes.

The example below shows how to implement a simple sensor that simply sets three input units to the constant value of 0.5. Sensors and motors classes have a series of methods for doing different things. The *size()* function can be used to specify the number of corresponding sensory units. The *update()* function is used to update the state of the sensory units on the basis of the current position of the robot and on the basis of the state of the environment. See `https://sourceforge.net/p/farsa/wiki/CustomSensorMotor/` for more information.

```
class FARSA_PLUGIN_API MinimalSensor : public Sensor
```

```
{
  FARSA_REGISTER_CLASS(Sensor)
public:
  MinimalSensor(ConfigurationParameters& params, QString prefix);
  ~MinimalSensor();
  virtual void save(ConfigurationParameters& params, QString prefix);
  static void describe(QString type);
  virtual void update();
  virtual int size();
protected:
  virtual void resourceChanged(QString resourceName, ResourceChangeType
    changeType);
  const QString m_neuronsIteratorResource;
  NeuronsIterator* m_neuronsIterator;
};
MinimalSensor::MinimalSensor(ConfigurationParameters& params, QString prefix
    ) :
  Sensor(params, prefix),
  m_neuronsIteratorResource(actualResourceNameForMultirobot(
    ConfigurationHelper::getString(params, prefix + "neuronsIterator", "
    neuronsIterator"))),
  m_neuronsIterator(NULL)
{
  addUsableResource(m_neuronsIteratorResource);
}
void MinimalSensor::update()
{
  ...
  m_neuronsIterator->setCurrentBlock(name());
  for (unsigned int i = 0; i < m_additionalInputs.size(); i++,
    m_neuronsIterator->nextNeuron()) {
    m_neuronsIterator->setInput(0.5);
  }
}
int MinimalSensor::size()
{
```

```
  return 3;
}
void MinimalSensor::resourceChanged(QString resourceName, ResourceChangeType
     changeType)
{
  ...
  if (resourceName == m_neuronsIteratorResource) {
    m_neuronsIterator = getResource<NeuronsIterator>();
    m_neuronsIterator->setCurrentBlock(name());
    for(int i = 0; i < size(); i++, m_neuronsIterator->nextNeuron()) {
      m_neuronsIterator->setGraphicProperties("m" + QString::number(i), 0.0,
     1.0, Qt::red);
    }
  }
  ...
}
```

A.7 Implementing a new robot

The user can also implement a new robotic platform by using the methods available in the Worldsim library. For more information see `https://sourceforge.net/p/farsa/wiki/NewRobot/`.

Bibliography

[1] Michael L. Anderson. Embodied Cognition: A Field Guide. *Artificial Intelligence*, 149(1):91–130, 2003.

[2] H. Arie, T. Endo, S. Jeong, M. Lee, S. Sugano, and J. Tani. Integrative learning between language and action: A neuro-robotics experiment. In *Proceedings of the* 20^{th} *International Conference on Neural Networks*, page In Press. Springer Verlag, Berlin, GE, 2010.

[3] Joshua E. Auerbach and Josh C. Bongard. Environmental Influence on the Evolution of Morphological Complexity in Machines. *PLoS Comput Biol*, 10(1):e1003399+, January 2014.

[4] Ruzena Bajcsy. Active perception. In *Proc IEEE, 76:996–1005*, 1988.

[5] Dana H. Ballard. Animate vision. *Artificial Intelligence*, 48(1):57–86, February 1991.

[6] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, January 2013.

[7] R. Beer. The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11:209–243, 2003.

[8] R. D. Beer and J. C. Gallagher. Evolving dynamic neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[9] Randall D. Beer. Toward the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes, M. Mataric, J. Meyer, J. Pollack, and S. Wilson,

editors, *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 421–429. MIT Press, Cambridge, MA, 1996.

[10] Katrien Beuls and Luc Steels. Agent-Based Models of Strategies for the Emergence and Evolution of Grammatical Agreement. *PLoS ONE*, 8(3):e58960+, March 2013.

[11] M. Bonani, V. Longchamp, S. Magnenat, P. Retornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, and F. Mondada. The marxbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4187–4193, 2010.

[12] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. A Bradford Book, February 1986.

[13] Rodney Allen Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1-2):3–15, June 1990.

[14] Rodney Allen Brooks. Intelligence Without Reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[15] Rodney Allen Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159, January 1991.

[16] A. Cangelosi and D. Parisi, editors. *Simulating the evolution of language*. Springer Verlag, New York, 2002.

[17] A. Cangelosi and T. Riga. An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots. *Cognitive Science*, 30(4):673–689, 2006.

[18] Angelo Cangelosi, Josh Bongard, Martin H. Fischer, and Stefano Nolfi. Embodied intelligence. In Janusz Kacprzyk and Witold Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 697–714. Springer, 2015.

[19] S. Carpin, M. Lewis, Jijun Wang, S. Balakirsky, and C. Scrapper. USARSim: a robot simulator for research and education. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1400–1405, 2007.

[20] Peter Carruthers. The cognitive functions of language. *Behavioral and Brain Sciences*, 25:657–674, 12 2002.

[21] N. Chater and C.D. Manning. Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, 10(7):335–344, 2006.

[22] Hillel J. Chiel and Randall D. Beer. The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*, 20(12):553–557, December 1997.

[23] Andy Clark. *Being There: Putting Brain, Body, and World Together Again*. A Bradford Book, reprint edition, January 1998.

[24] Andy Clark. Magic words: How language augments human computation. *Language and thought: Interdisciplinary themes*, pages 162–183, 1998.

[25] Andy Clark. An embodied cognitive science? *Trends in Cognitive Sciences*, 3(9):345–351, September 1999.

[26] T. H. J. Collet, B. A. MacDonald, and B. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2005.

[27] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science (New York, N.Y.)*, 307(5712):1082–5, February 2005.

[28] Steven H. Collins, Martijn Wisse, and Andy Ruina. A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees. *The International Journal of Robotics Research*, 20(7):607–615, July 2001.

[29] Antonio R. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Harper Perennial, 1 edition, November 1995.

[30] Daniel C. Dennett. *Consciousness Explained*. Penguin, 1991.

[31] E.A. Di Paolo. Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents. *Adaptive Behaviour*, 8(1):27–48, 2000.

[32] J.L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[33] J.L. Elman. Computational approaches to language acquisition. In K. Brown, editor, *Encyclopedia of Language and Linguistics*, volume 2, pages 726–732. Oxford: Elsevier, second edition, 2006.

[34] J. Felip and A. Morales. Robust sensor-based grasp primitive for a three-finger robot hand. In *IEEE/RSJ International. Conference on Intelligent Robots and Systems*, 2009.

[35] Martin H. Fischer and Rolf A. Zwaan. Embodied language: A review of the role of the motor system in language comprehension. *The Quarterly Journal of Experimental Psychology*, 61(6):825–850, May 2008.

[36] P. Fitzpatrick, E. Ceseracciu, D. Domenichelli, A. Paikan, G. Metta, , and L. Natale. A middle way for robotics middleware. *Journal of Software Engineering for Robotics*, 5(2):42–49, 2014.

[37] J.A. Fodor and Z.W. Phylysyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[38] B. Gerkey, R. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics*, pages 317–323, 2003.

[39] Brian P. Gerkey, Richard T. Vaughan, Gaurav S. Sukhatme, Kasper Stoy, Andrew Howard, and Maja J. Mataric. Most valuable player: A robot device server for distributed control, 2001.

[40] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin Harcourt (HMH), Boston, March 1979.

[41] M. Gienger, M. Toussaint, N. Jetchev, A. Bendig, and C. Goerick. Optimization of fluent approach and grasp motions. In *Procceding of the 8th IEEE-RAS International Conference on Humanoid Robots*, pages 111–117. IEEE Press, 2008.

[42] A. Glenberg and M. Kaschak. Grounding language in action. *Psychonomic Bulletin & Review*, 9:558–565, 2002.

[43] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[44] Eldan Goldenberg, Jacob Garcowski, and Randall D. Beer. May we have your attention: Analysis of a selective attention task. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer, editors, *From Animals to Animats 8: Proceedings of the Eighth International Conference on the Simulation of Adaptive Behavior*, pages 49–56. MIT Press, Cambridge, MA, 2004.

[45] Joachim Greeff and Stefano Nolfi. *Evolution of Communication and Language in Embodied Agents*, chapter Evolution of Implicit and Explicit Communication in Mobile Robots, pages 179–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[46] R. V. Guha and Douglas B. Lenat. Cyc: a mid-term report. *AI Mag.*, 11(3):32–59, 1990.

[47] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, June 1990.

[48] Stevan Harnad. To cognize is to categorize: Cognition is categorization. In Claire Lefebvre and Henri Cohen, editors, *Handbook of Categorization*. Elsevier, December 2005. UQaM Summer Institute in Cognitive Sciences on Categorization. 30 June - 11 July 2003 http://www.unites.uqam.ca/sccog/liens/program.html Event Dates: 30 June - 11 July 2003.

[49] S. Harnard. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[50] John Haugeland. *Artificial Intelligence: The Very Idea*. The MIT Press, January 1989.

[51] O. Hauk, I. Johnsrude, and F. Pulvermuller. Somatotopic representation of action words in human motor and premotor cortex. *Neuron*, 41(2):301–307, 2004.

[52] J.L. Elman J.D. Lewis. Learnability and the statistical structure of language: Poverty of stimulus arguments revisited. 2001.

[53] Julio Jerez and Alain Suero. *Newton Game Dynamics*, 2004.

[54] F. Kaplan, P-Y. Oudeyer, and B. Bergen. Computational models in the debate over language learnability. *Infant and Child Development*, 17(1):55–80, 2008.

[55] David Kirsh and Paul Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18(4):513–549, December 1994.

[56] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, 2004.

[57] George Lakoff and Mark Johnson. *Philosophy in the Flesh : The Embodied Mind and Its Challenge to Western Thought*. Basic Books, December 1999.

[58] Tobias Leugger and Stefano Nolfi. Action development and integration in an humanoid icub robot: How language exposure and self-talk facilitate action development. In Terry Bossomaier and Stefano Nolfi, editors, *COGNITIVE 2012, The Fourth International Conference on Advanced Cognitive Technologies and Applications*, pages 24–30. IARIA, 2012.

[59] A. Y. Ng M. Quigley, E. Berger. Stair: Hardware and software architecture. 2007.

[60] B. MacWhinney. Computational models of child language learning: an introduction. *Journal of Child Language*, 37:477–485, 2010.

[61] Gianluca Massera, Angelo Cangelosi, and Stefano Nolfi. Evolution of prehension ability in an anthropomorphic neurorobotic arm. *Front Neurorobot*, 1:4, 2007.

[62] Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta, and Stefano Nolfi. Farsa: An open software tool for embodied cognitive science. Cambridge, MA, 2013. MIT Press.

[63] Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta, and Stefano Nolfi. Designing adaptive humanoid robots through the farsa open-source framework. *Adaptive Behavior*, 22(4):255–265, 2014.

[64] Gianluca Massera, Elio Tuci, Tomassino Ferrauto, and Stefano Nolfi. The facilitatory role of linguistic instructions on developing manipulation skills. *Computational Intelligence Magazine*, 5(3):33–42, August 2010.

[65] M. K. McCarty, R. K. Clifton, D. H. Ashmead, P. Lee, and N. Goulet. How infants use vision for grasping objects. *Child Development*, 72:973–987, 2001.

[66] Tad McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9(2):62–82, April 1990.

[67] G. Metta, P. Fitzpatrick, and L. Natale. Yarp: Yet another robot platform. *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, 3(1):43–48, 2006.

[68] Olivier Michel. Webots™: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[69] Marco Mirolli and Domenico Parisi. Language as an aid to categorization: A neural network model of early language acquisition. pages 97–106, 2005.

[70] Marco Mirolli and Domenico Parisi. Towards a vygotskyan cognitive robotics: The role of language as a cognitive tool. *New Ideas in Psychology*, 29:298–311, 2011.

[71] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *In Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

[72] Francesco Mondada, Edoardo Franzi, and Paolo Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Proceedings of the 3rd International Symposium on Experimental Robotics*, 1993.

[73] Anthony F. Morse, Tony Belpaeme, Angelo Cangelosi, and Linda B. Smith. Thinking with your body: modelling spatial biases in categorization using a real humanoid robot. In *Proceedings of 2010 annual meeting of the Cognitive Science Society. Portland, USA*, pages 1362–1368, 2010.

[74] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

[75] S. Nolfi and M. Mirolli, editors. *Evolution of Communication and Language in Embodied Agents*. Springer Verlag, Berlin, 2010.

[76] Stefano Nolfi. Adaptation as a More Powerful Tool Than Decomposition and Integration. In *Proceedings of the Workshop on Evolutionary Computing and Machine Learning, 13th International Conference on Machine Learning*, 1996.

[77] Stefano Nolfi. Evorobot 1.1 user manual. Technical report, Institute of Psychology, CNR., 2000.

[78] Stefano Nolfi. Power and limits of reactive agents. *Neurocomputing*, 49:119–145, 2002.

[79] Stefano Nolfi. Category formation in self-organizing embodied agents. In Henri CohenClaire Lefebvre, editor, *Handbook of Categorization in Cognitive Science*, pages 869 – 889. Elsevier Science Ltd, Oxford, 2005.

[80] Stefano Nolfi. *New Perspectives on the Origins of Language*, chapter Emergence of communication and language in evolving robots, pages 533—554. John Benjamins Publishing Company, Amsterdam, 2013.

[81] Stefano Nolfi. *Adaptive Robots: Exploring the Complex Adaptive System Nature of Behaviuor and Cognition*. Roma, Italy, 2016.

[82] Stefano Nolfi and Onofrio Gigliotta. Evorobot*. In Stefano Nolfi and Marco Mirolli, editors, *Evolution of Communication and Language in Embodied Agents*, pages 297–301. Springer Berlin Heidelberg, 2010.

[83] Stefano Nolfi and Davide Marocco. Active perception: A sensorimotor account of object categorization. In *Proceedings of the 7th Intl. Conf. on Simulation of Adaptive Behavior*, pages 266–271. MIT Press, 2002.

[84] E. Oztop, N. S. Bradley, and M. A. Arbib. Infant grasp learning: a computational model. *Experimental Brain Research*, 158(4):480–503, 2004.

[85] D. Parisi. *Future Robots: Towards a robotic science of human beings*. John Benjamins Publishing Company, Amsterdam, The Netherlands, 2014.

[86] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini. An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1668–1674, 2010.

[87] Diane Pecher and Rolf A. Zwaan. *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking*. Cambridge University Press, January 2005.

[88] E. Yoshida P.F. Dominey, A. Mallet. Real-time spoken-language programming for cooperative interaction with a humanoid apprentice. *I. J. Humanoid Robotics*, 6(2):147–171, 2009.

[89] F. Warneken P.F. Dominey. The basis of shared intentions in human and robot cognition. *New Ideas in Psychology*, 29(3):260–274, December 2011.

[90] Rolf Pfeifer and Josh Bongard. *How the Body Shapes the Way We Think: A New View of Intelligence*. Computer Science and Intelligent Systems. The MIT Press, Cambridge, MA, USA, October 2006.

[91] Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[92] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[93] F. Reali and M.H. Christansen. Uncovering the richness of the stimulus: structure dependence and indirect statistical evidence. *Cognitive Science*, 29:1007–1028, 2005.

[94] Terry Regier. *The Human Semantic Potential - Spatial Language and Constrained Connectionism*. MIT Press, Cambridge, Mass., 1996.

[95] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE Transactions on*, 6(2):159–168, 2002.

[96] P. Rochat. Self-perception and action in infancy. *Experimental Brain Research*, 123:102–109, 1998.

[97] Matthias Rolf, Jochen J. Steil, and M. Gienger. Goal Babbling permits direct learning of inverse kinematics. *IEEE Trans. Autonomous Mental Development*, 2(3), 2010.

[98] T. G. Sandercock, D. C. Lin, and W. Z. Rymer. Muscle models. In M.A. Arbib, editor, *Handbook of brain theory and neural networks*, pages 711–715. MIT Press, Cambridge, MA, second edition, 2002.

[99] G. Sandini, G. Metta, and D. Vernon. The icub cognitive humanoid robot: An open-system research platform for enactive cognition. In M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer, editors, *50 Years of Artificial Intelligence*, pages 358–369. Springer Verlag, Berlin, GE, 2007.

[100] Giulio Sandini, Giorgio Metta, and David Vernon. Robotcub: an open framework for research in embodied cognition. In *Humanoids*, pages 13–32. IEEE, 2004.

[101] Piero Savastano and Stefano Nolfi. Incremental learning in a 14 dof simulated icub robot: Modeling infant reach/grasp development. In Tony J. Prescott, Nathan F. Lepora, Anna Mura, and Paul F. M. J. Verschure, editors, *Living Machines*, volume 7375 of *Lecture Notes in Computer Science*, pages 250–261. Springer, 2012.

[102] S. Schaal. Arm and hand movement control. In M.A. Arbib, editor, *Handbook of brain theory and neural networks*, pages 110–113. MIT Press, Cambridge, MA, second edition, 2002.

[103] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In Springer verlag, editor, *International Symposium on Robotics Research (ISRR2003)*, pages 1–10, 2004.

[104] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(03):417–424, September 1980.

[105] R. Shadmehr and S. P. Wise. *The Computational Neurobiology of Reaching and Pointing: A Foundation for Motor Learning*. MIT Press, Cambridge, MA, 2005.

[106] Karl Sims. Evolving 3D Morphology and Behavior by Competition. *Artificial Life*, 1(4):353–372, 1994.

[107] Linda B. Smith and Samuelson Larissa K. Objects in space and mind: From reaching to words. In Kelly S. Mix, Linda B. Smith, and Michael Gasser, editors, *Thinking Through Space: Spatial Foundations of Language and Cognition*. Oxford University Press, Oxford, UK, 2010.

[108] Russel Smith. *Open Dynamics Engine*, 2004.

[109] Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised learning of natural languages. *Proc. Natl. Acad. Sci.*, 102:11629–11634, 2005.

[110] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(2):97–119, June 2011.

[111] L. Steels. Experiments on the emergence of human communication. *Trends in Cognitive Sciences*, 10(8):347–349, 2006.

[112] L. Steels. The Symbol Grounding Problem Has Been Solved. So What's Next? In M. de Vega, editor, *Symbols and Embodiment: Debates on Meaning and Cognition*, chapter 12. Oxford University Press, Oxford, 2008.

[113] Luc Steels. Evolving grounded communication for robots. *Trends in Cognitive Sciences*, 7(7):308–312, July 2003.

[114] Luc Steels. The emergence and evolution of linguistic structure: from lexical to grammatical communication systems. *Connection Science*, 17(3-4):213–230, December 2005.

[115] Luc Steels and Tony Belpaeme. Coordinating Perceptually Grounded Categories Through Language: a Case Study for Colour. *Behavioral and Brain Sciences*, 28(04):469–489, 2005.

[116] Luc Steels and Martin Loetzsch. Perspective Alignment in Spatial Language. In Kenny R. Coventry, Thora Tenbrink, and John, editors, *Spatial Language and Dialogue*. Oxford University Press, 2007.

[117] Y. Sugita and J. Tani. Learning semantic combinatoriality from the interaction between linguistic and behvioral processes. *Adaptive Behavior*, 13(1):33–52, 2005.

[118] Y. Sugita and J. Tani. Acquiring a functionally compositional system of goal-directed actions of a simulated agent. In M. Asada, J.C.T. Hallam, J.-A. Meyer, and J. Tani, editors, *Proc. of the* 10^{th} *Int. Conf. on Simulation of Adaptive Behavior (SAB2008)*, pages 331–341. Springer Verlag, 2008.

[119] Esther Thelen and Linda B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action (Cognitive Psychology)*. A Bradford Book, reprint edition, January 1996.

[120] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori. An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 57–61, New York, NY, USA, 2008. ACM.

[121] M. Tomasello. *Constructing a language: a usage-based theory of language acquisition*. Cambridge, MA: Harvard University, 2003.

[122] E. Tuci. An investigation of the evolutionary origin of reciprocal communication using simulated autonomous agents. *Biological Cybernetics*, 101(3):183–199, 2009.

[123] E. Tuci, G. Massera, and S. Nolfi. Active categorical perception in an evolved anthropomorphic robotic arm. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 31–38, Piscataway, NJ, USA, 2009. IEEE Press.

[124] E. Tuci, G. Massera, and S. Nolfi. Active categorical perception of object shapes in a simulated anthropomorphic robotic arm. *Evolutionary Computation, IEEE Transactions on*, 14(6):885–899, Dec 2010.

[125] Elio Tuci, Tomassino Ferrauto, Arne Zeschel, Gianluca Massera, and Stefano Nolfi. An experiment on behavior generalization and the emergence of linguistic compositionality in evolving robots. *IEEE T. Autonomous Mental Development*, 3(2):176–189, 2011.

[126] M. Valenti. Learning of manipulation capabilities in a humanoid robot. Master's thesis, University of Rome, La Sapienza, 2013.

[127] T. Van Gelder. Compositionality: A connectionist variation on a classic theme. *Cognition*, 14:355–384, 1990.

[128] Francisco J. Varela, Evan T. Thompson, and Eleanor Rosch. *The Embodied Mind: Cognitive Science and Human Experience*. The MIT Press, new ed edition, November 1992.

[129] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

[130] C. von Hofsten. Eye-hand coordination in the newborn. *Developmental Psychology*, 18:450–461, 1982.

[131] C. von Hofsten. Developmental changes in the organization of prereaching movements. *Developmental Psychology*, 20:378–388, 1984.

[132] C. von Hofsten. Structuring of early reaching movements: a longitudinal study. *Journal of Motor behavior*, 23:280–292, 1991.

[133] L. S. Vygotsky. *Thought and language*. MIT Press, Cambridge, MA, 1962.

[134] L. S. Vygotsky. *Mind in society*. Harvard University Press, Cambridge, MA, 1978.

[135] J.L. Elman W.C. Morris, G.W. Cottrell. A connectionist simulation of the empirical acquisition of grammatical relations. volume 1778, pages 175–193, Berlin; New York, 2000. Springer-Verlag.

[136] J. Weng. Developmental robotics: Theory and experiments. *I. J. Humanoid Robotics*, 1(2):199–236, 2004.

[137] Yuichi Yamashita and Jun Tani. Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLoS Comput Biol*, 4(11):e1000220+, November 2008.

[138] Xin Yao and M. M. Islam. Evolving artificial neural network ensembles. *IEEE Computational Intelligence Magazine*, 3(1):31–42, 2008.

Bound copies of published papers

The Facilitatory Role of Linguistic Instructions on Developing Manipulation Skills

Gianluca Massera, Elio Tuci, Tomassino Ferrauto, and Stefano Nolfi
Institute of Cognitive Sciences and Technologies (ISTC), ITALY

Abstract–In this paper, we show how a simulated humanoid robot controlled by an artificial neural network can acquire the ability to manipulate spherical objects located over a table by reaching, grasping, and lifting them. The robot controller is developed through an adaptive process in which the free parameters encode the control rules that regulate the fine-grained interaction between the agent and the environment, and the variations of these free parameters are retained or discarded on the basis of their effects at the level of the behavior exhibited by the agent. The robot develops the sensory-motor coordination required to carry out the task in two different conditions; that is, with or without receiving as input a linguistic instruction that specifies the type of behavior to be exhibited during the current phase. The obtained results shown that the linguistic instructions facilitate the development of the required behavioral skills.

I. Introduction

In this paper, we describe a series of experiments in which a simulated iCub robot acquires through an adaptive process the ability to reach, grasp, and lift a spherical object. The robot develops the sensory-motor coordination required to carry out the whole task in two different conditions; that is, with or without receiving as input linguistic instructions that specify the type of behavior that should be exhibited during the current phase. These are binary input vectors associated with elementary behaviors that should be displayed by the robot during the task. The main objective of this study is to investigate whether the use of linguistic instructions facilitates the acquisition of a sequence of complex behaviors. The long term goal of this research is to verify whether the acquisition of elementary skills guided by linguistic instructions provides a scaffolding for more complex behaviors.

Digital Object Identifier 10.1109/MCI.2010.937321

The first theoretical assumption behind this work is that the activity of developing robots displaying complex cognitive and behavioral skills should be carried out by taking into account the empirical findings in psychology and neuroscience which show that there are close links between the mechanisms of action and those of language. As shown in [1], [2], [3], [4], [5] action and language develop in parallel, influence each other, and base themselves on each other. If brought into the world of robotics, the co-development of action and language skills might enable the transfer of properties of action knowledge to linguistic representations, and vice versa, thus enabling the synthesis of robots with complex behavioral and cognitive skills [6], [7].

The second theoretical assumption behind this work is that behavioral and cognitive skills in embodied agents are emergent dynamical properties which have a multi-level and multi-scale organization. Behavioral and cognitive skills arise from a large number of fine-grained[1] interactions occurring among and within the robot body, its control system, and the environment [8]. Handcrafting the mechanisms underpinning these skills may be a hard task. This is due to the inherent difficulty in figuring out from the point of view of an external observer, the detailed characteristics of the agent that, as a result of the interactions between the elementary parts of the agent and of the environment, lead to the exibition of the desired behavior. The synthesis of robots displaying complex behavioral and cognitive skills should instead be obtained through an adaptive process in which the detailed characteristics of the agent are subjected to variation and in which variations are retained or discarded on the basis of their effects at the level of the overall behavior exhibited by the robot situated in the environment [8]. Therefore, the role of the designer should be limited to the specification of the utility function, that determines whether variations should be preserved or discarded, and eventually to the design of the ecological conditions in which the adaptive process takes place [9], [10], [8].

II. Background and Literature Review

The control of arm and hand movements in human and non-human primates and in robots is a fascinating research topic actively investigated within several disciplines including psychology, neuroscience, and robotics. However, the task to model in detail the mechanisms underlying arm and hand movement control in humans and primates and the task of building robots able to display human-like arm/hand movements still represents an extremely challenging goal [11]. Moreover, despite the progress achieved in robotics through the use of traditional control methods [12], the attempt to develop robots with the dexterity and robustness of humans is still a long term goal. These difficulties can be explained by considering the need to take into account the role of several aspects including the morphological characteristics of the arm and of the hand, the bio-mechanics of the musculoskeletal system, the presence of redundant degrees of freedom and limits on the joints, non-linearity (e.g., the fact that small variations in some of the joints might have a strong impact on the hand position), gravity, inertia, collisions, noise, the need to rely on different sensory modalities, visual occlusion, the effects of movements on the next experienced sensory states, the need to coordinate arm and hand movements, the need to adjust actions on the basis of sensory feedback, and the need to handle the effects of the physical interactions between the robot and the environment. The attempt to design robots that develop their skills autonomously through an adaptive process permits, at least in principle, to delegate the solutions to some of these aspects to the adaptive process itself.

The research work described in this paper proposes an approach that takes into account most of the aspects discussed above, although often by introducing severe simplifications. More specifically, the morphological characteristics of the human arm and of the hand are taken into account by using a robot that reproduces approximately the morphological characteristics of a 3.5 year-old in term of size, shape, articulations, degrees of freedom and relative limits [13]. Some of the properties of the musculo-skeletal system have been incorporated into the model by using muscle-like actuators controlled by antagonistic motor neurons. For the sake of simplicity, the segments forming the arm, the palm, and the fingers are simulated as fully rigid bodies. However, the way in which the fingers are controlled, enable a certain level of compliance in the hand. The role of gravity, inertia, collision, and noise are taken into account by accurately simulating the physic laws of motion and the effect of collisions (see Section IV for details of the model).

One of the main characteristics of the model presented in this paper is that the robot controller adjusts its output on the basis of the available sensory feedback directly updating the forces exerted on the joints (see [14] for related approaches). The importance of the sensory feedback loop has been emphasized by other works in the literature. For example in [15] the authors describe an experiment in which a three-fingered robotic arm displays a reliable grasping behavior through a series of routines that keep modifying the relative position of the hand and of the fingers on the basis of the current sensory feedback. The movements tend to optimize a series of properties such as hand-object alignment, contact surface, finger position symmetry, etc.

In this work, the characteristics of the human brain that processes sensory and proprio-sensory information and control the state of the arm/hand actuators are modeled very loosely through the use of dynamical recurrent neural networks. The architecture of the artificial neural network employed is not inspired by the characteristics of the neuroanatomical pathways of the human brain. Also, many of the features of neurons and synapses are not taken into account (see [16], for an example of works that emulate some of the anatomical characteristics of the human brain). The use of artificial neural networks as robot

[1]The granularity refers to the extent to which the robot-environmental system is broken into small parts and to the extent to which the dynamics of the system is divided into short time periods. The term fine-grained interactions thus refer to interactions occurring at a high frequency between small parts.

controller provides several advantages with respect to alternative formalisms, such as robustness, graceful degradation, generalization and the possibility to process sensory-motor information in a way that is quantitative both in state and time. These characteristics also make neural networks particularly suitable to be used with a learning/adaptive process in which a suitable configuration of the free parameters is obtained through a process that operates by accumulating small variations.

Newborn babies display a rough ability to perform reaching, which evolves into effective reaching and grasping skills by 4/5 months, into adult-like reaching and grasping strategies by 9 months, up to precision grasping by 12/18 months [17], [18], [19]. Concerning the role of sensory modalities, the experimental evidence collected on humans indicates that young infants rely heavily on somatosensory and tactile information to carry out reaching and grasping action and they use vision to elicit these behaviors [20]. However, the use of visual information (employed to prepare the grasping behavior or to adjust the position of the hand by taking into account the shape and the orientation of the object) starts to play a role only after 9 months from birth [21]. On the basis of this, we provide our robot with proprioceptive and tactile sensors and with a vision system that only provides information concerning the position of the object but not about its shape and its orientation. Moreover, we do not simulate visual occlusions on the basis of the assumption that the information concerning the position of the object can be inferred in relatively reliable way even when the object is partially or totally occluded by the robot's arm and hand.

In accordance with the empirical evidence indicating that early manipulation skills in infants are acquired through self-learning mechanisms rather than by imitation learning [16], the robot acquires its skills through a trial and error process during which random variations of the free parameters of the robots' neural controller (which are initially assigned randomly) are retained or discarded on the basis of their effect at the level of the overall behavior exhibited by the robot in interaction with the environment. More precisely, the effect of variations is evaluated using a set of utility functions that determine the extent to which the robot manages to reach and grasp a target object with its hand, and the extent to which the robot succeeds in lifting the object over the table. The use of this adaptive algorithm and utility functions leaves the robot free to discover during the adaptive process its own strategy to reach the goals set by the experimenter. This in turn allows the robot to exploit sensory-motor coordination (i.e., the possibility to act in order to later experience useful sensory states) as well as the properties arising from the physical interactions between the robot and the environment. In [22] it is shown how this approach allows the robot to distinguish objects of different shapes by self-selecting useful stimuli through action, and in [23] it is shown how this approach allows for the exploiting of properties arising from the physical interaction between the robot body and the environment for the purpose of manipulating the object.

Finally, in this work we shape the ecological conditions in which the robot has to develop its skills by allowing the robot to access linguistic instructions that indicate the type of behavior that should be currently exhibited by the robot. We do not consider any other form of shaping, such as, for example, the possibility to expose the robot to simplified conditions in some of the trials (in which, for example, the object to be grasped is initially placed within the robot's hand) although we assume that other forms of shaping might favour the developmental process as well.

III. Experimental Setup

Our experiments involve a simulated humanoid robot that is trained to manipulate a spherical object located in different positions over a table in front of the robot by reaching, grasping, and lifting it. More specifically the robot is made up of an anthropomorphic robotic arm with 27 actuated degrees of freedom (DOF) on the arm and hand, 6 tactile sensors distributed over the inner part of the fingers and palm, 17 propriosensors encoding the current angular position of the joints of the arm and of the hand, a simplified vision system that detects the relative position of the object (but not the shape of the object) with respect to the hand and 3 sensory neurons that encode the category of the elementary behaviors that the robot is required to exhibit (i.e., reaching, grasping, or lifting the sphere). The neural controller of the robot is a recurrent neural network trained through an evolutionary algorithm for the ability: (i) to reach an area located above the object, (ii) to wrap the fingers around the object, and (iii) to lift the object over the table. The condition in which the linguistic instructions are provided has been compared with the condition in which the linguistic instructions are not provided. For each condition, the evolutionary process has been repeated 10 times with different random initializations. The robot and the robot/environmental interactions have been simulated by using Newton Game Dynamics (NGD, see: www.newtondynamics.com), a library for accurately simulating rigid body dynamics and collisions. For related approaches, see [23], [22], [24].

In section IV, we describe the structure and the actuators of the arm and hand. In section V, we describe the architecture of the robot controller and the characteristics of the sensors. In section VI, we describe the adaptive process that has been used to train the robot. In section VII, we describe the results obtained, and, finally, in section VIII, we discuss the significance of these results and our plans for the future.

IV. Robot Structure

A. Arm Structure

The arm consists mainly of three elements (the arm, the forearm, and the wrist) connected through articulations placed into the shoulder, the arm, the elbow, the forearm and wrist (see Figure 1).[2]

[2]Details about arm and hand dimensions are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

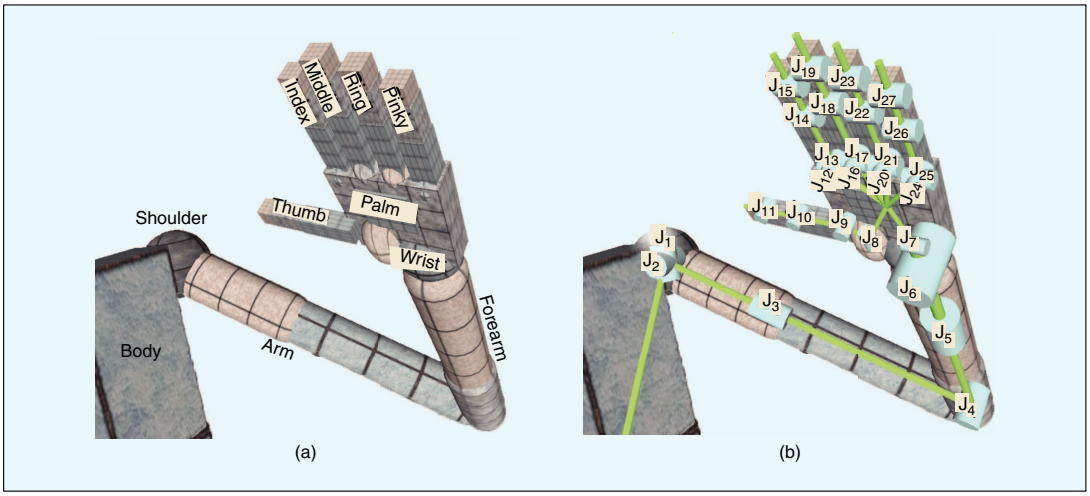


FIGURE 1 (a) The robot structure and (b) its kinematic chain. Cylinders represent rotational DOFs where its main axis indicates the corresponding axis of rotation; the links amongst cylinders represents the rigid connections that make up the arm structure.

The joints J_1, J_2 and J_3 provide *abduction/adduction, extension/flexion* and *supination/pronation* of the arm in the range $[-140^\circ, +100^\circ]$, $[-110^\circ, +90^\circ]$ and $[-110^\circ, +90^\circ]$, respectively. These three degrees of freedom (DOFs) acts like a ball-and-socket joint moving the arm in a way analogous to the human shoulder joint. J_4, located in the elbow, is a hinge joint which provides *extension/flexion* within the $[-170^\circ, +0^\circ]$ range. J_5 twists forearm providing *pronation/supination* of the wrist (and the palm) within $[-100^\circ, +100^\circ]$. J_6 and J_7 provide *flexion/extension* and *abduction/adduction* of the hand within $[-40^\circ, +40^\circ]$ and $[-100^\circ, +100^\circ]$ respectively (see Figure 1).

B. Arm Actuators

The arm joints $(J_1, \ldots, J_7)$ are actuated by two simulated antagonist muscles implemented accordingly to Hill's muscle model [25], [26]. More precisely, the total force exerted by a muscle is the sum of three forces $T_A(\alpha, x) + T_P(x) + T_V(\dot{x})$ which depend on the activity of the corresponding motor neuron (α) on the current elongation of the muscle (x) and on the muscle contraction/elongation speed $(\dot{x})$ which are calculated on the basis of the following equations:

$$
\begin{aligned}
T_A &= \alpha\left(-\frac{A_{sh}T_{\max}(x-R_L)^2}{R_L^2}+T_{\max}\right)\\
A_{sh} &= \frac{R_L^2}{(L_{\max}-R_L)^2}\\
T_P &= T_{\max}\frac{\exp\left\{K_{sh}\left(\frac{x-R_L}{L_{\max}-R_L}\right)\right\}-1}{\exp\{K_{sh}\}-1}\\
T_V &= b\cdot\dot{x},
\end{aligned}
\tag{1}
$$

where $L_{\max}$ and R_L are the maximum and resting lengths of the muscle, $T_{\max}$ is the maximum force that can be generated, K_{sh} is the passive shape factor, and b is the viscosity coefficient.

The active force T_A depends on the activation of muscle α and on the current elongation/compression of the muscle. When the muscle is completely elongated/compressed, the active force is zero regardless of the activation α. At the resting length R_L, the active force reaches its maximum that depends on the activation α. The red curves in Figure 2 show how the active force T_A changes with respect to the elongation of the muscle for some possible values of α. The passive force T_P depends only on the current elongation/compression of the muscle (see the blue curve in Figure 2). T_P tends to elongate the muscle when it is compressed less than R_L and tends to compress the muscle when it is elongated above R_L. T_P differs from a linear spring for its exponential trend that produces a large opposition to muscle elongation and

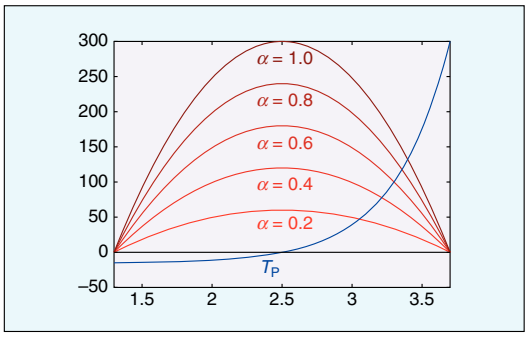


FIGURE 2 An example of the force exerted by a muscle; the graph shows how the force exerted by a muscle varies as a function of the activity of the corresponding motor neuron and of the elongation of the muscle for a joint in which $T_{\max}$ is set to 300 *N*.

little to muscle compression. T_V is the viscosity force. It produces a force proportional to the velocity of the elongation/compression of the muscle.

The parameters of the equation are identical for all 14 muscles controlling the seven DOFs of the arm and have been set to the following values: $K_{sh} = 3.0$, $R_L = 2.5$, $L_{\max} = 3.7$, $b = 0.9$, $A_{sh} = 4.34$ with the exception of parameter $T_{\max}$ which is set to $3000\,N$ for joint J_2, $300\,N$ for joints J_1, J_3, J_4, and J_5, and $200\,N$ for J_6 and J_7.

Muscle elongation is computed by linearly mapping the angular position of the DOF, on which the muscle acts, into the muscle length range. For instance, in the case of the elbow where the limits are $[-170^o, +0^o]$, this range is mapped onto $[+1.3, +3.7]$ for the agonist muscle and $[+3.7, +1.3]$ for the antagonist muscle. Hence, when the elbow is completely extended (angle 0), the agonist muscle is completely elongated (3.7) and the antagonist muscle is completely compressed (1.3), and vice versa when the elbow is flexed.

C. Hand Structure

The hand is attached to the robotic arm just after the wrist (at joint J_7 as shown in Figure 1). One of the most important features of the hand is its compliance. In details, the compliance has been obtained setting a maximum threshold of 300 N to the force exerted by each joint. When an external force acting on a joint exceeds this threshold, either the joint cannot move further, or the joint moves backward due to the external force.

The robotic hand is composed of a palm and 15 phalanges that make up the digits (three for each finger) connected through 20 DOFs, $J_8, \ldots, J_{27}$ (see Figure 1).

Joint J_8 allows the opposition of the thumb with the other fingers and it varies within the range $[-120^\circ, +0^\circ]$, where the lower limit corresponds to thumb-pinky opposition. The knuckle joints J_{12}, J_{16}, J_{20} and J_{24} allow the *abduction/adduction* of the corresponding finger and their ranges are $[0^\circ, +15^\circ]$ for the index, $[-2^\circ, +2^\circ]$ for the middle, $[-10^\circ, +0^\circ]$ for the ring, and $[-15^\circ, +0^\circ]$ for the pinky. All others joints are for the *extension/flexion* of phalanges and vary within $[-90^\circ, +0^\circ]$ where the lower limit corresponds to complete flexion of the phalanx (i.e., the finger closed).

D. Hand Actuators

The joints are not controllable independently of each other, but they are grouped. The same grouping principle used for developing the iCub hand [13] has been used. More precisely, the two distal phalanges of the thumb move together as do the two distal phalanges of the index and the middle fingers. Also, all *extension/flexion* joints of the ring and pinky fingers are linked as are all the joints of *abduction/adduction* of the fingers. Hence, only 9 actuators move all the joints of the hand, one actuator for each of the following group of joints: $\langle J_8 \rangle$, $\langle J_9 \rangle$, $\langle J_{10}, J_{11} \rangle$, $\langle J_{13} \rangle$, $\langle J_{14}, J_{15} \rangle$, $\langle J_{17} \rangle$, $\langle J_{18}, J_{19} \rangle$, $\langle J_{12}, J_{16}, J_{20}, J_{24} \rangle$ and $\langle J_{21}, J_{22}, J_{23}, J_{25}, J_{26}, J_{27} \rangle$. These actuators are simple motors controlled by position.

V. Neural Controller

The architecture of the neural controllers (see Figure 3) varies slightly depending on the ecological conditions in which the robot develops its skills. In the case of the development supported by linguistic instructions, the robot is controlled by a neural network which includes 29 sensory neurons, 12 internal neurons with recurrent connections and 23 motor neurons. In the case without the support of linguistic instructions, the neural network lacks the sensory neurons dedicated to the linguistic instructions. Thus, it is composed of 26 sensory neurons instead of 29. The sensory neurons are divided into four blocks.

The Arm Sensors encode the current angles of the 7 DOFs located on the arm and on the wrist normalized in the range $[0, 1]$.

The Hand Sensors encode the current angles of hand's joints. However, instead of feeding the network with all joint angles of the hand, the following values are used:

$$\left\langle a(J_8), a(J_9), \frac{a(J_{10}) + a(J_{11})}{2}, a(J_{13}), \frac{a(J_{14}) + a(J_{15})}{2}, a(J_{17}), \frac{a(J_{18}) + a(J_{19})}{2}, a(J_{21}), \frac{a(J_{22}) + a(J_{23})}{2}, a(J_{12}) \right\rangle,$$

where $a(J_i)$ is the angle of the joint J_i normalized in the range $[0, 1]$ with 0 meaning fully extended and 1 fully flexed. This way of representing the hand posture mirrors the way in which the hand joints are actuated (see section IV-D).

The Tactile Sensors encode how many contacts occur on the hand components. The first tactile neuron corresponds to the palm and its activation is set to the number of contacts normalized in the range $[0, 1]$ between the palm and another body (i.e., an object or other parts of the hand). Normalization is performed using a ramp function that saturates to 1 when there are more than 20 contacts. The other five tactile neurons correspond to the fingers and are activated in the same way.

The Target Position Sensors can be seen as the output of a vision system (which has not been simulated) that computes the relative distance in cm of the object with respect to the hand over three orthogonal axes. These values are fed into the networks as they are without any normalization.

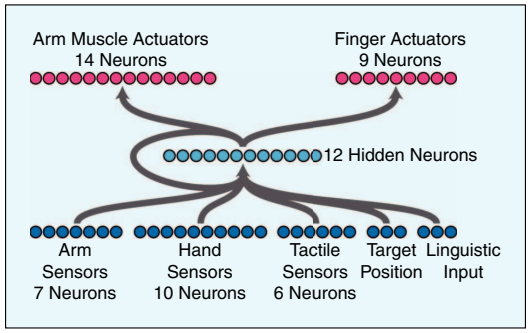


FIGURE 3 The architecture of the neural controllers. The arrows indicated blocks of fully connected neurons

The Linguistic Instruction Sensors is a block of three neurons each of which represents one of the commands *reach, grasp* and *lift*. Specifically, the vector $\langle 50, 0, 0\rangle$ corresponds to the linguistic instruction *"reach the object"*, $\langle 0, 50, 0\rangle$ corresponds to the linguistic instruction *"grasp the object"* and $\langle 0, 0, 50\rangle$ corresponds to the linguistic instruction *"lift the object"*. The way in which the state of these sensors is set is determined by equation 4 explained below.

Note that the state of the Linguistic Instruction and Target Position Sensors varies on a larger interval than the other sensors in order to increase the relative impact of these neurons. Indeed, control experiments in which all sensory neurons were normalized within the $[0, 1]$ interval led to significantly lower performance (result not shown).

The outputs $H_i(t)$ of the Hidden Neurons are calculated on the basis of following equation:

$$\begin{aligned} \gamma_i(t) &= \sigma\left(\sum_{j=1}^{29} w_{ji} I_j(t) + \beta_i\right) \\ H_i(t) &= \delta_i \cdot \gamma_i(t) + (1 - \delta_i) \cdot \gamma_i(t-1), \end{aligned} \tag{2}$$

where $I_j(t)$ is the output of the *jth* sensory neuron, w_{ji} is the synaptic weight from the *jth* sensory neuron to the *ith* hidden neuron, β_i is the bias of the *ith* hidden neuron, δ_i is the decay-factor of the *ith* hidden neuron, and $\sigma(x)$ is the logistic function with a slope of 0.2.

The output neurons are divided into two blocks, the Arm Muscle Actuators and the Finger Actuators. All outputs of these neurons are calculated in the same way using the following equation:

$$O_i(t) = \sigma\left(\sum_{j=1}^{12} w_{ji} H_j(t)\right), \tag{3}$$

where $H_j(t)$ is the output of hidden neuron j as described in 2, w_{ji} is the synaptic weight from the *jth* hidden neuron to the *ith* output neuron and $\sigma(x)$ is the logistic function with slope 0.2. With respect to the hidden neurons, the output neurons do not have any bias or decay-factor.

The Arm Muscle Actuators output sets the parameter α used in equation 1 to update the position of the arm as described in section IV-B while the Finger Actuators output sets the desired *extension/flexion* position of the nine hand actuators as described in IV-D. The state of the sensors, the desired state of the actuators, and the internal neurons are updated every 10 ms.

This particular type of neural network architecture has been chosen to minimize the number of assumptions and to reduce, as much as possible, the number of free parameters. Also, this particular sensory system has been chosen in order to study situations in which the visual and tactile sensory channels need to be integrated.

VI. The Adaptive Process

The free parameters of the neural controller (i.e., the connection weights, the biases of internal neurons and the time constant of leaky-integrator neurons) are set using an evolutionary algorithm [27], [28].

The initial population consists of 100 randomly generated genotypes, which encode the free parameters of 100 corresponding neural controllers. In the conditions in which Linguistic Instruction Sensors are employed (hereafter, referred to as Exp. A), the neural controller has 792 free parameters. In the other condition without the Linguistic Instruction Sensors (hereafter, referred to as Exp. B) there are 756 free parameters. Each parameter is encoded into a binary string (i.e., a gene) of 16 bits. In total, a genotype is composed of $792 \cdot 16 = 12672$ bits in Exp. A and $756 \cdot 16 = 12096$ bits in Exp. B. In both experiments, each gene encodes a real value in the range $[-6, +6]$, but for genes encoding the decay-factors δ_i the encoded value is mapped in the range $[0, 1]$.

The 20 best genotypes of each generation are allowed to reproduce by generating five copies each. Four out of five copies are subject to mutations and one copy is not mutated. During mutation, each bit of the genotype has a 1.5% probability to be replaced with a new randomly selected value. The evolutionary process is repeated for 1000 generations.

A. Fitness Function

The agents are rewarded for reaching, grasping and lifting a spherical object of radius 2.5 cm placed on the table in exactly the same way in both Exp. A and Exp. B. Each agent of the population is tested 4 times. Each time the initial position of the arm and the sphere change. Figure 4 shows the four initial positions of the arm and of the sphere superimposed on one another. For each initial arm/object configuration, a random displacement of $\pm 1^{o}$ is added to each joint of the arm and a random displacement of ± 1.5 cm is added on the x and the y coordinates of the sphere position. Each trial lasts 6 sec corresponding to 600 simulation steps. The sphere can move freely and it can eventually fall off the table. In this case, the trial is stopped prematurely.

The fitness function is made up of three components: *FR* for reaching, *FG* for grasping and *FL* for lifting the object. Each trial is divided in 3 phases in which only a single fitness component is updated. The conditions that define the current phase at each timestep and consequently which component has to be updated are the following:

$$\begin{aligned} r(t) &= 1 - e^{(-0.1 \cdot ds(t))} \\ g(t) &= e^{(-0.2 \cdot \text{grasp}Q(t))} \\ l(t) &= 1 - e^{(-0.3 \cdot \text{contact}(t))} \\ \text{Phase}(t) &= \begin{cases} \text{reach} & r(t) > g(t) < 0.5 \\ \text{grasp} & \text{otherwise} \\ \text{lift} & g(t) > 0.7 \wedge l(t) > 0.6, \end{cases} \end{aligned}$$

where $ds(t)$ is the distance from the center of the palm to a point located 5 cm above the center of the sphere. $\text{grasp}Q(t)$ is the distance between the centroid of the fingertips-palm polygon and the center of the sphere. $\text{contacts}(t)$ is the number of contacts between the fingers and the sphere. The shift between

the three phases is irreversible (i.e. the reach phase is always followed by the reach or grasp phases and the grasp phase is always followed by the grasp or lift phases).

Essentially, the current phase is determined by the values $r(t)$, $g(t)$ and $l(t)$. When $r(t)$ is high (i.e., when the hand is far from the object) the robot should reach the object. When $r(t)$ decreases and $g(t)$ increases (i.e., when the hand approaches the object from above) the robot should grasp the object. Finally, when $l(t)$ increases (i.e., when the number of activated contact sensors are large enough) the robot should lift the object. The rules and the thresholds included in equation 4 have been set manually on the basis of our intuition and have not been adjusted through a trial and error process. In Exp. A, the phases are used to define which linguistic instruction the robot perceives.

The three fitness components are calculated in the following way:

$$FR = \sum_{t \in T_{\text{Reach}}} \left(\frac{0.5}{1 + ds(t)/4} + \frac{0.25}{1 + ds(t)} (\text{fingersOpen}(t) + \text{palmRot}(t)) \right)$$

$$FG = \sum_{t \in T_{\text{Wrap}}} \left(\frac{0.4}{1 + \text{graspQ}(t)} + \frac{0.2}{1 + \text{contacts}(t)/4} \right)$$

$$FL = \sum_{t \in T_{\text{Lift}}} \text{objLifted}(t),$$

where T_{Reach}, T_{Wrap} and T_{Lift} are the time ranges determined by equation 4. $\text{fingersOpen}(t)$ correspond to the average degree of extension of the fingers, where 1 occurs when all fingers are extended and 0 when all fingers are closed. $\text{palmRot}(t)$ is the dot product between the normals of the palm and the table, with 1 referring to the condition in which the palm is parallel to the table and 0 to the condition in which the palm is orthogonal to the table). $\text{objLifted}(t)$ is 1 only if the sphere is not touching the table and it is in contact with the fingers, otherwise it is 0.

The total fitness is calculated at the end of four trials as: $F = \min(500, FR) + \min(720, FW) + \min(1600, FL) + bonus$, where *bonus* adds 300 for each trial where the agent switches from reach phase to grasp phase only, and 600 for each trial where the agent switches from reach to grasp phase and from grasp to lift phase.

During the reach phase the agent is rewarded for approaching a point located 5 cm above the center of the object with the palm parallel to the table and the hand open. Note that the rewards for the hand opening and the rotation of the palm are relevant only when the hand is near the object (due to $0.25/(1 + ds(t))$ factor); in this way the agent is free to rotate the palm when the hand is away from the sphere allowing any reaching trajectory.

During the grasp phase, the centroid of the fingertips-palm polygon can reach the center of the sphere only when the hand wraps the sphere with the fingers, producing a potential power grasp. During the lift phase, the reward is given when the agent effectively moves the sphere upward of the table.

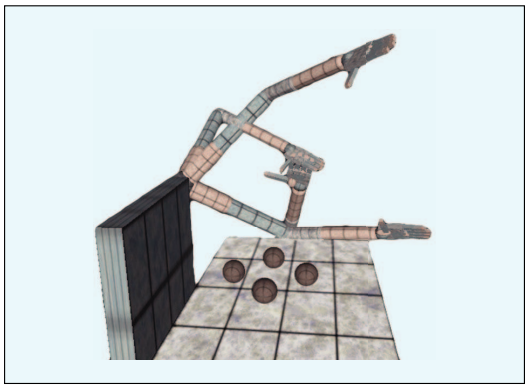


FIGURE 4 Initial positions of the arm and the sphere over imposed; the joints $J_1, \ldots J_4$ are initialized to $\langle -73, -30, -40, -56 \rangle$, $\langle -73, -30, -40, -113 \rangle$, $\langle -6, +30, -10, -56 \rangle$ and $\langle -73, -30, +45, -113 \rangle$; the initial sphere positions are $\langle -18, +10 \rangle$, $\langle -26, +18 \rangle$, $\langle -18, +26 \rangle$ and $\langle -10, +18 \rangle$.

VII. Results

For both Exp. A (with linguistic instructions) and Exp. B (without linguistic instructions), we run 10 evolutionary simulations for 1,000 generations, each using a different random initialization. Looking at the fitness curves of the best agents at each generation of each evolutionary run, we noticed that, for Exp. A, there are three distinctive evolutionary paths (see Figure 5a). The most promising is run 7, in which the last generation's agents have the highest fitness. The curve corresponding to run 2 is representative of a group of seven evolutionary paths which, after a short phase of fitness growth, reach a plateau at $F = 2{,}000$. The curve corresponding to run 9 is representative of a group of two evolutionary paths which are characterized by a long plateau slightly above $F = 1{,}000$. Generally speaking, these curves progressively increase by going through short evolutionary intervals in which the fitness grows quite rapidly followed by a long plateau [3]. For Exp. B, all the runs show a very similar trend, reaching and constantly remaining on a plateau at about $F = 3{,}000$ (see Figure 5b).

Due to the nature of the task and of the fitness function, it is quite hard to infer from these fitness curves what could be the behavior of the agents during each evolutionary phase. However, based on what we know about the task, and by visual inspection of the behavior exhibited by the agents, we found out how the agents behave at different generations of each evolutionary run. In Exp. A, the phases of rapid fitness growth are determined by the *bonus* factor, which substantially rewards those agents that successfully

[3]The fitness curves of the runs not shown are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

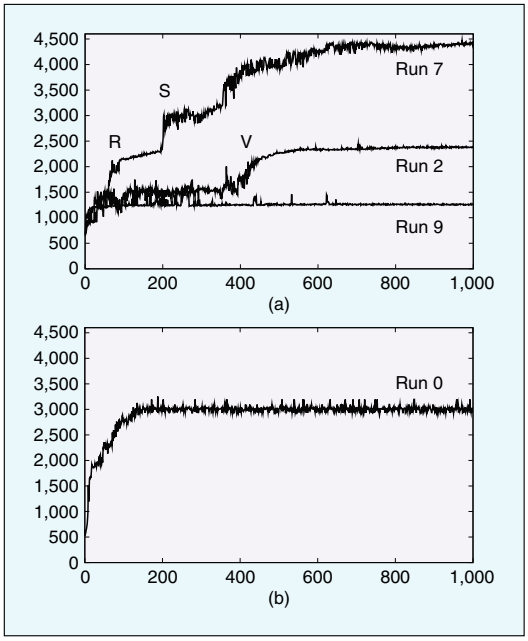


FIGURE 5 Fitness of the best agents at each generation of (a) run 2, run 7, and run 9 of Exp. A, and (b) run 0 of Exp. B.

accomplish single parts of the task. The first fitness jump is due to the *bonus* factor associated to the execution of a successful reaching behavior. This jump corresponds to the phase of fitness growth observed in run 7 in correspondence of label R Figure 5a, and in run 2 in correspondence of label V Figure 5a. The agents generated after these fitness jumps are able to systematically reach the object. Run 9 does not go through the first fitness jump, and the agents of this run lack the ability to systematically carry out a successfull reaching behavior.

The second fitness jump is due to the *bonus* factor associated with the execution of a successful grasping behavior. Only in run 7 is it possible to observe a phase of rapid fitness growth corresponding to a second fitness jump (see label S Figure 5a). The agents generated after this jump are able to successfully carry out reaching and grasping. Note also that, in run 7, the fitness curve keeps on growing until the end of the evolution. This growth is determined by the evolution of the capability to lift the object. Thus, in run 7, the best agents following generation 400 are capable of reaching, grasping, and lifting the object. The constant increment of the fitness is determined by the fact that the agents become progressively more effective in lifting the object. Run 2 does not go through a second fitness jump. The agents of this run lack the ability to systematically carry out a successfully grasping behavior.

In summary, only run 7 has generated agents (i.e., those best agents generated after generation 400) capable of successfully accomplishing reaching, grasping, and lifting.[4] The best agents of run 2, and of the other six runs that show a similar evolutionary trend, are able to systematically reach but not grasp the object and completely lack the ability to lift the object. The best agents of run 9, and of the other run that show a similar evolutionary trend, are not even able to systematically reach the object. In Exp. B, they are able to successfully reach and grasp the object, but not lift it.

A. Robustness and Generalization

In this section, we show the result of a series of post-evaluation tests aimed at establishing the effectiveness and robustness of best agents' behavioral strategies of the four runs show in Figure 5. In these tests, the agents, from generation 900 to generation 1000 of each run, are subjected to a series of trials in which the position of the object as well as the initial position of the arm are systematically varied. For the position of the object, we define a rectangular area (28 cm $\times$ 21 cm) divided in 11 $\times$ 11 cells. The agents are evaluated for reaching, grasping and lifting the object positioned in the center of each cell of the rectangular area. For the initial position of the arm, we use the four initial positions employed during evolution as prototypical cases (see Figure 4). For each prototypical case, we generate 100 slightly different initial positions with the addition of a $\pm 10^{\circ}$ random displacement on joints J_1, J_2, J_3, and J_4. Thus, this test is comprised of 48400 trials, given by 400 initial positions ($4 \cdot 100$) for each cell, repeated for 121 cells corresponding to the different initial positions of the object during the test. In each trial, reaching is considered successful if an agent meets the conditions to switch from the *reach* phase to the *grasp* phase (see equation 4). Grasping is considered successful if an agent meets the conditions to switch from the *grasp* phase to the *lift* phase (see equation 4). Lifting is considered successful if an agent manages to keep the object at more than 1 cm from the table until the end of the trial. In this section, we show the results of a single agent for each run. However, agents belonging to the same run obtained very similar performances. Thus, the reader should consider the results of each agent as representative of all the other agents of the same evolutionary run.

All the graphs in Figure 6 show the relative position of the rectangular area and the cells with respect to the agent/table system. Moreover, each cell of this area is colored in shades of grey, with black indicating 0% success rate, and white indicating 100% success rate. As expected from the previous section, the agent chosen from run 7 Exp. A proved to be the only one capable of successfully accomplishing all the three phases of the task. This agent proved capable of successfully reaching the object placed almost anywhere within the rectangular area. Its grasping and lifting behavior are less robust than the reaching behavior. Indeed, the grasping and lifting performances are quite good everywhere except in

[4]Movies of the behavior and corresponding trajectories are available at the supplementary web page http://laral.istc.cnr.it/esm/linguisticExps.

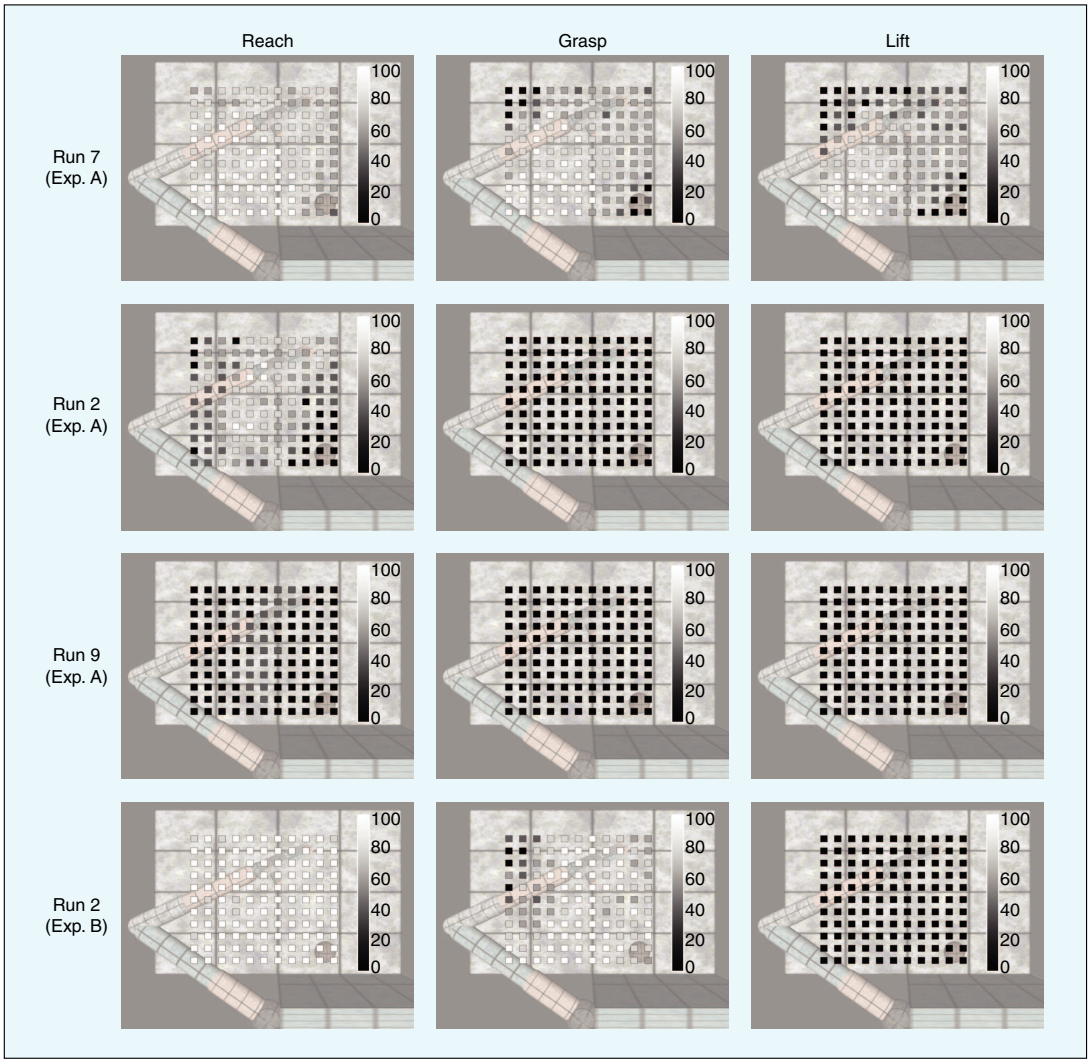


FIGURE 6 Results of post-evaluation tests on the robustness of reaching, grasping and lifting behavior of the best agent at generation 1,000 of run 7, run 2, and run 9 in Exp. A and run 0 in Exp. B. The cells in shades of grey indicate the percentage of successful trials (from 0% success rate in black, to 100% success rate in white), with the object located in the center of each cell.

two small zones located in the top left and bottom right of the rectangular area in which cells are colored black. The agent chosen from run 2 Exp. A proved to be capable of successfully performing reaching behavior for a broad range of object initial positions, and completely unable to perform grasping and lifting behavior. The agent chosen from run 9 Exp. A does not even manage to systematically bring the hand close to the object regardless of the object's initial position. The agent chosen from run 0 Exp. B, proved capable of successfully performing reaching and grasping behavior but not lifting behavior.

VIII. Conclusion

In this paper, we showed how a simulated humanoid robot controlled by an artificial neural network can acquire the ability to manipulate spherical objects located over a table by reaching, grasping and lifting them. The agent is trained through an adaptive process in which the free parameters encode the control rules that regulate the fine-grained interaction between the agent and the environment, and the variations of these free parameters are retained or discarded on the basis of their effects at the level of the behavior exhibited by the agent. This means that the agents develop their skills autonomously in interaction

with the environment. Moreover, this means that the agents are left free to determine the way in which they solve the task within the limits imposed by i) their body/control architecture, ii) the characteristics of the environment, and iii) the constraints imposed by the utility function that rewards the agents for their ability to reach an area located above the object, wrap the fingers around the object, and lift the object. The analysis of the best individuals generated by the adaptive process shows that the agents of a single evolutionary run manage to reach, grasp, and lift the object in an reliable and effective way. Moreover, when tested in new conditions with respect to those experienced during the adaptive process, these agents proved to be capable of generalising their skills with respect to new object positions never experienced before. The comparison of two experimental conditions (i.e., with or without the use of linguistic instructions that specify the behaviors that the agents are required to exhibit during the task) indicates that the agents succeed in solving the entire problem only with the support of linguistic instructions (i.e., in Exp. A). This result confirms the hypothesis that the possibility to access linguistic instructions, representing the category of the behavior that has to be exhibited in the current phase of the task, might be a crucial prerequisite for the development of the corresponding behavioral skills and for the ability to trigger the right behavior at the right time. More specifically, the fact that the best agents of Exp. B succeed in exhibiting the reaching and then the grasping behavior but not the lifting behavior suggests that the linguistic instructions represent a crucial pre-requisite in situations in which the agent has to develop an ability to produce different behaviors in similar sensory-motor circumstances. The reaching to grasping transitions are marked by well differentiated sensory-motor states, which are probably sufficient to induce the agents to stop the reaching phase and to start the grasping phase, even without the support of a linguistic instruction. The grasping to lifting transition is not characterized by well differentiated sensory-motor states. Thus, in Exp. A, it seems to be that the valuable support of the linguistic instruction induces successful agents to move on to the lifting phase.

In future work, we plan to verify whether these agents can be trained to self-generate linguistic instructions and use them to trigger the corresponding behaviors autonomously (i.e., without the need to rely on external instructions). In other words, we would like to verify whether the role played by linguistic instructions can be later internalized in agents' cognitive abilities [29], [30], [31]. Moreover, we plan to port the experiments performed in simulation in hardware by using the iCub robot and the compliant system recently developed [32]. Even though the iCub joints are stiff, the implementation of the muscle model used in this article is still possible. Two 6 axis force sensors placed on the arms and a module developed by the robotcub consortium allow the joints to react as if they were compliant. In this way, it is possible to move the joint applying a torque on its axis and thanks to the opensource aspect of the project, it would be possible to implement muscle actuation directly on the motor control boards.

IX. Acknowledgment

This research work was supported by the ITALK project (EU, ICT, Cognitive Systems and Robotics Integrating Project, grant no 214668). The authors thank their colleagues at LARAL for stimulating discussions and feedback during the preparation of this paper.

References

[1] S. F. Cappa and D. Perani, "The neural correlates of noun and verb processing," *J. Neurolinguistics*, vol. 16, no. 2–3, pp. 183–189, 2003.
[2] A. Glenberg and M. Kaschak, "Grounding language in action," *Psychon. Bull. Rev.*, vol. 9, pp. 558–565, 2002.
[3] O. Hauk, I. Johnsrude, and F. Pulvermuller, "Somatotopic representation of action words in human motor and premotor cortex," *Neuron*, vol. 41, no. 2, pp. 301–307, 2004.
[4] F. Pulvermuller, *The Neuroscience of Language. On Brain Circuits of Words and Serial Order.* Cambridge, U.K.: Cambridge Univ. Press, 2003.
[5] G. Rizzolatti and M. A. Arbib, "Language within our grasp," *Trends Neurosci.*, 1998.
[6] A. Cangelosi, V. Tikhanoff, J. F. Fontanari, and E. Hourdakis, "Integrating language and cognition: A cognitive robotics approach," *IEEE Comput. Intell. Mag.*, vol. 2, no. 3, pp. 65–70, 2007.
[7] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. L. Nehaniv, K. Fischer, J. Tani, G. Sandini, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel, "Integration of action and language knowledge: A roadmap for developmental robotics," *Tech. Rep.*, 2010.
[8] S. Nolfi, "Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour," *Complexus*, vol. 2, no. 3–4, pp. 195–203, 2005.
[9] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen, "Autonomous mental development by robots and animals," *Science*, vol. 291, no. 5504, pp. 599–600, 2001.
[10] J. Weng, "Developmental robotics: Theory and experiments," *Int. J. Humanoid Robot.*, vol. 1, no. 2, pp. 199–236, 2004.
[11] S. Schaal, "Arm and hand movement control," in *Handbook of Brain Theory and Neural Networks*, 2nd ed., M. Arbib, Ed. Cambridge, MA: MIT Press, 2002, pp. 110–113.
[12] M. Gienger, M. Toussaint, N. Jetchev, A. Bendig, and C. Goerick, "Optimization of fluent approach and grasp motions," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots*. IEEE Press, 2008, pp. 111–117.
[13] G. Sandini, G. Metta, and D. Vernon, "Robotcub: An open framework for research in embodied cognition," *Int. J. Humanoid Robot.*, 2004.
[14] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Proc. Int. Symp. Robotics Research (ISRR2003)*, S. verlag, Ed. 2004, pp. 1–10.
[15] J. Felip and A. Morales, "Robust sensor-based grasp primitive for a three-finger robot hand," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2009.
[16] E. Oztop, N. S. Bradley, and M. A. Arbib, "Infant grasp learning: A computational model," *Exp. Brain Res.*, vol. 158, no. 4, pp. 480–503, 2004.
[17] C. von Hofsten, "Eye-hand coordination in the newborn," *Dev. Psychol.*, vol. 18, pp. 450–461, 1982.
[18] C. von Hofsten, "Developmental changes in the organization of prereaching movements," *Dev. Psychol.*, vol. 20, pp. 378–388, 1984.
[19] C. von Hofsten, "Structuring of early reaching movements: a longitudinal study," *J. Mot. Behav.*, vol. 23, pp. 280–292, 1991.
[20] P. Rochat, "Self-perception and action in infancy," *Exp. Brain Res.*, vol. 123, pp. 102–109, 1998.
[21] M. K. McCarty, R. K. Clifton, D. H. Ashmead, P. Lee, and N. Goulet, "How infants use vision for grasping objects," *Child Dev.*, vol. 72, pp. 973–987, 2001.
[22] E. Tuci, G. Massera, and S. Nolfi, "Active categorical perception of object shapes in a simulated anthropomorphic robotic arm," *IEEE Trans. Evol. Comput.*, to be published.
[23] G. Massera, A. Cangelosi, and S. Nolfi, "Evolution of prehension ability in an anthropomorphic neurorobotic arm," *Front. Neurorobot.*, vol. 1, pp. 1–9, 2007.
[24] T. Buehrmann and E. A. Di Paolo, "Closing the loop: Evolving a model-free visually-guided robot arm," in *Proc. 9th Int. Conf. Simulation and Synthesis of Living Systems*, J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. Watson, Eds. Cambridge, MA: MIT Press, 2004, pp. 63–68.
[25] T. G. Sandercock, D. C. Lin, and W. Z. Rymer, "Muscle models," in *Handbook of Brain Theory and Neural Networks*, 2nd ed., M. Arbib, Ed. Cambridge, MA: MIT Press, 2002, pp. 711–715.
[26] R. Shadmehr and S. P. Wise, *The Computational Neurobiology of Reaching and Pointing: A Foundation for Motor Learning*. Cambridge, MA: MIT Press, 2005.
[27] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press, 2000.
[28] X. Yao and M. M. Islam, "Evolving artificial neural network ensembles," *IEEE Comput. Intell. Mag.*, vol. 3, no. 1, pp. 31–42, 2008.
[29] L. S. Vygotsky, *Thought and Language*. Cambridge, MA: MIT Press, 1962.
[30] L. S. Vygotsky, *Mind in Society*. Cambridge, MA: Harvard Univ. Press, 1978.
[31] M. Mirolli and D. Parisi. (2009). *Towards a vygotskyan cognitive robotics: The role of language as a cognitive tool. New Ideas Psychol.* [Online]. Available: http://www.sciencedirect.com/science/article/B6VD4-4X00P73-1/2/5eb2e93d' 3fc615eea3ec0fb37af6fc89
[32] V. Mohan, J. Zenzeri, P. Morasso, and G. Metta, "Equilibrium point hypothesis revisited: Advances in the computational framework of passive motion paradigm," pp. 1–3.

An Experiment on Behavior Generalization and the Emergence of Linguistic Compositionality in Evolving Robots

Elio Tuci, Tomassino Ferrauto, Arne Zeschel, Gianluca Massera, and Stefano Nolfi

***Abstract*—Populations of simulated agents controlled by dynamical neural networks are trained by artificial evolution to access linguistic instructions and to execute them by indicating, touching, or moving specific target objects. During training the agent experiences only a subset of all object/action pairs. During postevaluation, some of the successful agents proved to be able to access and execute also linguistic instructions not experienced during training. This owes to the development of a semantic space, grounded in the sensory motor capability of the agent and organized in a systematized way in order to facilitate linguistic compositionality and behavioral generalization. Compositionality seems to be underpinned by a capability of the agents to access and execute the instructions by temporally decomposing their linguistic and behavioral aspects into their constituent parts (i.e., finding the target object and executing the required action). The comparison between two experimental conditions, in one of which the agents are required to ignore rather than to indicate objects, shows that the composition of the behavioral set significantly influences the development of compositional semantic structures.**

***Index Terms*—Artificial neural networks, behavior generalization, compositional semantics, evolutionary robotics.**

I. Introduction

RECENT research on action and language processing in humans and animals clearly demonstrates the strict interaction and codependence between language and action (e.g., [1]–[5]).

For example, in [3] the authors describe a seminal psychological study showing that the execution of actions (e.g., bringing something close to or far away from to the body) facilitates/disrupts the comprehension of concurrently presented sentences which imply similar/opposite actions (e.g., sentence direction toward/away from the body). According to the authors, the results of this study show that understanding a sentence invokes the same cognitive mechanisms as those used in planning and executing actions. On the neurophysiological side, the authors in [6] performed a study in which by means of single-pulse transcranial magnetic stimulation, either the hand or the foot/leg motor area in the left hemisphere was stimulated in distinct experimental sessions, while participants were listening to sentences expressing hand and foot actions. The results of the study show that processing verbally presented actions activates different sectors of the motor system, depending on the effector used in the described action. The authors conclude that certain action words modulate areas of the brain concerned with performing those actions.

Developmental psychology studies based on emergentist and constructivist approaches also support a view of cognitive development strongly dependent on the contribution of various cognitive capabilities (e.g., [7]–[9]). These studies demonstrate the gradual emergence of linguistic constructions built through the child's experience with her social and physical environment. This is in line with the cognitive linguistic assumption that linguistic categorization involves the same principles and mechanisms that also underlie nonlinguistic cognition (see [10] and [11]).

In recent years, a fruitful exchange of ideas between roboticists and cognitive linguists has begun to develop. On the one hand, more and more language-related research in robotics embraces key ideas of the usage-based language model developed in cognitive linguistics [12], [13]. Several roboticists explicitly acknowledge this framework as their main theoretical inspiration on the language side (e.g., [14]–[17]). On the other hand, it is becoming progressively more common for cognitive linguists to draw on insights and suggestions from works on computational modelling (see, e.g., [18] and [19]). This is especially evident in the field of language acquisition, where computational modelling has become a prominent aspect of the research agenda of various scientists (see [20]–[22], for recent reviews).

In this paper, we describe a further robotic model designed to look at aspects related to the emergence of compositional semantic structures in simulated agents. Our results demonstrate how the agents, trained to execute several actions by responding to linguistic instructions, can generalize their linguistic and behavioral skills to never experienced instructions through the production of appropriate behaviors. The analysis of the best agents and the comparison of different experimental conditions, in which the representation of the linguistic instructions is the same but in which the behavioral set is varied, demonstrates how the emergence of compositional semantics is affected by the presence of behavioral regularities in the execution of different actions. Postevaluation tests also unveil further details of

Manuscript received August 19, 2010; revised November 19, 2010; accepted January 28, 2011. Date of publication February 14, 2011; date of current version June 15, 2011. This work was supported by the *ITALK* Project (EU, ICT, Cognitive Systems and Robotics Integrating Project, Grant 214668).

E. Tuci, T. Ferrauto, G. Massera, and S. Nolfi are with the ISTC-CNR, 00185 Rome, Italy (e-mail: elio.tuci@istc.cnr.it; tomassino.ferrauto@istc.cnr.it; gianluca.massera@istc.cnr.it; stefano.nolfi@istc.cnr.it).

A. Zeschel is with the Institute of Business Communication and Information Science, University of Southern Denmark, 6400 Sønderborg, Denmark, (e-mail: zeschel@uni-bremen.de).

Digital Object Identifier 10.1109/TAMD.2011.2114659

the behavioral and linguistic strategies used by agents equipped with compositional semantics to accomplish the task.

The paper is structured as follow. Section II reviews the most relevant works in the literature and in particular those described in (see [23]–[25]), which have been particularly inspiring for our work. Section III describes the task investigated in this research work and the agents' morphological structure. In Sections IV, V, and VI, we describe the agent's control system, the evolutionary algorithm and the fitness function used to design it. In Section VII, we illustrate the results of a series of postevaluation analyses. In Section VIII, we express some reflections on potential connections between empirical studies of child language learning and robotic models trying to indicate fruitful directions for future work. Conclusions are presented in Section IX.

II. Background

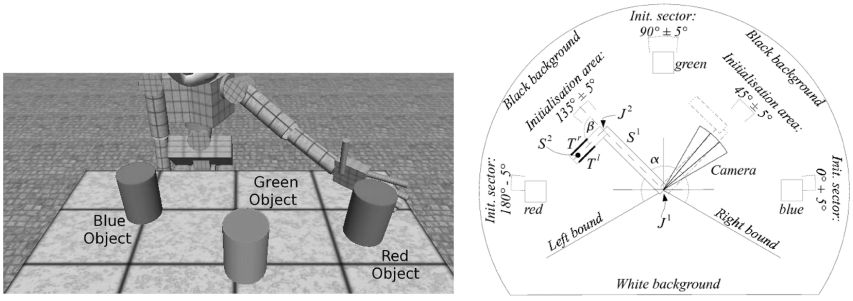
By the term "compositional semantics," we refer to a functional dependence of the meaning of an expression on the meaning of its parts. Compositional semantics in natural language refers to the human ability to understand the meaning of spoken or written sentences from the meaning of their parts, and the way in which these parts are put together. For example, the meaning of an unknown sentence like "Susan likes tulips" can be understood by learning the following three sentences: "Julie likes daisies," "Julie likes tulips," and "Susan likes daisies." In this example, the meaning of the original sentence is achieved through compositional semantics by generalizing the meaning of single words from a known (already learned) to an unknown (yet to be learned) context.

During the cognitivist era, compositionality was supposed to be underpinned by concatenative processes in which the tokens of an expression's constituents (and the sequential relations among them) are preserved in the expression itself [26]. The difficulties shown by classic symbolic AI in accounting for general associations between semantic representations and sensory–motor profiles, and in particular in accounting for the acquisition of linguistic semantics through behavioral experiences, determined a paradigm shift in which an alternative perspective on compositionality emerged (see [27] for a critical perspective on classic AI). In the last decade of the previous century, the connectionist approach to cognition proposed the idea of functional compositionality; that is compositional semantics systems in which the tokens of an expression's constituents (and the sequential relations among them) are not preserved in the expression itself [28]. Various connectionist models proved that artificial neural networks can be employed to physically instantiate functional compositional semantic structures [29].

More recently, autonomous (real or simulated) robots have been used to investigate how a form of language can emerge and evolve in a population of robots interacting between themselves and with the physical environment [30]–[33]. Moreover, several works have investigated how a robot can acquire a language by interacting with a human user. For example, in [34], the authors designed robotic experiments with robots that, in addition to react to language commands issued by the user are also able to acquire both the meaning of new linguistic instructions and new behavioral skills on the fly, by grounding the new commands in preexisting motor skills. In [35] the authors designed robots able to cooperate and to share attention with a human user in a restricted experimental setting. This is achieved by allowing the robot to observe the goal-directed behavior exhibited by the user and to adopt her plan. In [36], the author designed a developmental learning architecture that allows a robot to progressively expand its behavioral repertoire while interacting with a human trainer that shapes its behavior. In [37], the authors studied how new, higher-order behavioral abilities can be autonomously built upon previously-grounded basic action categories, acquired through language-mediated interactions with human users.

In [23]–[25], the authors investigate the issue of grounding compositional semantic structures in an agent's sensory-motor skills in tasks that require the shift from rote knowledge to systematized knowledge. In particular, in [23] and [25] a robot learns to execute actions in response to linguistic instructions consisting in two-words sentences. The robots neural controller comprises a behavioral and a linguistic module. The behavioral module is trained through a learning-by-demonstration method in which the sensory–motor states experienced while the robot is moved by the experimenter, through teleoperation or kinesthetic teaching, are used as a training set. The linguistic module is trained to predict the next word of a two-word linguistic instructions in which the words are provided to the agent sequentially. In [25], both thebehaviorall and the linguistic module are trained only on a subset of all possible linguistic instructions resulting from the combination of all possible objects with all possible actions. In [23], the linguistic module is trained only on a subset of all possible linguistic instructions whereas the behavioral module is trained to execute all the possible instructions. In all three studies [23]–[25], the agent proves capable of performing actions associated with linguistic instructions not experienced during training. The authors claim that behavioral and/or linguistic generalization is achieved by "conceiving something not experienced as a recombination of learned examples" (see [23], for details). The contribution of these works is in bringing evidence for a dynamical perspective on compositional semantic systems, alternative to the one in which neural correlates of language are viewed as atomic elements semantically associated to basic units of the linguistics system. The authors show that compositional systems can be underpinned by neural structures in which the neural correlates of the linguistic instructions are dynamically self-organized topological properties of the neural substrate, induced by similarities among sensory–motor sequences. Each instruction (i.e., action plus object) is represented in a two-dimensional semantic space by a single point which lies in a grid-like geometrical structure in which one dimension refers to actions and the other to objects. The geometrical arrangement of neural correlates that emerged during the simultaneous training of the behavioral and linguistic modules, allows the agent to successfully respond to nonexperienced linguistic instructions.

In this paper, we describe a series of simulations in which a robot is required to perform a task very similar to the one described in [23]. As in [23], our goal is also to investigate the emergence and the underlying properties of a functionally compositional semantic system in a task that requires the shift from rote knowledge to systematized knowledge. However, we look



(a) (b)

Fig. 1. (a) Image of the simulated iCub and its word. (b) Schematic representation of the agent structure and its world in the 2-D simulator. The vision system of the agent is drawn only with respect to the arm initialized on the right initialization area. α refers to the angular position of S^1. β refers to the angular position of S^2 with respect to S^1. See text for further details.

at the problem with different methods that, as we will see, lead to a qualitatively different type of solution. In our case, a neural controller is trained to execute a subset of possible linguistic instructions through an evolutionary method in which the robot is rewarded for the ability to achieve a certain goal without specifying the sequence of movements through which this goal should be realized. As shown in Section VII, this allows the robot to codevelop linguistic skills to access the meaning of the instructions and behavioral skills to execute them.

III. The Agent Structure and the Task

The experimental scenario concerns a humanoid iCub robot [38] placed in front of a table with a red, green, and blue object as shown in Fig. 1(a). The robot is trained to execute seven actions on the object by responding to linguistic instructions formed by all the possible combinations of the three action words "INDICATE," "TOUCH," and "MOVE" and the three object word "Red," "Green," and "Blue" with the exception of the sentences "TOUCH Green object" and "MOVE Blue object." After training, the robot is then tested on the two *nonexperienced* sentences to assess whether it produces the appropriate corresponding behaviors even though it had neither experienced these sentences before nor received training on the two corresponding behaviors. To reduce the computational costs associated to the simulation of such a complex robot, we carried out our experiments on a simpler experimental 2-D scenario involving a two-segments arm described below. We then port the obtained results on a simulated iCub by controlling the robots hand position on the basis of the current position of the end-effector of the simplified arm through the inverse kinematic software described in [39]. The best evolved controllers have been successfully ported on the iCub simulator.[1]

In the simple two-dimensional simulated world, an agent is composed of an arm with two segments referred to as S^1 (100 cm) and S^2 (50 cm), and two degrees of freedom (DOF). Each DOF comprises a rotational joint which acts as the fulcrum and an actuator. The first actuator causes S^1 to rotate clockwise or anticlockwise around joint J^1, with the movement restricted in the right $(-30°)$ and the left $(210°)$ bound. The other actuator causes S^2 to rotate clockwise or anticlockwise around joint J^2 within the range [90°, 0°] with respect to S^1 [see Fig. 1(b)]. Friction and momentum are not considered.

In the environment there are three objects of different colors (i.e., a blue, a green, and a red object). The objects are placed 150 cm from J^1 with their center placed anywhere on the chord delimiting their corresponding Init. sector [see Fig. 1(b)]. The objects do not move unless pushed by the arm. The agent is equipped with a linear camera with a receptive field of 30°, divided in three sectors, each of which has three binary sensors (C_i^B for blue, C_i^G for green, and C_i^R for red, with $i \in [1, 2, 3]$ sectors). Each sensor returns 1 if the blue/green/red object falls within the corresponding sector. If no colored object is detected, the readings of the sensors are set to 0 (i.e., the camera perceives a black background). The camera and S^1 move together. The experimental set up is built in a way that at each time step there can be only one object in the camera view.

The agent has means to perceive whenever S^1 reaches the right or the left bound through the activation of the camera sensors. That is, when S^1 reaches the right bound C_1^B, C_1^G, and C_1^R are set to 1 (i.e., the first camera sector perceives a white background). When S^1 reaches the right bound C_3^B, C_3^G, and C_3^R are set to 1 (i.e., the third camera sector perceives a white background). Finally, two binary touch sensors (i.e., T^r, T^l) are placed on the right, and left side of S^2. Collisions between the agent and an object are handled by a simple model in which whenever S^2 pushes the object the relative contact points remain fixed.

To assess whether the composition of the behavioral set affects the developmental process and the generalization capabilities of the agents, we run two sets of evolutionary experiments. In the **With-Indicate** experimental condition, the task consists in the execution of the following instructions: TOUCH Blue object $(\mathrm{Inst}^T_{\mathrm{blue}})$, TOUCH Red object $(\mathrm{Inst}^T_{\mathrm{red}})$, MOVE Green object $(\mathrm{Inst}^M_{\mathrm{green}})$, MOVE Red object $(\mathrm{Inst}^M_{\mathrm{red}})$, INDICATE Blue object $(\mathrm{Inst}^{\mathrm{IN}}_{\mathrm{blue}})$, INDICATE Green object $(\mathrm{Inst}^{\mathrm{IN}}_{\mathrm{green}})$, and INDICATE Red object $(\mathrm{Inst}^{\mathrm{IN}}_{\mathrm{red}})$. In the **With-Ignore** experimental condition, the action INDICATE is substituted with the action IGNORE. Thus, $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{blue}}$ refers to IGNORE Bkue object, $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{green}}$ refers to IGNORE Green object, and $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{red}}$ refers to IGNORE Red object. For both

[1]Movies and further methodological details concerning the porting can be found at http://laral.istc.cnr.it/esm/tuci-etal-IEEE_TAMD2010/.

TABLE I
THE LINGUISTIC INSTRUCTIONS. IN GREY THE *NONEXPERIENCED* INSTRUCTIONS, THAT IS, THOSE NOT EXPERIENCED DURING TRAINING. THE TABLE ALSO SHOWS THE NOTATION USED IN (1) TO REFER TO EACH BIT OF THE LINGUISTIC INSTRUCTIONS

	MOVE $Inst_o^M$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	0	1	1
Green	1	0	1	0	1	1
Red	0	1	1	0	1	1

	TOUCH $Inst_o^T$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	1	0	1
Green	1	0	1	1	0	1
Red	0	1	1	1	0	1

	INDICATE $Inst_o^{IN}$ - IGNORE $Inst_o^{IG}$					
	Object			Action		
	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}
Blue	1	1	0	1	1	0
Green	1	0	1	1	1	0
Red	0	1	1	1	1	0

evolutionary conditions, the linguistic instructions experienced during training are referred to as *experienced* instructions, while the instructions TOUCH Green object ($\mathrm{Inst}^T_{\mathrm{green}}$) and MOVE Blue object ($\mathrm{Inst}^M_{\mathrm{blue}}$), never experienced during training, are referred to as *nonexperienced* instructions (see also Table I). The object-label and the action-label are given to the agent concurrently and for the entire duration of a trial.

TOUCH and MOVE require the agent to rotate S^1 and S^2 until S^2 collides with the target object. TOUCH requires an agent to remain in contact with the target object with the right side of S^2 (that is, by activating the touch sensor T^r) for an uninterrupted interval of 100 time steps. During this interval, S^1 must not rotate. MOVE requires an agent to rotate S^1 more than 35° while S^2 is touching the object with its right side. The rotation of S^1 while S^2 is touching the object determines the movement of the object. INDICATE requires an agent to rotate S^1 until the angular distance between S^1 and the object is less than 30°. INDICATE is correctly executed only if S^1 remains at less than 30° from the target object for more than 100 time steps. IGNORE requires the agent to look at anything except the target object. The agent has to move away from positions in which the target object falls within its visual field. During the execution of INDICATE and IGNORE, an agent must not collide with any object. During the execution of TOUCH and MOVE, an agent must not collide with the nontarget objects (i.e., the objects not mentioned in the current linguistic instruction).

After training, all the agents are evaluated for their capability to access *experienced* and *nonexperienced* linguistic instructions and to execute the corresponding behaviors.

IV. THE AGENT CONTROLLER

The agent controller is composed of a continuous time recurrent neural network (CTRNN) of 18 sensor neurons, three interneurons, and four motor neurons [40]. At each time step sensor neurons are activated using an input vector I_i with $i \in [1 \dots 18]$ corresponding to the sensors readings. In particular,

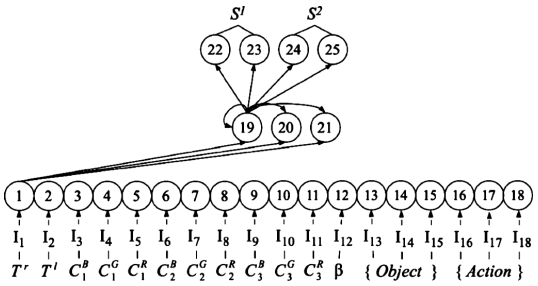


Fig. 2. Neural network. Continuous line arrows indicate the efferent connections for the first neuron of each layer. Underneath the input layer, it is shown the correspondences between sensors/linguistic instructions, the notation used in (1) to refer to them, and the sensory neurons.

I_1 and I_2 are the readings of touch sensors T^r and T^l, respectively; I_3 to I_{11} are the readings of the camera sensors; I_{12} is refers to the normalized angular position of S^2 with respect to S^1 (i.e., β); I_{13} to I_{18} are the linguistic input and their value depend on the current linguistic instruction. I_{13}, I_{14}, and I_{15} identify the object, I_{16}, I_{17}, and I_{18} identify the action to execute (see Fig. 2).

The interneuron network is fully connected. Additionally, each interneuron receives one incoming synapse from each sensory neuron. Each motor neuron receives one incoming synapse from each interneuron. There are no direct connections between sensory and motor neurons. The states of the motor neurons are used to control the movement of S^1 and S^2 as explained later. The states of the neurons are updated using the following equation:

$$\Delta y_i = -y_i + gI_i; \quad \text{for } i \in \{1 \dots 18\} \tag{1}$$

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{21} \omega_{ji}\sigma(y_j + \beta_j); \quad \text{for } i \in \{19 \dots 21\} \tag{2}$$

$$\Delta y_i = -y_i + \sum_{j=19}^{21} \omega_{ji}\sigma(y_j + \beta_j); \quad \text{for } i \in \{22 \dots 25\} \tag{3}$$

with $\sigma(x) = (1 + e^{-x})^{-1}$. In these equations, using terms derived from an analogy with real neurons, y_i represents the cell potential, τ_i the decay constant, g is a gain factor, I_i the intensity of the perturbation on sensory neuron i, ω_{ji} the strength of the synaptic connection from neuron j to neuron i, β_j the bias term, $\sigma(y_j + \beta_j)$ the firing rate (hereafter, f_i). All sensory neurons share the same bias (β^I), and the same holds for all motor neurons (β^O). τ_i and β_i with $i \in \{19 \dots 21\}$, β^I, β^O, all the network connection weights ω_{ij}, and g are genetically specified networks' parameters. At each time step the angular movement of S^1 is $2.9H(f_{22} - 0.5)\mathrm{sgn}(0.5 - f_{23})$ degrees and of S^2 is $2.9H(f_{24} - 0.5)\mathrm{sgn}(0.5 - f_{25})$ degrees, where H is the Heaviside step function and sgn is the sign function. Cell potentials are set to 0 when the network is initialized or reset, and (2) is integrated using the forward Euler method with an integration time step $\Delta T = 0.1$.

V. The Evolutionary Algorithm

A simple generational genetic algorithm is employed to set the parameters of the networks [41]. At generation 0, a random population of 100 vectors is generated by initializing each component of each vector to a value chosen uniformly random in the range [0, 1]. Each vector comprises 84 real values (i.e., 75 connection weights ω_{ji}, three decay constants τ_i, five bias term β, and one gain factor g shared by all the sensory neurons). Hereafter, using terms derived from an analogy with biological systems, a vector is referred to as genotype and its components as genes.

Generations following the first one are produced by a combination of selection with elitism and mutation. For each new generation, the three highest scoring genotypes ("the elite") from the previous generation are retained unchanged. The remainder of the new population is generated by fitness-proportional selection from the 50 best genotypes of the old population. New genotypes, except "the elite," are produced by applying mutation. Mutation entails that a random Gaussian offset is applied to each gene, with a probability of 0.4. The mean of the Gaussian is 0, and its standard deviation is 0.1. During evolution, all genes are constrained to remain within the range [0, 1]. That is, if due to mutations a gene falls below zero, its value is fixed to 0; if it rises above 1, its value is fixed to 1.

Genotype parameters are linearly mapped to produce network parameters with the following ranges: $\beta^I \in [-4,-4]$, β^O in $[-5,-5]$, β_i in $[-5,-5]$ with $i \in \{19\ldots21\}$, $\omega_{ij} \in [-8,8]$, with $i \in \{1\ldots18\}$, and $j \in \{19\ldots21\}$, $\omega_{ij} \in [-10,10]$, with $i \in \{19\ldots21\}$, and $j \in \{19\ldots25\}$, gain factor $g \in [1,13]$. Decay constants τ_i with $i \in \{19\ldots21\}$, are firstly linearly mapped into the range $[-1.0, 2.0]$ and then exponentially mapped into $\tau_i \in [10^{-1.0}, 10^{2.0}]$. The lower bound of τ_i corresponds to the integration step-size used to update the controller; the upper bound, arbitrarily chosen, corresponds to about 4% of the maximum length of a trial.

VI. The Fitness Function

During evolution, each genotype is translated into an arm controller and evaluated more than once for all the object–action *experienced* instructions by varying the starting positions. The agents perceive *experienced* instructions and they are required to execute the corresponding behaviors. Agents are evaluated 14 times initialized in the left and 14 times in the right initialization area, for a total of 28 trials. For each initialization area, an agent experiences all the *experienced* linguistic instructions twice. The *nonexperienced* linguistic instructions $\mathrm{Inst}^M_{\mathrm{blue}}$ and $\mathrm{Inst}^T_{\mathrm{green}}$ are never experienced during the training phase. At the beginning of each trial, the agent is randomly initialized in one of the two initialization area, and the state of the neural controller is reset. A trial lasts 25 simulated seconds ($T = 250$ time steps). A trial is terminated earlier in case the arm collides with a non target object. In each trial k, an agent is rewarded by an evaluation function which seeks to assess its ability to execute the desired action on the target object.

A. With-Indicate

In **With-Indicate**, the fitness F^{tot}_k attributed to an agent in trial k is the sum of three fitness components F^1_k, F^2_k, and F^3_k. F^1_k rewards the agent for reducing the angular distance between S^1 and the target object. F^2_k rewards the agent for performing the required action on the target object. F^3_k rewards the agent for extending S^2 when it is perceiving the target object and it is required to touch or to move it

$$F^{\mathrm{tot}} = \frac{1}{K}\sum_{k=1}^{K} F^{\mathrm{tot}}_k; \quad \text{with } K = 28;\ F^{\mathrm{tot}}_k = F^1_k + F^2_k + F^3_k \tag{4}$$

F^1_k, F^2_k, and F^3_k are computed as follows:

$$F^1_k = \max\left(0, \frac{d^i - d^f}{d^i} \cdot P^1_k, \mathbb{1}_{d^f<4.6^\circ}\right) \tag{5}$$

where d^i and d^f are, respectively, the initial (i.e., at $t = 0$) and final (i.e., at the end of the trail k) angular distances between S^1 and the target object and $\mathbb{1}_{d^f<4.6^\circ}$ is 1 if $d^f < 4.6^\circ$, 0 otherwise. P^1_k is the penalty factor. It is set to 0.6 if the agent collides with a non target object, otherwise to 1.0. The angle between S^1 and the target object o can be measured *clockwise* ($\alpha^{\mathrm{clock}}_o$) or *anticlockwise* ($\alpha^{\mathrm{anti}}_o$). In (5), d^i and d^f are the minimum between the clockwise and anticlockwise distance, that is $d = \min(\alpha^{\mathrm{clock}}_o, \alpha^{\mathrm{anti}}_o)$. See (6a)–(6b) at the bottom of the page, where $max{-}steps{-}on{-}target = 100$, $P^2_k = 0$ if $F^1_k < 1$ otherwise $P^2_k = 1$, $max - angular - offset = 34.4^\circ$. For the action INDICATE, *steps-on-target* refers to the number of time steps during which $F^1_k = 1$, and S^2 does not touch the target object. For the action TOUCH, *steps-on-target* refers to the number of time steps during which $F^1_k = 1$, S^2 touches the target object by activating the touch sensor T^r, and S^1 does not change its angular position. $\Delta\theta$ is the angular displacement of the orientation of S^1 recorded while $F^1_k = 1$, and S^2 is touching the target object by activating the touch sensor T^r.

$$F^3_k = 1.0 - \frac{\beta}{0.5\pi}; \tag{7}$$

with β corresponding to the angular position of S^2 with respect to S^1. F^3_k is computed only when the target object is falling within the visual field of the agent and in those trials in which the agent is required to touch or to move the target object. If the current linguistic instruction requires the agent to indicate an object and during the time of a trial in which the agent is not

$$F^2_k = \begin{cases} \dfrac{\text{steps-on-target}}{\text{max-steps-on-target}} \cdot P^2_k; & \text{for TOUCH or INDICATE} \quad (6a) \\ \dfrac{\Delta\theta}{\text{max-angular-offset}} \cdot P^2_k; & \text{MOVE} \quad (6b) \end{cases}$$

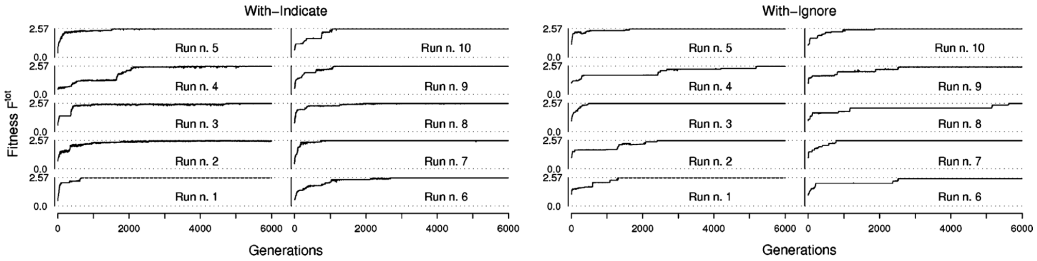


Fig. 3. Graphs showing the fitness curves of the best agent at each generation of ten evolutionary Runs in condition (a) **With-Indicate**; (b) **With-Ignore**.

perceiving the target object $F_k^3 = 0$. A trial is terminated earlier if $steps{-}on{-}target = max{-}steps{-}on{-}target$ during the execution of INDICATE or TOUCH and when $\Delta\theta = max - angular - offset$ during the execution of MOVE.

B. With-Ignore

With-Ignore differs from **With-Indicate** only in the computation of F_k^1 and F_k^2 during the execution of the linguistic instructions IGNORE Blue object $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{blue}}$, IGNORE Green object $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{green}}$, and IGNORE Red object $\mathrm{Inst}^{\mathrm{IG}}_{\mathrm{red}}$. During the trials in which an agent is required to IGNORE an object $F_k^1 = 1$ if at the end of the trial the target object does not fall within the visual field of the agent, otherwise $F_k^1 = 0$.

$$F_k^2 = \frac{\text{steps-out-of-target}}{\text{max-steps-out-of-target}} \cdot P_k^2;\ \text{for IGNORE} \quad (8)$$

where $max{-}steps{-}out{-}of{-}target = 100$, and *steps-out-of-target* refers to the number of time steps during which $F_k^1 = 1$, and S^2 does not touch the target object.

VII. Results

For each experimental condition (**With-Indicate**, and **With-Ignore**), we run ten evolutionary simulations for 6000 generations, each using a different random initialization. Recall that our objective is to generate agents that are capable of successfully accessing and executing *experienced* linguistic instructions. Moreover, we are interested in investigating whether agents develop semantic structures that are functionally compositional. Agents endowed with functionally compositional semantics should be able to successfully access and execute *experienced* linguistic instructions and to generalize their linguistic and behavioral skills to *nonexperienced* instructions (i.e., linguistic instructions never experienced during training). We run two different series of simulations to test whether a different training bears upon the development of the required mechanisms for compositional semantics.

Fig. 3 shows the fitness of the best agent at each generation of ten evolutionary Runs per condition. All the curves reach a stable plateau with fitness either firmly fixed or progressing with small oscillation around the maximum score (i.e., $F^{\mathrm{tot}} \simeq 2.57$). There are Runs in which the agents reach the maximum fitness very quickly (e.g., Run n° 1 condition **With-Indicate**, or in Run n° 2 condition **With-Ignore**) other in which it takes longer (e.g., Run n° 4 condition **With-Indicate**, or in Run n° 3 condition **With-Ignore**). For all the Runs, before reaching the last fitness plateau, we have periods of very rapid fitness growth induced by the acquisition of new skills to access and execute either entire linguistic instructions or just single linguistic labels. These periods are always followed by either long or short fitness plateaus characterized by rather small oscillations. Just by looking at the fitness curves, we can say that, at the end of the simulation, most of the best agents in both conditions looked capable of correctly solving the linguistic task. However, to estimate the effectiveness and robustness of some of the best evolved agents, with respect to the initial position of the arm, we postevaluated them for a larger number of trials.

A. First Postevaluation Test: Performances on Experienced and Nonexperienced Linguistic Instructions

In the first postevaluation test, the best five agents of each generation, from generation 4000 to generation 6000, of each evolutionary Run in both conditions, have been repeatedly postevaluated in each *experienced* and *nonexperienced* linguistic instruction. We decided to test the best five agents instead of the best one of each generation, because, during evolution, the agents have been ranked according to their fitness, which does not take into account the agent capability to access and execute *nonexperienced* linguistic instructions. Recall that *nonexperienced* linguistic instructions have not been presented during evolution. Thus, with respect to the capability to access and execute *nonexperienced* linguistic instructions, the best agent of each generation may not represent the most effective solution that appeared at each evolutionary time. Overall, 100 000 agents per condition have been postevaluated (i.e., five agents, times 2000 generations, times 10 Runs).

During this postevaluation test, each agent is required to execute 80 times each of the nine instructions [40 trials with the agents randomly initialized in the right initialization area and, 40 trials in the left one, see also Fig. 1(b)]. The position of the objects is also randomly varied as explained in Section III. In each trial k, an agent can be either successful or unsuccessful. It is successful if $F_k^{\mathrm{tot}} = 1$, otherwise it is unsuccessful (see (4), Section VI for details on F_k^{tot}). At the end of the postevaluation test, an agent capability to solve the linguistic and behavioral task is represented by nine scores, one for each linguistic instruction. Recall that each score ranges from 0 to 80, and it represents the number of times an agent is successful at the execution of the corresponding linguistic instruction.

It is worth noting that, the results of this test gave us a rather heterogeneous picture, with performances that, even for a single

agent, vary remarkably from one linguistic instruction to the other. We felt that readings and interpreting these data by only concentrating on general trends, it would have significantly impoverished the message or this research work. Therefore, we chose a way of representing the results which gives the reader a coherent and exhaustive, although a bit articulated, synthesis of what the postevaluated agents are capable of doing at the linguistic task. In particular, for each condition, the performances of the agents are compared with respect to four different subtasks. For each subtask, the comparison were accomplished by grouping the 100 000 agents in eleven different categories. We first describe what the subtasks are and then we explain the meaning of each category.

Subtask I takes into account only the seven scores recorded during the execution of the *experienced* linguistic instructions.

Subtask II takes into account the seven scores recorded during the execution of the *experienced* linguistic instructions plus the score recorded during the execution of the *nonexperienced* linguistic instruction MOVE Blue object.

Subtask III takes into account the seven scores recorded during the execution of the *experienced* linguistic instructions plus the score recorded during the execution of the *nonexperienced* linguistic instruction TOUCH Green object.

Subtask IV takes into account all the nine scores (i.e., seven of them for the *experienced* instructions plus two for the *nonexperienced* instructions).

For each subtask, the agents are allocated to one of eleven possible categories (from Cat^0 to Cat^{10}). For a given subtask, an agent is assigned to Cat^n with $n \in [0 \dots 10]$, if its lowest score among those considered for that particular subtask, is within the interval $(80(n-1/10) \dots 80(n/10)]$. Cat^0 comprises all agents that failed to complete a single trial out of 80 attempts on at least one of the instructions. The higher the category, the better the overall performance of the agent. For example, Cat^6 subsumes those agents for whom the lowest score among those considered in a given subtask is within the interval $(40, 48]$. Cat^7 subsumes those agents for whom the lowest score among those considered in a given subtask is within the interval (48,56], etc. Let's consider an agent whose performances at the postevaluation test are represented by the following nine scores vector (80, 80, 80, 80, 80, 80, 80, 52, 67), in which the first seven scores refer to the performances while executing *experienced* instructions, the eighth score refers to the performance while executing the *nonexperienced* instruction TOUCH Green, and the ninth score refers to the performance while executing the *nonexperienced* instruction MOVE Blue object. This agent would be assigned to the following categories: 1) category Cat^{10} as far as it concerns subtask I; 2) category Cat^9 as far as it concerns subtask II; and 3) category Cat^7 as far as it concerns subtask III and subtask IV.

Table II shows the number of postevaluated agents for each category and for each subtask. These results can be summarized in the following:

- for both conditions, more than half of the postevaluated agents (about 60% of the agents in **With-Indicate**, and about 66% of them in **With-Ignore**), are perfectly capable of accessing and executing the seven linguistic instruction *experienced* during evolution (see subtask I, Cat^{10}, condition **With-Indicate**, and **With-Ignore**). This is expected from what was previously observed in the fitness curves shown in Fig. 3.
- for both conditions, only a very small number of postevaluated agents is perfectly capable of accessing and executing all the *experienced* plus one single *nonexperienced* linguistic instruction, no matter which one of the two we consider (see Table II, subtask II, and III, Cat^{10}, condition **With-Indicate**, and **With-Ignore**). The great majority of the agents in subtask II and III completely fails to access and execute exactly the single *nonexperienced* linguistic instruction included in the corresponding subtask. This has been confirmed by further checks on the data. However, it can also be inferred from the fact that the same agents that are in Cat^{10} for subtask I tend to be in Cat^0 for subtasks II and III.
- for both conditions, only a tiny fraction of the postevaluated agents is perfectly capable of accessing and executing both the *experienced* and *nonexperienced* linguistic instructions (see Table II, subtask IV, Cat^{10}, **With-Indicate**, and **With-Ignore**).

From these results, it clearly emerges that only a tiny fraction of the postevaluated agents is capable of accessing and executing all the linguistic instructions, independently from the initial position of the arm. However, since the number of agents in condition **With-Indicate**, Cat^{10}, subtask II, III, and IV, is

TABLE II
RESULTS OF POSTEVALUATION TESTS SHOWING, FOR EACH EVOLUTIONARY CONDITION, THE NUMBER OF AGENTS FOR EACH PERFORMANCE CATEGORY AND FOR EACH SUBTASK. THE TOTAL NUMBER OF POSTEVALUATED AGENTS PER CONDITION IS 100 000 (I.E., FIVE AGENTS, TIMES 2000 GENERATIONS, TIMES 10 RUNS)

	With-Indicate			
	Subtask I	Subtask II	Subtask III	Subtask IV
Cat^0	9408	75200	70787	90263
Cat^1	1545	3962	5840	3313
Cat^2	578	1252	2477	1092
Cat^3	823	1314	2174	889
Cat^4	1458	1703	2016	939
Cat^5	3558	2161	8217	2430
Cat^6	2483	2004	1493	346
Cat^7	2780	2061	922	197
Cat^8	5020	1668	957	174
Cat^9	12116	1906	995	135
Cat^{10}	60231	6769	4122	222
Total	100000	100000	100000	100000
	With-Ignore			
Cat^0	8127	87238	92457	98516
Cat^1	15	3502	2439	643
Cat^2	26	1220	1069	218
Cat^3	102	989	1021	220
Cat^4	275	890	928	160
Cat^5	15733	3836	1363	178
Cat^6	451	382	208	15
Cat^7	822	215	145	6
Cat^8	2049	231	141	10
Cat^9	6107	302	121	8
Cat^{10}	66293	1195	108	26
Total	100000	100000	100000	100000

significantly different from the number of agents in condition **With-Ignore**, Cat^{10}, subtask II, III, and IV (pairwise Wilcoxon test with 99% confidence interval), we conclude that condition **With-Indicate** facilitates the evolution of agents capable of accessing and executing both *experienced* and *nonexperienced* linguistic instructions. In other words, evolutionary pressures to evolve a behavioral repertoire to execute the INDICATE behavior seem to facilitate the development of compositional semantics. In the next Section, we will further investigate this issue and present a closer look at what makes condition **With-Indicate** more suitable to the evolution of compositional semantic structures.

Obviously, it is important to emphasize the fact that the evolutionary conditions detailed in previous Sections, and in particular those in condition **With-Indicate**, generate the neural mechanisms required by the agents to go beyond what was experienced during evolution. Nevertheless, the fact remains that in either condition, the agents capable of generalizing their skills are only a tiny fraction of the agents capable of successfully accomplishing the evolutionary task. This can be explained by the fact that: 1) evolution only seldom produced agents fully capable of generalizing their skills; and 2) the selective process does not differentiate between compositional and noncompositional agents since they tend to produce equally good performance with respect to the conditions in which they are evaluated. We noticed that agents capable of generalizing appear only in six Runs out of ten, and they are never more than one or two per generation.[2] When they appear, they generally have the highest fitness recorded at that particular generation, which almost always is the highest possible fitness. However, they tend to appear when there are already many more agents with the same fitness in the population that are nevertheless not capable of generalizing their linguistic and behavioral skills to *nonexperienced* linguistic instructions. The selection mechanism, which can not distinguish on the basis of the fitness alone, agents capable of generalizing from those not capable of generalizing, tends to favor the latter, to the detriment of the former, simply because the latter are more frequent in the population.

A final point of minor significance is that generalization capabilities with respect to the MOVE Blue object instruction are more frequent than that with respect to the TOUCH Green object instruction. That is, for both conditions, the number of agents in Cat^{10} subtask II is significantly different from the number of agents in Cat^{10} subtask III (pairwise Wilcoxon test with 99% confidence interval). Although we have no empirical explanation for this finding, we know that the action MOVE, which requires the agents to rotate both arms around their joints, is an action that, in evolutionary terms, appears earlier than the capability to TOUCH an object, which requires the agents to stop rotating both arms. At the beginning of the evolution, when the agents' linguistic and behavioral skills are rather simple, it pays more to be able to rotate the arms in order to approach the target objects, rather than to be able to stop a not existing yet rotation of the arms. This evolutionary progression of the behavioral skills may explain why the *nonexperienced* instruction which requires to MOVE a target object turns out to be more easily accessible and executable than the *nonexperienced* instruction that requires to TOUCH a target object.

[2]Data not shown in the paper can be found at http://laral.istc.cnr.it/esm/tuci-etal-IEEE_TAMD2010/.

B. Compositionality: Operational Principles

What kind of operational principles do agents employ to be able to access and execute both *experienced* and *nonexperienced* instructions? What are the mechanisms underpinning compositional semantics? By visually inspecting the behavior of some of the agents, we notice that, contrary to the behavior of the agents evolved in condition **With-Ignore**, the behavior of compositional agents evolved in condition **With-Indicate** is the result of the combination of two types of elementary behavior: an "INDICATE Red object" or "INDICATE Green object," or "INDICATE Blue object" behavior produced during the first phase of the trial, eventually followed by a "TOUCH" or "MOVE" behavior, in the second phase of the trial. During the first phase of the trial, regardless of the action to be performed on the object, the agents search the target object by rotating S^1 in order to reduce the angular distance between the target object and S^1, keeping S^2 bent as at start until the target object falls into the agent visual field. During the second phase of the trial, regardless of the target object, the agents rotate S^2 without moving S^1 if TOUCH is required. They rotate S^2 until this segment collides with the target object and then they start rotating S^1 again if MOVE is required. They keep S^1 pointing to the object and S^2 fully bent as at start if INDICATE is required. This qualitative analysis of the behavior of compositional agents suggests that the agents have developed behavioral skills that, being independent from the particular nature of the linguistic instructions in which they are employed, can be used in contexts already experienced as well as in context not experienced during training.

From this observation, we hypothesized that compositional semantics is underpinned by simple mechanisms by which, during the first part of the trial, the agents regulate their actions on the basis of the object-label and not on the basis of the action-label, and vice-versa, during the second part of the trial. This quite intuitive hypothesis suggests that, in any given trial, there may be a first temporal phase, which starts right at the beginning of the trial, in which agents access the part of the linguistic instruction that defines the target object (i.e., the object-label) and act in order to execute the appropriate search behavior. During this phase, the other part of the linguistic instruction (i.e., the action-label) should not influence the agent's behavior. The first phase would be followed by a second one, which begins roughly when the target object is visually found. In the second phase, the agents regulate their behavior on the basis of the action-label only (i.e., the object-label does not have any influence) in case the instruction is TOUCH or MOVE. In the case of INDICATE, instead, the agents keep producing the same behavior during the entire trial. On this account of compositionality, linguistic instructions not experienced during training (i.e., MOVE Blue object, TOUCH Green object), would be:

- accessed by exploiting the capability to extract from a *nonexperienced* instruction already experienced linguistic labels (i.e., TOUCH, MOVE, Blue object, and Green object).

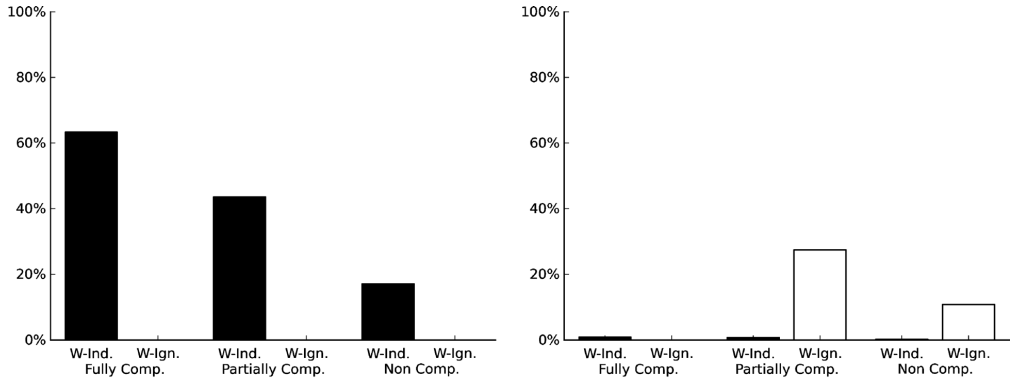


Fig. 4. Graphs showing the results of the (a) *action–transition test*; (b) *object–transition test*. In both graphs, each bar indicates the percentage of agents that managed to obtain a success rate higher than 75% in all possible transitions of the corresponding test. Black bars refer to the agents evolved in condition **With-Indicate**, white bars refer to the agents evolved in condition **With-Ignore**. See the text for the definition of *fully-compositional*, *partially-compositional*, and *noncompositional* agents.

- executed by calling upon known elementary behaviors associated to or triggered by one or the other linguistic label.

In what remains of this Section, we show the results of two postevaluation tests designed in order to verify whether the agents temporally and functionally decompose the linguistic and behavioral task into two sequential phases as suggested by our hypothesis. These tests are referred to as the *action–transition test* and the *object–transition test*. Both tests follow a similar logic. In the *action–transition test*, the action-label is changed during the course of a trial, while in the *object–transition test*, the object-label is changed during the course of a trial. In both tests, the change takes place in a single time step randomly chosen within a 10 time steps interval which starts at the time when the target object falls within an agent visual field. Based on our hypothesis, agents equipped with compositional semantics are expected to execute the second given action–label and neglect[3] the first given one, at the *action–transition test*. This is because the first given action-label is experienced during the first phase of a trial, when the attention of the agents should be focused on the object-label. At the *object–transition test*, these agents are expected to neglect the second given object-label. This is because this object-label is experienced during a time in which the agents already see the first given target. Consequently, they should pay attention only to the action-label.

The *action–transition test* and the *object–transition test* have been run on a pool of agents selected on their results at the first postevaluation test (see Section VII.A). In particular, for each evolutionary condition (i.e., **With-Indicate**, and **With-Ignore**), we chose the agents that proved to be more than 75% successful at executing each *experienced* instruction. For the purposes of these tests, these agents have been further selected, and the following three categories have been created: 1) *noncompositional* agents, referring to those agents that, at the first postevaluation test, proved to be less than 10% successful at executing each of the *nonexperienced* instructions; 2) *partially-compositional* agents, referring to those agents that, at the first postevaluation test, proved to be more than 75% successful at executing only one of the two *nonexperienced* instructions, and less than 10% successful at executing the other *nonexperienced* instructions; and 3) *fully-compositional* agents, referring to those agents that, at the first postevaluation test, proved to be more than 75% successful at executing each of the *nonexperienced* instructions.

For both tests, the agents are evaluated 80 times (i.e., 80 trials) on each transition. In half of the trials, the agents are randomly initialize in the right, and in half of the trials, in the left initialization area. In each trial k, an agent can either succeed, if at the end of the trial $F_k^{\text{tot}} = 1$, or fail, if $F_k^{\text{tot}} < 1$. Following the logic of each test, the fitness components F_k^1, F_k^2, and F_k^3 are updated with respect to the execution of the second given action-label on the current target object, in the *action–transition test*, and with respect to the execution of the current action-label on the first given target object, in the *object–transition test*. For both tests, an agent's overall performance on each specific transition is considered a success if the agent successfully executes the transition in more than 60 out of 80 trials (i.e., 75% success rate). Since both tests are indiscriminately done on *noncompositional*, *partially-compositional*, and *fully-compositional* agents, we removed from the two sets of possible transitions, those which, assuming our hypothesis holds, require a response that *noncompositional*, and *partially-compositional* agents are not capable of performing. That is, we remove those transitions which require a MOVE Blue object, or a TOUCH Green object response.[4]

Fig. 4(a) and (b) show the results of the *action–transition test* and of the *object–transition test*, respectively. In both graphs, each bar indicates the percentage of agents that managed to obtain a success rate higher than 75% in all possible transitions of the corresponding test. Black bars refer to the agents evolved in

[3]In this Section, we often take an anthropomorphic stance, by talking about agents that attend or neglect linguistic labels. This is purely for ease of exposition. It is not our intention to claim that the agents are cognitively rich enough to intentionally attend or neglect sensory stimuli.

[4]In particular, in the *action–transition test*, the transitions experienced by the agents are those in which the second given action-label in combination with the object-label does not produce a *nonexperienced* instruction. Similarly, in the *object–transition test*, the transitions experienced by the agents are those in which the first given object-label in combination with the action-label does not produce a *nonexperienced* instruction

condition **With-Indicate**, white bars refer to the agents evolved in condition **With-Ignore**. Before commenting the results, the reader should be aware of the following. These are quite severe tests since they demands a high success rate on part of the agents on each experienced transition. If our hypothesis on the mechanisms underpinning compositionality holds, we expect *noncompositional* and *partially-compositional* agents to be very bad at least in one of the experienced transitions. This is because we assume that the test can be successfully performed only by agents possessing the capability to functionally and temporally decompose the linguistic and behavioral task into two sequential phases, and that this capability can only be found in *fully-compositional* agents. However, the agents may not need to fully decompose every single trial into two sequential phases in order to be able to successfully access and execute *nonexperienced* instructions. In this sense, the test may demand more than what is required to be capable of behavioral and linguistic generalization. Moreover, in these tests the agents' performance is influenced by whether the label change takes place exactly at the time when the agents switch the focus of their attention from the object–label to the action–label. For methodological convenience, we treated all the agents in the same way, by arbitrarily making this switch in a single time step randomly located in a 10 time steps interval that starts when the agents see the target object. Nevertheless, this may not fully comply with each agent's own strategy, causing failure even in those agents that can functionally and temporally decompose the task.

In spite of these limitations, these graphs tell us several important things. We first concentrate on the results of the *action–transition test*. Fig. 4(a) indicates that the majority of *fully-compositional* agents evolved in condition **With-Indicate**, relies on strategies in which the action–label does not influence the agents' behavior during the first phase of the task [see Fig. 4(a), black bar on the left]. This suggests that the capability to neglect the action-label while searching for the target object is associated with the presence of compositional semantic structures, since it tends to be observed in *fully-compositional* agents. However, some of the *partially-compositional* and *noncompositional* agents in condition **With-Indicate** proved also capable of accomplishing their task without failing in any transition of the *action–transition test* [see Fig. 4(a), central and right black bars]. Thus, the first conclusion we draw is that neglecting the action–label while reaching the target object is not sufficient to attain compositionality, since it does not allow those *partially-compositional* and *noncompositional* agents that possess it to access and execute *nonexperienced* instructions.

Fig. 4(a) also shows that the capability to cope with the action-label change is completely absent in the agents evolved in condition **With-Ignore**. This result seems to suggest that the significant differences, illustrated in the previous Section, between the two evolutionary conditions in the generation of agents capable of accessing and executing *nonexperienced* linguistic instructions, could be explained by the fact that solutions based on the combination of independent elementary behaviors are more rare in the **With-Ignore** condition. Thus, we further conclude that the condition **With-Indicate** seems to contain the evolutionary pressures that facilitate the emergence of compositionality by indirectly favoring those agents whose behavior is not influenced by the action-label while they reach the target object.

Fig. 4(b), which refers to the *object–transition test*, tell us that the capability to neglect the object-label during the second phase of a trial, when the target object is already within an agent's visual field, is completely absent in agents evolved in condition **With-Indicate**, and in particular is completely absent in *fully-compositional* agents. Only some of the *partially-compositional* and of the *noncompositional* agents evolved in condition **With-Ignore** seem to be able to cope with the object-label change [see Fig. 4(b), central and right white bars]. How do we explain these results? As far as it concerns the unexpected failure of the *fully-compositional* agents evolved in condition **With-Indicate**, we found out that, contrary to what hypothesized by us, the agents use the object-label during the second phase of the task to keep the target object within their visual field. We observed that, when the object-label does not match what is visually perceived, *fully-compositional*, *partially-compositional*, and *noncompositional* agents perform a search behavior, loosing visual contact with the object indicated by the first given object-label. Thus, the object-label influences the agents' behavior during both the first and second phase of a trial, by triggering the agents' response of searching and orienting toward the appropriate object. As far as it concerns the performance of the agents evolved in condition **With-Ignore**, we think that their successes at the *object–transition test* can be explained by considering the evolutionary circumstances in which they evolved. In particular, the action IGNORE can be accomplished by executing a common act for all the objects. Behavioral inspections have indeed demonstrated that *partially-compositional* and *noncompositional* agents evolved in condition **With-Ignore** and capable of coping with the object–label change, once required to IGNORE an object simply do not move at all from their position. This is a strategy which can be successfully applied to execute the action IGNORE regardless of the target object. This may have facilitated the emergence of mechanisms to be able to neglect the object-label while executing the required action. However, this is speculative and further analyses are required to test it.

Overall, these tests indicate that in *fully-compositional* agents obtained in condition **With-Indicate**, the "INDICATE Red object," "INDICATE Blue object," and "INDICATE Green object" behaviors are executed during the entire trial, as demonstrated by the fact that the agents are able to search for a new object and then keep indicating it when the object-label is modified during the second phase of the trial. The execution of the "INDICATE" behavior during the second phase of the trial is not apparent in normal condition (i.e., when the position or the color of the objects do not change) simply because the execution of this behavior do not produce any movement. Thus, during the second phase of the trial, when the action label is "INDICATE," agents keep producing the same behavior. When the action label is "TOUCH" or "MOVE," the agents perform the corresponding elementary behavior that operates in parallel with the "INDICATE" behavior. The key mechanism that enables compositionality, therefore, is the fact that the action–label does not affect the agents behavior during the first part of the trial. In other words, "TOUCH" and "MOVE" behaviors con-

stitute independent behavioral units realized through the execution of the same sequence of micro-actions irrespectively from the object–label. Moreover, we can now state that a different training bears upon the development of the required mechanisms for compositional semantics, and that condition **With-Indicate** facilitates the emergence of compositionality by favoring the emergence of the functional independence of the action-label from the behavioral experience of searching for the target object.

Indeed, by looking at the phylogeny of *fully-compositional* and *partially-compositional* agents in condition **With-Indicate**, we notice that in early stages of their evolutionary history, one of the first behavioral skill to appear is indeed related to the capability of these agents to systematically reduce the angular distance between S^1 and the target object regardless of what type of action the current linguistic instruction is demanding. For example, the ancestors of *fully-compositional* agents, when required to MOVE or to TOUCH an object, they successfully bring S^1 in correspondence of the target object, and they keep S^2 bent until the end of the trial, by systematically failing to execute the action MOVE and TOUCH. In other words, these agents proved to be capable of accessing the linguistic label that defines the object without being able to appropriately execute the linguistic label that defines the TOUCH and MOVE actions. The ability to handle these type of actions is developed later. This can be considered a further evidence that, since the early generation of evolution in condition **With-Indicate**, *fully-compositional* and *partially-compositional* agents learn to decompose the trial into two parts, in the first one of which their behavior is entirely triggered by the object-label. It is important to note that the early appearance of the capability to decompose the task into two parts is not enforced by any means by the design of the fitness function, it emerges through the dynamics of evolution, and it is facilitated in condition **With-Indicate** by the presence of the instruction INDICATE. However, in the absence of further tests, we can not exclude that these phenomena are induced by design constraints, such as the fact that the segment S^1 and the vision system are bound together. This is because, this particular constraint makes it impossible for an agent to develop a visual search strategy without concurrently acting as required by the instruction INDICATE.

VIII. Discussion: Perspectives for Research on Child Language Acquisition

Computational approaches to language learning are an intensely researched topic in several disciplines (for recent reviews, cf. [20]–[22]). As yet, however, there is still a marked gap between language learning research in cognitive robotics on the one hand and language acquisition studies in computational linguistics on the other. One reason for this is the different thrust of typical research in the two disciplines: in robotics, the focus is commonly on semantic issues to do with the grounding of individual linguistic symbols in agents' sensory–motor experience [42]. In computational linguistics, the focus is usually on structural issues to do with the induction of complex grammars from unrestricted text [43], [44]. In a nutshell, roboticists tend to concentrate on words as carriers of meaning (but neglect their combinatorial properties), while linguists tend to concentrate on their grammar (but neglect their meanings).

Given this apparent opposition, it is interesting to note that a currently influential theory of child language acquisition assumes both a phenomenological continuum and a developmental connection between these two seemingly complementary learning targets (i.e., meaningful "words" and meaningless "rules" in traditional terminology). In usage-based models of language learning, children are assumed to acquire linguistic "rules" (i.e., grammatical categories and constructional patterns thereof) through piecemeal abstractions over utterance-length concrete "words" (i.e., unanalyzed holophrastic strings like "there+you+go" and "look+at+this" that are associated with a holistic communicative intention, see [9]). Learners' discovery of the internal structure of these units, coupled with the realization that the segmented building blocks can be productively recombined within the abstracted constructional patterns, marks the crucial transition from finite lexicons to open-ended grammars. From this perspective, the above experiment is therefore concerned with the emergence of a genuine breakthrough on the way to language.

Needless to say, both the learning target and the learning architecture are substantially less complex here. However, most computational models of language acquisition do not purport to provide an accurate representation of the precise learning mechanisms and processes at work in human children. Rather, the more modest aim is usually to show that it is possible to solve a given task through learning at all (i.e., without innate domain-specific biases). In this way, computational models have made an important contribution to the debate over language learnability, innateness and the purported "poverty of the stimulus" (e.g., [45] and [46]). However, none of the models in these debates is grounded in the way that human children's internal representation of language is. In other words, such research has focused on the combinatorial dimension of language alone, but has ignored the additional challenge of linking linguistic structures to the embodied conceptualizations that constitute their meanings. The present study takes steps towards closing this gap, and several of its findings can indeed be related to similar observations made in empirical studies of child language learning.

To better appreciate these connections, it will be helpful to translate aspects of the design into the terminology of usage-based models of child language acquisition. Agents' capacity to correctly access and execute a *nonexperienced* linguistic instruction corresponds to their acquisition of an "item-based construction," for example, [move N] in the sense of [9]. As the term "item-based" implies, the generalizations that child language learners have acquired at this developmental stage do not apply across the board. For instance, they may begin to use grammatical marking on some verbs but not on others, indicating that the more inclusive generalization that both items belong to the same overall category has not yet been formed. Empirical evidence for such item-specific effects in early language acquisition is abundant (cf. [9]), and the theoretical vision of a transition from holophrastic units over networks of item-specific "islands" to ever more schematic grammars has also received support from different computational simulations of (nongrounded)

language learning [47]. From this perspective, agents' differential performance on the two *nonexperienced* instructions in the present experiment does not come as a surprise: also in child language acquisition, the transition from holophrases to compositional grammars is not instantaneous.

Similarly, also the second major finding of this study, that is the significant effect of learning condition (**With-Indicate** versus **With-Ignore**) on agents' generalization performance readily translates into a concept of usage-based models of child language learning: if the above assumptions about what makes the behavior INDICATE more similar to MOVE and TOUCH than IGNORE are plausible, agents' poorer generalization performance in condition **With-Ignore** would be said to be the outcome of a lower cue consistency (i.e., regularity of form-function mapping) of the category "Verb" in this condition. Furthermore, since such constellations of inconsistency, competition and syncretism are in fact taken to be the norm in natural language processing and learning, a look to usage-based acquisition models in linguistics could also suggest certain useful extensions of the present approach that might be worthwhile to explore in future work (e.g., studying agents' generalization performance across more than one construction, with or without semantic similarity between actions and/or referents, with balanced or statistically skewed training input, etc.). In other words, we will investigate the characteristics that favor the emergence of compositional solutions (i.e., that ensure behavioral generalization) and/or that reduce the chance to converge on noncompositional solutions. We will also investigate the possibility to scale the model with respect to the number and the complexity of the linguistic/behavioral repertoire of the agent.

IX. Conclusion

In this paper, we described a robotic model that allows a simulated robot to interact with three colored objects (a Red, a Green, and a Blue object) located in its peripersonal space by executing three actions (INDICATE, TOUCH, and MOVE) during a series of trials. In each trial, the agent receives as input a linguistic instruction and is rewarded for the ability to exhibit the corresponding behavior. The results of this study show that dynamical neural networks designed by artificial evolution allow the robot to access and correctly execute the linguistic instructions formed by all the possible combinations of the three action-labels and the three object-labels with the exception of the instructions "TOUCH Green object" and "MOVE Red object," which are *nonexperienced* during training. Postevaluation tests showed that some of the evolved agents generalize their linguistic and behavioral skills by responding to the two *nonexperienced* instructions with the production of the appropriate behaviors.

Our study shows that behavioral and linguistic competences can coevolve in a single nonmodularized neural structure in which the semantics is fully grounded in the sensory–motor capabilities of the agents and fully integrated with the neural mechanisms that underpin the agent's behavioral repertoire. Owe to the use of artificial evolution, we leave the agents free to determine how to achieve the goals associated to each linguistic instruction. This allows the agents to oreganos their behavioral skills in ways that facilitate the development of compositionality thus enabling the possibility to display a generalization ability at the level of behaviors (i.e., the ability to spontaneously produce behaviors in circumstances that have not been encountered or rewarded during training).

The comparison between two experimental conditions, in one of which the action–label INDICATE is substituted with the action–label IGNORE, shows that the composition of the behavioral set significantly influences the development of solutions that generalize to *nonexperienced* instructions. Only individuals evolved in condition **With-Indicate** are characterized by a particularly successful linguistic and behavioral organization based on the decomposition of the task into two phases, each of which can be associated with the execution of an elementary behavior. In the first phase only the object-label bears upon the agents' behavior by triggering the object search strategy. In the second phase, both the object-label and the action-label determine the agents' response. In particular, the object-label keeps an agent eliciting the same behavior produced during the first phase (i.e., the agent keeps on searching/pointing the target object with the first segment of its arm). At the same time, the action-label triggers a different behavior that consists in bending the second segment of the arm so to touch or move the object. The capability to decompose the task into two sequential phases as described above, and the use of elementary behaviors employed in different circumstances, are features that, although not sufficient *per se* to explain compositional semantics, they certainly facilitate its evolution.

The use of elementary behavioral skills to generate instructions denoting complex actions resembles the process described in [37], in which the ability to execute more complex linguistic commands, such as GRAB, is acquired by associating two or more previously acquired elementary behaviors (e.g., CLOSE-LEFT-ARM and CLOSE-RIGHT-ARM). However, in [37], the relation between complex and elementary behaviors is established through explicit teaching (i.e., through linguistic input such as: GRAB is CLOSE-LEFT-ARM and CLOSE-RIGHT-ARM). By contrast, in the experiments reported in this paper, behavioral decomposition emerge as a side effect of the need to acquire the ability to execute several related linguistic commands. Moreover, the way in which the agents accomplished the required functionality (i.e., by combining in sequence or in parallel relatively independent behavioral units) represents an important prerequisite for the emergence of compositionality. Therefore, leaving the agents as free as possible to organize how they produce the required skills might be advantageous since it might allow them to decompose the problem in a way that maximize skills reuse. This in turn implies that methods such as the evolutionary method that rewards the agent on the basis of the ability to achieve a given functionality without specifying in details the behavioral that should be produced might be advantageous with respect to alternative methods in that respect.

The results of our postevaluation analyses also suggests us that there are further distinctive operational principles underpinning compositionality, other than those considered in this work, that are most probably related to the structural and functional characteristics of the agents' neural controller. In future work, we will specifically target these principles, trying to provide a

clear description of their nature. Moreover, we mentioned that compositional agents tend to appear very rarely during evolution. It is our intention to work on the characteristics of the task to identify the elements that bear upon the evolutionary origins of agents equipped with compositional semantic structures. With respect to this issue, we think that it may be worth to vary linguistic features and behavioral aspects of the task. For example, in this simulation, the objects have fixed positions with respect to the agent (i.e., Red object on the left, Green object in front, and Blue object on the right of the agent). We wonder whether the necessity to evolved more robust exploration strategies, induced by the variability of the object position relative to the agent, facilitates or hinders the development of compositional structures. Moreover, we are interested in studying whether the use of more cognitively plausible coding schemes, in which the labels are perceived by the agent in a sequential order and just for a short interval of time, bears upon the emergence of compositional semantics. We are also interested in studying whether the development, during training, of a wider and more heterogeneous behavioral repertoire facilitates the emergence of more robust generalization capabilities.

ACKNOWLEDGMENT

The authors would like to thank C. Burani *et al.* at the Laboratory of Autonomous Robotics and Artificial Life (LARAL) for stimulating discussions and feedback during the preparation of this paper.

REFERENCES

[1] G. Rizzolatti and M. Arbib, "Language within our grasp," *Trends Neurosci.*, vol. 21, pp. 188–194, 1998.
[2] S. Cappa and D. Perani, "The neural correlates of noun and verb processing," *J. Neuroling.*, vol. 16, no. 2–3, pp. 183–189, 2003.
[3] A. Glenberg and M. Kaschak, "Grounding language in action," *Psychonomic Bulletin Rev.*, vol. 9, pp. 558–565, 2002.
[4] F. Pulvermuller, *The Neuroscience of Language. On Brain Circuits of Words and Serial Order*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[5] V. Gallese, "Mirror neurons and the social nature of language: The neural exploitation hypothesis," *Social Neurosci.*, vol. 3, pp. 317–333, 2008.
[6] G. Buccino, T. Riggio, G. Mellia, F. Binkofski, V. Gallese, and G. Rizzolatti, "Listening to action-related sentences modulates the activity of the motor system: A combined tms and behavioral study," *Cogn. Brain Res.*, vol. 24, pp. 355–363, 2005.
[7] M. Bowerman and S. Levinson, *Language Acquisition and Conceptual Development*. Cambridge: Cambridge Univ. Press, 2001.
[8] B. MacWhinney, "The emergence of linguistic form in time," *Connect. Sci.*, vol. 17, no. 3–4, pp. 191–211, 2005.
[9] M. Tomasello, *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Cambridge, MA: Harvard Univ. Press, 2003.
[10] G. Lakoff, *Women, Fire, and Dangerous Things: What Categories Reveal About the Mind*. Chicago, IL: Univ. Chicago Press, 1987.
[11] R. Langacker, *Foundations of Cognitive Grammar*. Stanford, CA: Stanford Univ. Press, 1987.
[12] A. Goldberg, *Constructions at Work: The Nature of Generalization in Language*. London, U.K.: Oxford Univ. Press, 2006.
[13] R. Langacker, *Cognitive Grammar: A Basic Introduction*. Oxford, U.K.: Oxford Univ. Press, 2008.
[14] P. Dominey, "From holophrases to abstract grammatical constructions: Insights from simulation studies," in *Constructions in Acquisition*, E. Clark and B. Kelly, Eds. Stanford: CSLI Publications, 2006, pp. 137–162.
[15] E. Hutchins and C. Johnson, "Modelling the emergence of language as an embodied collective cognitive activity," *Topics Cogn. Sci.*, vol. 1, pp. 523–546, 2009.
[16] S. Wermter, M. Page, M. Knowles, V. Gallese, F. Pulvermller, and J. Taylor, "Multimodal communication in animals, humans and robots: An introduction to perspectives in brain-inspired informatics," *Neural Netw.*, vol. 22, no. 2, pp. 111–115, 2009.
[17] A. Cangelosi, G. Metta, G. Sagerer, S. Nolfi, C. Nehaniv, K. Fischer, J. Tani, T. Belpaeme, G. Sandini, F. Nori, L. Fadiga, B. Wrede, K. Rohlfing, E. Tuci, K. Dautenhahn, J. Saunders, and A. Zeschel, "Integration of action and language knowledge: A roadmap for developmental robotics," *IEEE Trans. Autonom. Mental Develop.*, vol. 2, no. 3, pp. 1–28, Sep. 2010.
[18] R. W. Langacker, "A dynamic usage-based model," in *Usage-Based Models of Language*, M. Barlow and S. Kemmer, Eds. Stanford: CSLI Publications, 2000, pp. 1–63.
[19] A. Goldberg, "Constructions work," *Cognitive Linguistics*, vol. 20, no. 1, pp. 201–224, 2009.
[20] J. Elman, "Computational approaches to language acquisition," in *Encyclopedia of Language and Linguistics*, K. Brown, Ed., 2nd ed. Oxford, U.K.: Elsevier, 2006, vol. 2, pp. 726–732.
[21] F. Kaplan, P. Oudeyer, and B. Bergen, "Computational models in the debate over language learnability," *Inf. Child Develop.*, vol. 17, no. 1, pp. 55–80, 2008.
[22] B. MacWhinney, "Computational models of child language learning: An introduction," *J. Child Lang.*, vol. 37, pp. 477–485, 2010.
[23] Y. Sugita and J. Tani, "Learning semantic combinatoriality from the intercation between linguistic and behvioral processes," *Adapt. Behav.*, vol. 13, no. 1, pp. 33–52, 2005.
[24] Y. Sugita and J. Tani, "Acquiring a functionally compositional system of goal-directed actions of a simulated agent," in *Proc. 10th Int. Conf. Simul. Adapt. Behav. (SAB2008)*, M. Asada, J. Hallam, J.-A. Meyer, and J. Tani, Eds., Berlin, Germany, 2008, pp. 331–341.
[25] H. Arie, T. Endo, S. Jeong, M. Lee, S. Sugano, and J. Tani, "Integrative learning between language and action: A neuro-robotics experiment," in *Proc. 20th Int. Conf. Neural Netw.*, Berlin, Germany, 2010.
[26] J. Fodor and Z. Phylysyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition*, vol. 28, pp. 3–71, 1988.
[27] S. Harnard, "The symbol grounding problem," *Physica D*, vol. 42, pp. 335–346, 1990.
[28] T. Van Gelder, "Compositionality: A connectionist variation on a classic theme," *Cognition*, vol. 14, pp. 355–384, 1990.
[29] J. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.
[30] *Simulating the Evolution of Language*, A. Cangelosi and D. Parisi, Eds. New York: Springer-Verlag, 2002.
[31] *Evolution of Communication and Language in Embodied Agents*, S. Nolfi and M. Mirolli, Eds. Berlin, Germany: Springer-Verlag, 2010.
[32] L. Steels, "Experiments on the emergence of human communication," *Trends Cogn. Sci.*, vol. 10, no. 8, pp. 347–349, 2006.
[33] E. Tuci, "An investigation of the evolutionary origin of reciprocal communication using simulated autonomous agents," *Biol. Cybernet.*, vol. 101, no. 3, pp. 183–199, 2009.
[34] P. Dominey, A. Mallet, and E. Y. E. , "Real-time spoken-language programming for cooperative interaction with a humanoid apprentice," *Int. J. Human. Robot.*, vol. 6, no. 2, pp. 147–171, 2009.
[35] P. Dominey and F. W. F. , "The basis of shared intentions in human and robot cognition," New Ideas in Psychology 2010, in Pres.
[36] J. Weng, "Developmental robotics: Theory and experiments," *Int. J. Human. Robot.*, vol. 1, no. 2, pp. 199–236, 2004.
[37] A. Cangelosi and T. Riga, "An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots," *Cogn. Sci.*, vol. 30, no. 4, pp. 673–689, 2006.
[38] G. Sandini, G. Metta, and D. Vernon, "The icub cognitive humanoid robot: An open-system research platform for inactive cognition," in *50 Years of Artificial Intelligence*, M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 358–369.
[39] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2010.
[40] R. D. Beer and J. C. Gallagher, "Evolving dynamic neural networks for adaptive behavior," *Adapt. Behav.*, vol. 1, no. 1, pp. 91–122, 1992.
[41] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[42] E. A. Di Paolo, "Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents," *Adapt. Behav.*, vol. 8, no. 1, pp. 27–48, 2000.

[43] N. Chater and C. Manning, "Probabilistic models of language processing and acquisition," *Trends Cogn. Sci.*, vol. 10, no. 7, pp. 335–344, 2006.
[44] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised learning of natural languages," in *Proc. Nat. Acad. Sci.*, 2005, vol. 102, pp. 11 629–11 634.
[45] F. Reali and M. Christansen, "Uncovering the richness of the stimulus: Structure dependence and indirect statistical evidence," *Cogn. Sci.*, vol. 29, pp. 1007–1028, 2005.
[46] J. Lewis and J. Elman, "Learnability and the statistical structure of language: Poverty of stimulus arguments revisited," in *Proc. 26th Annu. Boston Univ. Conf. Lang. Develop.*, 2001.
[47] W. Morris, G. W. Cottrell, and J. Elman, "A connectionist simulation of the empirical acquisition of grammatical relations," in *Hybrid Neural Symbolic Integr.*, S. Wermter and R. Sun, Eds. Berlin, GE: Springer-Verlag, 2000.

Elio Tuci received the M.Sc. (Laurea) degree in experimental psychology from "La Sapienza" University, Rome, Italy, in 1996, and the Ph.D. degree in computer science and artificial intelligence from University of Sussex, Sussex, U.K., in 2004.

He is currently a Lecturer in Developmental Robotics in the Department of Computer Science, Aberystwyth University, Aberystwyth, U.K. His research interests concern the development of real and simulated embodied agents to look at scientific questions related to the mechanisms and/or the evolutionary origins of individual and social behavior.

Tomassino Ferrauto received the M.Sc. degree in engineering from the University of L'Aquila, L'Aquila, Italy. He is currently working toward the Ph.D. degree at the University of Plymouth, Plymouth, U.K., working under the supervision of Prof. A. Cangelosi and Dr. S. Nolfi.

His main research interests are within the domain of evolutionary robotics, active perception, and language and behavior codevelopment in robots.

Arne Zeschel received the Staatsexamen degree at the University of Hannover, Hannover, U.K., in 2001 and the Ph.D. degree in English linguistics at the University of Bremen, Bremen, Germany, in 2007.

He is currently working as a Postdoctoral Researcher on construction-based approaches to child language acquisition at the University of Southern Denmark, Sønderborg, Denmark. His main research interests are cognitive linguistics, corpus linguistics, and usage-based linguistic model building.

Gianluca Massera received the M.Sc. degree in computer science from the University of Rome "Sapienza," Rome, Italy. He is currently working toward the Ph.D. degree at the Plymouth University, Plymouth, U.K., working under the supervision of Prof. A. Cangelosi and Dr. S. Nolfi.

His research interests are within the domain of evolutionary robotics, active perception, and sensory–motor coordination in artificial arms.

Stefano Nolfi received the M.A. degree in literature and philosophy from the University of Rome "Sapienza," Rome, Italy.

He is currently the Research Director at the Institute of Cognitive Sciences and Technologies of the Italian National Research Council (ISTC-CNR) and Head of the Laboratory of Autonomous Robots and Artificial Life, Rome, Italy. His research activities focus on embodied cognition, adaptive behavior, autonomous robotics, and complex Systems. He authored or coauthored more than 130 scientific publications and a book on evolutionary robotics published by MIT Press.

Original Paper

Adaptive Behavior

Adaptive Behavior
2014, Vol. 22(4) 255–265

DOI: 10.1177/1059712314536909
adb.sagepub.com
SAGE

Designing adaptive humanoid robots through the FARSA open-source framework

Gianluca Massera[1], Tomassino Ferrauto[1], Onofrio Gigliotta[2] and Stefano Nolfi[1]

Abstract
We introduce FARSA, an open-source Framework for Autonomous Robotics Simulation and Analysis, that allows us to easily set up and carry on adaptive experiments involving complex robot/environmental models. Moreover, we show how a simulated iCub robot can be trained, through an evolutionary algorithm, to display reaching and integrated reaching and grasping behaviours. The results demonstrate how the use of an implicit selection criterion, estimating the extent to which the robot is able to produce the expected outcome without specifying the manner through which the action should be realized, is sufficient to develop the required capabilities despite the complexity of the robot and of the task.

Keywords
Evolutionary robotics, embodied cognition, open software, simulation framework

1 Introduction

Adaptive behaviour models focus on the study of how embodied agents develop their capabilities autonomously while interacting with their physical and (eventually) social environment. For many years, these studies have been confined to relatively simple agents and tasks. Recent research, however, demonstrated how this method can be extended to studies that involve agents with complex morphologies and rich sensory–motor systems mastering relatively hard tasks (Baranes & Oudeyer, 2013; Massera, Tuci, Ferrauto, & Nolfi, 2010; Reil & Husbands, 2002; Rolf, Steil, & Gienger, 2010; Savastano & Nolfi, 2012; Tuci, Massera, & Nolfi, 2010; Yamashita & Tani, 2008). From a modelling point of view complexity does not represent a value in itself. We fully bound the Occam's razor argument that claims that given two explanations of the data, all other things being equal, the simpler explanation is preferable. After all, one of the key contribution of adaptive behaviour research consists in the demonstration of how complex abilities can emerge from the interactions between relatively simple agents and the environment. On the other hand, the modelization of a given phenomenon necessarily require the inclusion of the characteristics that constitute key aspects of the targeted objective of study. In some cases, therefore, the use of complex agents and/or tasks is necessary. For example, the modelization of the morphological characteristics and of the articulated structure of the human arm constitutes a prerequisite for modelling human object manipulation skills. Likewise, the use of agents provided with rich sensory systems constitutes a necessary prerequisite for modelling sensory integration and fusion.

From a methodological perspective, however, the need to build rather complex models for tackling these research issues currently represents a barrier that might significantly slow down research progress in this area. In this paper we introduced FARSA, an open source software tool that allows to easily set up and carry out adaptive experiments based on the iCub humanoid robot (Metta, Sandini, Vernon, Natale, & Nori, 2008; Sandini, Metta, & Vernon, 2004) as well as on other robotic platforms. FARSA does not only provide a simulator, since it consists of a set of integrated libraries: a robot/environmental simulator, a sensor and actuator library, a controller library, and an adaptive library. Moreover, it comes with a rich graphical interface that facilitates the visualization and analysis

[1]Institute of Cognitive Sciences and Technologies, CNR, Rome, Italy
[2]NAC Laboratory, Department of Humanities, University of Naples Federico II, Naples, Italy

Corresponding author:
Onofrio Gigliotta, NAC Laboratory, Department of Humanities, University of Naples Federico II, 1 via Porta di Massa, Naples 80133 Italy.
Email: onofrio.gigliotta@unina.it

of the characteristics of the model and of the behavioural and cognitive processes originating from the agent/environmental interaction. For these reasons we believe that it can contribute to boost adaptive behaviour research addressing the acquisition of multiple skills and the development complex capabilities.

We then illustrate a series of experiments in which an iCub robot (Metta et al., 2008; Sandini et al., 2004) is trained through an evolutionary algorithm for the ability to display integrated reaching and grasping capabilities. The results obtained in these experiments demonstrate how the use of an implicit selection criterion, estimating the extent to which the robot is able to produce the expected outcome of the actions, is sufficient to develop the required capabilities despite the complexity of the robot, of the robot's sensory–motor system, and of the task. These experiments have been realized through the use of FARSA and constitute two of the exemplificative examples provided with the tools. Therefore, they can be easily be replicated and varied by the reader.

In the next section we introduce FARSA. In Section 3 we describe the relation of our experiment on integrated reaching and grasping to the state of the art. In Section 4 and 5 we describe our experiments and results. Finally in Section 6 we draw our conclusions.

2 FARSA

FARSA (see http://laral.istc.cnr.it/farsa/) is an open-source tool designed to carry on experimental research in embodied cognitive science and adaptive behaviour.

It combines in a single framework the following features:

- It is open-source, so it can be freely modified, used, and extended by the research community.
- It is constituted by a series of integrated libraries that allow it to easily design the different components of an embodied model (i.e. the agents' body and sensory–motor system, the agents' control systems, and the ecological niche in which the agents operate) and that allow to simulate accurately and efficiently the interactions between the agent and the environment.
- It comes with a rich graphical interface that facilitates the visualization and analysis of the elements forming the embodied model and of the behavioural and cognitive processes originating from the agent/environment interactions.
- It is based on a highly modular software architecture that enables a progressive expansion of the tool features and simplifies the implementation of new experiments and of new software components.
- It is multi-platform, i.e. it can be compiled and used on Linux, Windows, and Mac OS X operating systems.
- It comes with a set of exemplificative experiments and with a synthetic but comprehensive documentation that should enable users to quickly master the tool usage.

Other related tools include: Webots™ (Michel, 2004), USARSim (Carpin, Lewis, Wang, Balakirsky, & Scrapper, 2007), Gazebo (Koenig & Howard, 2004), ARGOS (Pinciroli et al., 2012), and LpzRobots (Der & Martius, 2012).

In the following sub-sections we briefly review the characteristics of its main components.

2.1 The robots/environment simulator library

The robots/environment simulator (worldsim) is a library that allows the simulation of robots and the environment in which they operate. The library supports both individual robot simulation and collective experiments in which several robots are placed in the same environment. The physical and dynamical aspects of the robots and of the robot/environment interactions can be simulated accurately by using a 3D dynamics physics simulator or by using a faster but simplified kinematic engine. For what concern the dynamics simulation, FARSA relies on the Newton Game Dynamics engine (Jerez & Suero, 2004) that enables accurate and fast simulations. The underlying dynamic engine has been encapsulated so as to enable the inclusion of alternative engines.

Currently, FARSA supports the following robotic platforms: the Khepera (Mondada, Franzi, & Ienne, 1993), the e-Puck (Mondada et al., 2009), the marXbot (Bonani et al., 2010), and the iCub (Sandini et al., 2004). These robots have been designed by assembling a series of building blocks (physical elements, sensors, and motorized joints) that users can re-use to implement alternative, not yet supported, robots.

In the case of the iCub, the simulator is based on the YARP (Metta, Fitzpatrick, & Natale, 2006) middleware library (the same command used to read the robot's sensors and control the robot's motor can be used to work with the simulated or real robot). This strongly facilitates the possibility to port results from simulation to reality and the possibility to integrate into FARSA projects the software modules available from the iCub software repository (http://wiki.icub.org/iCub_documentation).

With respect to the iCub simulator developed by Tikhanoff et al. (2008), the simulation library included in FARSA presents a series of advantages: it strictly conforms to the real kinematic joint structure of the robot, it allows to simulate multiple robots, it includes both a dynamic and kinematic engine, and it provides an enhanced visualization tool.

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

2.2 The sensor and motor library

FARSA also includes a library of ready-to-use sensors and actuators. In some cases, sensors and actuators include software routines that pre-elaborate sensory or motor information (e.g. to reduce its dimensionality) and/or integrate different kinds of sensory–motor information (as in the case of actuators that set the torque to be produced by a joint motor on the basis of the current and desired position of the controlled joint).

Wheeled robots are provided with infrared, ground, traction force, linear vision, and communication sensors, among others. Moreover, they are provided with wheels, grippers, LEDs, and communication actuators.

The iCub robot is provided with proprioceptors that measure the current angular position of the robot's joints, tactile sensors, and vision sensors among others and with actuators that control all the available DOFs. The state of the robot's sensors and actuators, as well as the state of selected variables of the robot's control system, can be graphically visualized while the robot interacts with the environment. This provides a useful analysis and debugging tool.

2.3 The controller libraries

These libraries enable the user to design, modify, and visualize the robot's control system. Currently FARSA includes two libraries that support the design of neuro-controllers. Users willing to use other architectures or formalisms can integrate into FARSA alternative libraries.

Evonet is an easy-to-use library that enables users to graphically design, modify, and visualize the architecture of the robot's neural controller as well as the properties of the neurons and of the connection weights. The library supports logistic, leaky integrator, and threshold neurons. NNFW is an alternative object-oriented library that provides a larger variety of neuron types and output functions (Gaussian, winner-take-all, ramp, periodic, etc.) and supports the use of a radial basis function neural network.

Thanks to the integration between the controller and the sensory and motor libraries, the sensory and motor layer of the neural controller is automatically generated on the basis of the selected sensors and actuators. Moreover, the update of the sensory neurons and the update of the actuators on the basis of the state of the motor neurons is handled automatically.

Finally, the graphic viewer of the robot's controller also enables users to lesion and/or to manually manipulate the state of the sensors, internal, and motor neurons in order to analyse the relationship between the state of the controller and the behaviour that originates from the robot/environmental interaction.

2.4 The adaptation libraries

These libraries enable the user to subject a robot or a population of robots to an adapting process (i.e. to a evolutionary and/or learning process during which the characteristics of the robots are varied and variations are selected so as to improve the abilities of the robots to cope with a given task/environment).

The adaptation libraries that are currently available support the use of evolutionary algorithms (including steady state, truncation selection, and Pareto-front algorithms), supervised learning algorithms (i.e. back-propagation), and unsupervised learning algorithms (i.e. Hebbian learning). The evolutionary algorithms are parallelized at the level of the individual's evaluation and can therefore run significantly faster in multi-core machines and computer clusters.

In the case of evolutionary and supervised algorithm, the variation in performance during the adaptation can be monitored and analysed in the associated graphics renderer.

2.5 Usability and speed

FARSA is well documented, easy to use, and provided with a rich graphical interface that facilitates monitoring and debugging. The inclusion of exemplificative experiments (including the two experiments described in this paper) enables easy replication and a variety of interesting case studies.

A large spectrum of experiments can be configured and varied through parameters. More specifically, the type of robotic platform, the sensors and actuators of the robot, the characteristics of the neural controller, and the type and the characteristics of the adaptive process can be set and varied easily through the graphical interface or through a text editor. The realization of experiments that involve non-parametric variations (i.e. that require a new type of fitness function or a new type of sensor) require writing C++ extensions. This task, however, is facilitated by the fact that experiments are defined as plugins, i.e. relatively short programs that can be compiled separately from FARSA and loaded at runtime. Plugins can also be used to implement larger software extensions (e.g. new learning algorithms or new graphics widgets).

FARSA is optimized and parallelized so to reduce as much as possible the time required to carry on computationally expensive experiments. The simulation speed clearly depends on the complexity of the robotic platform and of the robot/environment interactions. In the case of the experiments described below, the simulation of the robot/environmental interaction on a standard single processor (Quad-Core AMD Opteron™ Processor 2374 HE at 2.2 GHz) under Linux runs 116 times and 2.6 times faster than real time (in the case of the experiments reported in Sections 4 and 5,

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

respectively). Moreover the simulation of the evolutionary process on a multi-thread cluster runs approximately 666 times and 19 times faster on two quad-core processors using 8 threads (the same type of processor and operating system as above, experiment reported in Sections 4 and 5, respectively).

3 Relation to the state of the art

Reaching and grasping capabilities can be developed through trial-and-error and/or supervised learning methods (Barto, 2003). In trial-and-error methods, the motor capability is acquired without the help of an explicit teacher or trainer, and the adaptive process is driven by intrinsic feedback. Examples of intrinsic feedback are the kinesthetic and tactile sensations experienced when an object has been successfully grasped or the sight of a ball entering inside the net after a kicking action. In supervised training methods, instead, the intrinsic feedback is augmented with extrinsic information provided by the teacher. This information might consist of the sequence of sensory states experienced by the robot while its arm is driven by a caretaker toward a target object to be reached (in kinesthetic teaching methods, see for example Yamashita and Tani (2008)) or by the demonstration performed by the teacher of the action that should be performed by the robot (in learning by demonstration methods, see for example Miyamoto and Kawato (1998)). Most of the research in the field of artificial intelligence and adaptive behaviour focus on the latter paradigm. In this paper, instead, we will focus on trial-and-error methods relying on intrinsic feedback (e.g. the robot's capability to perceive whether or not and eventually to what extent a reaching and/or grasping action has been successfully carried out). Previous attempts to study how robots can develop reaching or grasping capabilities through trial-and-error methods include experiments with non-redundant systems provided with two actuated DOFs (Berthier, Rosenstein, & Barto, 2005; Schlesinger, Parisi, & Langer, 2000) or experiments in which the robots were provided with significant built-in competences (Oztop, Bradley, & Arbib, 2004). More specifically, Schlesinger et al. (2000) studied the development of reaching behaviour in a simulated agent provided with a 2-dimensional arm with two actuated DOFs, a bi-dimensional vision system with one actuated DOF, and a tactile sensor located on the final portion of the arm. The robot's neural network controller received as input the angular state of the arm joints, the state of the tactile sensor, and the visual information extracted from the camera and control the two DOFs of the arm and one DOF of the visual system. The neural network controller was trained through an evolutionary method (Nolfi & Floreano, 2000) on the basis of a performance criterion calculated by computing the average number of time steps spent by the robot touching the target object. Oztop et al. (2004) studied the development of grasping behaviour in a simulated robot provided with an arm and hand with 19 actuated DOFs. The reaching behaviour was pre-programmed in the robot on the basis of the Jacobian transpose method (Sciavicco & Siciliano, 2004). Learning was thus confined to the mapping of a series of sensorily extracted object affordances into a series of grasping parameters able to shape the pre-existing reaching capability into an effective grasping behaviour. The neural network controller was trained through a reinforcement learning algorithm (Sutton & Barto, 1998) and received positive reward for the trials producing successful or nearly successful grasps and negative reward for trials leading to unstable grasps or no object contact. Berthier et al. (2005) studied the development of a reaching behaviour in a simulated robot provided with an arm with 2 controlled DOFs on the shoulder (flexion–extension and adduction–abduction). The robot's neural network controller received as input the current state and velocity of the two joints and produced as output the intensity of the torque to be applied by two muscle-like actuators. The network was trained through a reinforcement learning algorithm (Sutton & Barto, 1998) by providing to the robot positive and negative rewards when the hand of the robot approached or moved away from the target, respectively. The experiments described in this paper, instead, concern the study of how a highly redundant humanoid robot can develop reaching and grasping capabilities from scratch or on-top of simple reflex-like competences. The relation between the experiments presented in this paper and our own previous related work (Massera, Cangelosi, & Nolfi, 2007; Massera et al., 2010; Savastano & Nolfi, 2012, 2013) will be discussed below. Although the first phase of reaching and grasping development in children are clearly characterized by a trial-and-error learning process (Oztop et al., 2004), the objective of this paper is not to model human learning but rather to demonstrate how apparently complex behavioural capabilities can be successfully acquired through a simple trial-and-error adaptive process that do not require specification of the manner through which the target actions should be realized.

4 Reaching

In this section we describe how a simulated iCub robot can acquire the capability to reach with its left arm any arbitrary target position in its peripersonal space by controlling six actuated joints (two joints of the iCub's torso and four joints of the iCub's left arm). The connection weights of the robots' neural controller are adapted through an evolutionary method for the ability to minimize the average distance between the left hand of the robot and the target location averaged over

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

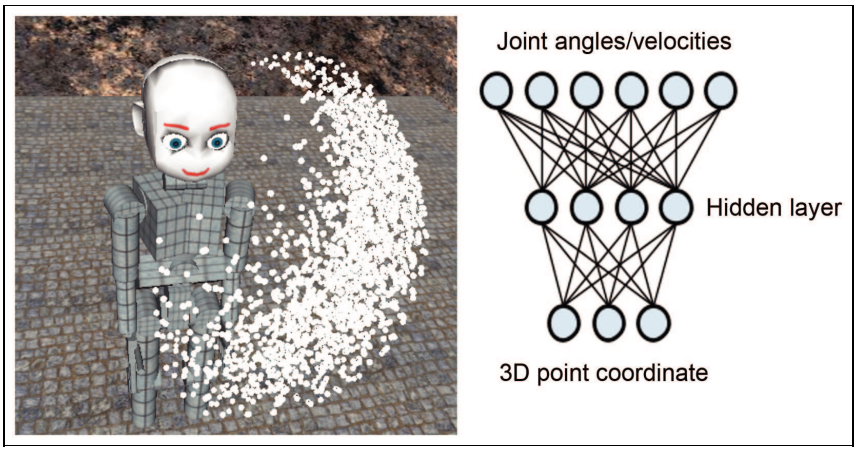


Figure 1. (*Left*) The simulated iCub. The white points shows all the possible target positions (see text for an explanation). (*Right*) The architecture of the neural controller. The lower, intermediate, and upper layer indicate the sensory, internal, and motor neurons. Lines represents connections from the lower to the upper layer.

several trials in which the robot has to reach different target positions.

4.1 Method

The robot's neural controller (Figure 1) is provided with three sensory neurons that encode the position of the target object in Cartesian coordinates, four internal neurons, and six motor neurons that control the desired angular position or velocities (depending on experimental setup, see below) of the six actuated DOFs (i.e. the rotation and the extension–flexion of the torso; the extension–flexion, the abduction–adduction and the supination–pronation of the arm; the extension–flexion of the forearm). The sensory neurons are fully connected to internal neurons that are fully connected to motor neurons. To verify the role of the sensory feedback during the robot/environment interactions we ran two sets of experiments. In **steady-encoding** experiments the sensory neurons encode the offset of the current target position along the X, Y, and Z axes, normalized in the range [0,1], with respect to centre of the iCub body, and the motor neurons encode the desired angles for the final posture of the arm by using a linear mapping (actual torques are set through a PID controller). The offset between the desired and the target angular positions are then used to set the velocity of the joint motors on the basis of a simple proportional controller. In the **unsteady-encoding** experiments, instead, the three sensory neurons encode the offset of the current target position along the X, Y, and Z axes, normalized in the range [0,1], with respect to the centre of the left palm, and the motor neurons encode directly the velocity of the joint motors. In the latter case the robot can use the perceptual feedback of its own actions to refine its behaviour while it interacts with the environment. In the former case, instead, the sensory state does not change while the robot moves and consequently the robot cannot exploit the sensorial effects of its own actions. Moreover, the sensorial information perceived in the **unsteady-encoding** correlate directly with the extent to which the robot has successfully carried out its action.

The output of internal and motor neurons was computed accordingly the following equation:

$$o_i = \sigma\left(\sum_{j=0}^{N} x_j w_{ji}\right) \quad (1)$$

where σ is the standard logistic function: $1/(1+e^{-x})$, x_j is the output of the jth presynaptic neuron and w_{ji} is the synaptic weight from the jth presynaptic to ith postsynaptic neuron. The update rate of the state of the sensors, of the neural controller, of the actuators, of the robot and of the environment is 25 Hz. The characteristics of the robot and of the architecture of the robots' neural network are kept fixed. The strength of the connection weights are adapted by using an evolutionary method (Nolfi & Floreano, 2000). The initial population consists of 20 randomly generated genotypes, which encode the 46 free parameters of 20 corresponding neural controllers. Each gene is constituted by 8 bits that encode a corresponding floating point value in the range $[-5.0, +5.0]$. During each generation, each individual is allowed to produce an offspring (i.e. a genotype identical to that of the parent with 5% of its bit randomly mutated). The 20 parent and the 20 offspring

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

Table 1. Angular positions selected uniformly within the joints limits used to generate a representative set of all possible reachable positions.

Joint	Limits
Torso rotation	$[-25.0, +25]$
Arm abduction–adduction	$[32.16, 64.32, 96.48, 128.64]$
Torso extension–flexion	$[-2.5, 5, 12.5]$
Arm supination–pronation	$[-13.6, 9.8, 33.2, 56.6]$
Arm extension–flexion	$[-74.4, -53.3, -32.2, -11.1]$
Forearm extension–flexion	$[33.2, 51.4, 69.6, 87.8]$

individuals are evaluated. The genotypes of offspring individuals that outperform parents are used to replace the genotypes of the worst parents. The genotypes of the remaining offspring are discarded. The reproduction, evaluation, and selection process is repeated for 5000 generations. Each individual is evaluated for 400 trials, each lasting up to 17.5 s, during which it should reach 400 corresponding target positions extracted from a set of 2304 reachable positions. To get a subset of points as equidistributed as possible, we use the crowding distance (Deb, Agrawal, Pratap, & Meyarivan, 2000) to sort all reachable positions on the basis of their Cartesian coordinates. The 2304 reachable points have been calculated by storing the position that the centre of the robot's left palm assumed when the six actuated joints were moved to all possible combination of states within the values indicated in Table 1. Furthermore, to favour the selection of individuals that are able to generalize their abilities to any possible reachable position, the target position experienced during each trial was randomly chosen within a spherical area with a diameter of 2 cm centred around the current extracted reachable position.

The fitness is calculated on the basis of the following equation:

$$F = \sum_{t=1}^{400} e^{\left(-\frac{\|\text{palmPos}-\text{targetPos}\|}{0.04}\right)} \quad (2)$$

Where t is the trial, and $\| \text{palmPos} - \text{targetPos} \|$ is the Euclidean distance in meters between the centre of the robot's left palm and the centre of the target location measured at the end of each trial. To verify whether an incremental adaptive process can lead to better performance, we ran an additional set of experiments referred to below as incremental. More specifically, to simulate the condition on which the problem is initially simplified and become progressively harder as soon as the skills of the individuals improve, they are rewarded with the maximum fitness during trials in which the palmPos – targetPos distance is below a threshold . This threshold is initially set to 5 cm at generation 0 and it is progressively reduced by 20% after a generation in which the average fitness of all individuals is greater than 0.6, and it is definitely set to 0.0 when it becomes lower than 1 cm. For sake of comparison, consider that the height of the iCub is about 1 m.

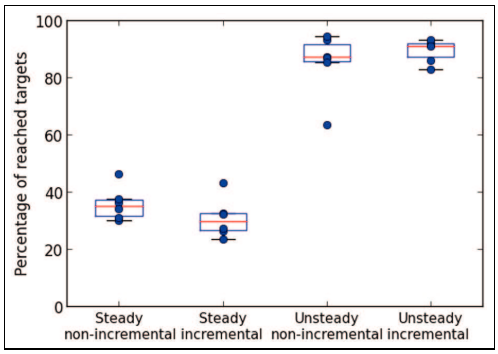


Figure 2. Percentage of target locations reached with an accuracy of at least 5 cm (i.e. with a distance between the centre of the palm and the centre of the target location less or equal to 5 cm). The four box plots show the distribution of performance of the six best individuals each from an independent run with random initial conditions.

4.2 Results

The combination of the steady versus unsteady encoding and incremental versus non-incremental adaptive process leads to four sets of experiments. For each experiments six replications starting from different randomly generated populations were run. Evolved individuals were then post-evaluated on the entire set of 2304 reachable target locations by calculating percentage of target location reached with an accuracy of at least 5 cm.

By analysing the performance of the best evolved individuals in the four experimental conditions (see Figure 2), we can see how the individuals evolved in the unsteady condition significantly outperform those evolved in the steady condition. This results confirms that the possibility to exploit the sensory feedbacks caused by the robot's actions and/or the availability of information that strongly correlates with the extent to which the robot successfully accomplishes the current action strongly facilitates the development of the effective solutions.

The comparison of the performance obtained in the incremental versus non-incremental experimental conditions does not reveal significant differences.

Overall the analysis of the results in the best experimental conditions indicates how the adapted individuals can reach close to optimal performance. This is a remarkable result given the simplicity of the neural controller and given that some of the targets located in

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

peripheral areas of the robot's peripersonal space are hard to reach due to the limits and constraints that affect the robot's movements in these regions.

For instructions on how to replicate this experiment with FARSA and on how to analyze the evolved solutions, see http://laral.istc.cnr.it/res/reach.

5 Reaching and grasping

In this section we describe how a simulated iCub robot can develop integrated reaching and grasping capabilities that enable it to reach a ball located in varying positions over a table, grasp it, handle it, and elevate it. Beside the difficulties concerning the need to control an articulated arm with many DOFs (Bernstein, 1967), this represents a rather challenging task since it requires interaction with physical objects (including a sphere that can easily roll away from the robot's peripersonal space) and integration of three interdependent behaviours (reaching, grasping, and lifting).

5.1 The method

In the case of this experiment, the robot's controller includes a richer set of sensors and actuators, a larger neural network, and a greater number of parameters to be varied during the adaptive process. Adapting individuals are provided with an hand-coded neural circuit that produce a simple reflex behaviour consisting in turning the robot head toward red objects.

The sensory system (Figure 3(b), bottom layer) includes two neurons that encode the offset between the sphere and the hand over the visual plane (*dx*, *dy*, by visual plane we means the two dimensional image perceived by the robot's camera), four neurons that encode the current angular position of the pitch and yaw DOFs of the neck (n0, n1) and of the torso (t0, t1), and nine sensory neurons that binarily encode whether the five tactile sensors located on the fingertips (Rf1, Rf2, Rf3, Rf4, Rf5) and the four tactile sensors located on the palm (Rp1, Rp2, Rp3, Rp4) are stimulated.

The motor system (Figure 3(b), top layer) includes two motor neurons that control the desired angular position of pitch and yaw DOFs of the torso (T0 and T1), seven motor neurons that control the desired angular position of the seven corresponding DOFs of the right arm and wrist (RA0, RA1, RA2, RA3, RA4, RA5 and RA6) and a right-hand motor (RF0) that controls the desired angular position of all joints of the hand (the fingers' abduction–adduction is kept fixed). This means that all fingers extend/flex together.

The neural network is also provided with seven internal neurons that receive connections from all sensory neurons and project connection to all motor neurons (Figure 3(b), intermediate layer). These neurons are leaky integrators, that is their activation at a given time step depends on both the input at that time step and on the activation at the previous time step. The output of the ith internal neuron is computed as follows:

$$o_{i,t} = \alpha_i o_{i,t-1} + (1 - \alpha_i)\sigma\left(\sum_{j=0}^{N} x_{j,t} w_{ji}\right) \quad (3)$$

where $o_{i,t}$ is the output of the ith internal neuron at time step t, α_i is a time integrator parameter that determines how much the output at the current time step depends on the output at the previous time step, $\sigma(z)$ is the standard logistic function as before, $x_{j,t}$ is the output of the jth presynaptic neuron at time step t and w_{ji} is the synaptic weight from the jth presynaptic to ith postsynaptic neuron. The update rate of the state of the sensors, of the neural controller, of the actuators, of the robot and of the environment is 25 Hz (50 ms per step).

The reflex behaviour is realized by a neural circuit (Figure 3(a)) with two sensory neurons, that encode the average offsets of red pixels over the vertical and horizontal axis of the visual field, which are directly connected to two motor neurons, that control the angular position of the neck (N0, N1). The four weights and the two biases of the motor neurons are set manually. The other 226 parameters are adapted.

At the beginning of each trial the sphere is placed in a random position inside one of four square areas with a side of 4 cm (Figure 4). The first two of these areas are located in front of the iCub at a distance of 25 cm and 35 cm, the other two are located 10 cm on the left and on the right side and at a distance of 30 cm. Each trial lasts 300 time steps (i.e. 15 s) plus 10 additional time steps during which the plane is removed to verify whether or not the ball is held by the robot.

The fitness is computed on the basis of the following equations:

$$F_t = 0.3D_t + 0.2T_t + 0.2O_tC_t + 0.3Q_t + G_t \quad (4)$$

$$F = \sum_{t=1}^{N} F_t \quad (5)$$

Where F is the overall fitness of the individual, F_t is the fitness at trial t, N is the number of trials and D_t, T_t, O_t, C_t, Q_t, and G_t are fitness components, ranging from 0 to 1, that reward the individuals for bringing their hand near the object (D_t),touching the object with the palm (T_t), opening the fingers far from the object (O_t), closing the finger near the object (C_t), closing the finger around the object (Q_t), holding and elevating the object (G_t). These fitness components have been introduced to increase the individuals' evolvability (i.e. the probability that random variations might lead to performance improvements) and to channel the adaptive process toward the acquisition of abilities that constitute a prerequisite for the development of the required

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

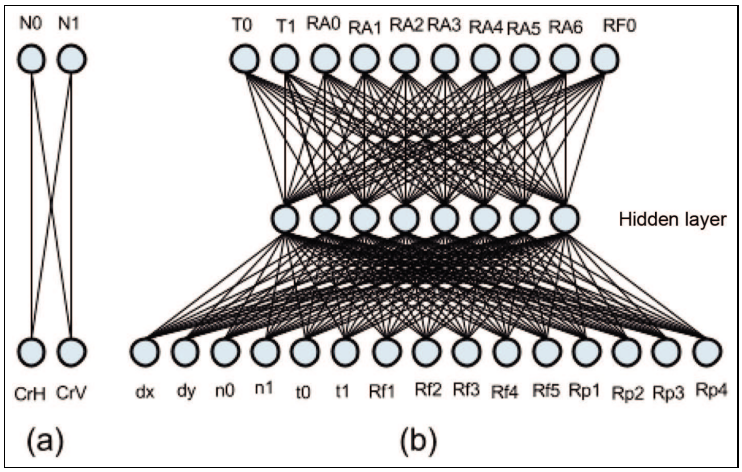


Figure 3. The architecture of robot's neural controller. The lower, intermediate, and upper layer represent the sensory, internal, and motor neurons, respectively. Lines represents connections from the lower to the upper layer. The connection weights and biases and of the neural circuit shown in (a) are manually set and fixed. All other connection weights and biases are adapted.

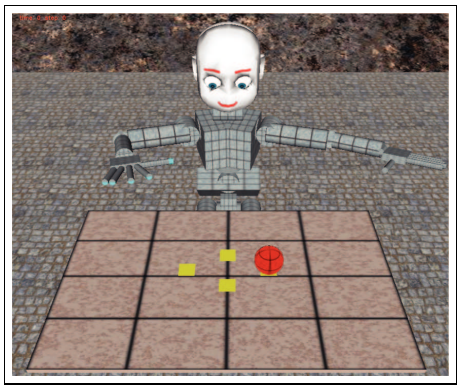


Figure 4. The experimental setup. The robot is shown in the posture set at the beginning of each trial. The left arm of the robot is not moved. The yellow squares on the table show the areas where the object can be located.

capabilities (we will come back to this issue in the discussion section). The used components and their parameters have been chosen on the basis of our intuition and have not be optimized on the basis of a trial and error approach. They are computed on the basis of the following equations (the subscript indicating the dependency on the trial has been removed for clarity):

$$D = e^{-5d} \tag{6}$$

$$T = \min\left(\frac{n}{10}, 1\right) \tag{7}$$

$$O = \frac{1}{n_0}\sum_{s=1}^{n_0} E_s \tag{8}$$

$$C = \begin{cases} \frac{1}{300-n_c}\sum_{s=n_c}^{300} 1 - E_s, & \text{if } n_c \neq 300 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$Q = \min\left(\frac{f}{4}, 1\right) \tag{10}$$

$$G = \begin{cases} 0, & \text{if } o_z \leqslant -0.1 \\ 0.5 + \frac{o_z + 0.1}{0.2}0.5, & \text{if } -0.1 < o_z < 0.1 \\ 1, & \text{if } o_z \geq 0.1 \end{cases} \tag{11}$$

where d is the distance between the centre of the palm and the surface of the object at the end of the trial; n is the number of steps in which the palm of the robot touched the object during the current trial; n_0 and n_c are the steps at which palm enters in contact with the object for the first and for the fifth time, respectively, or 300 when the conditions are never satisfied; E_s is the extension of the fingers at step s; f is the maximum number of fingers that entered in contact with the object concurrently during the trial; o_z is the displacement along the vertical axis of the object centre (0 means the object is exactly on the table).

To support the evolution of robust behaviours while minimizing the simulation costs, the number of trials is initially set to 4 and is then increased to 8, 12, 16, 20, 24, and 28 as soon as an evolving individual successfully grasps and holds the objects during 50%, 60%, 70%, 80%, 90% and 100% of the trials. Five replications of the experiment lasting 2000 generations were run. All other parameters were identical to that described in Section 3.

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

Table 2. Percentage of trials in which the best evolved robot of each replication successfully grasps and hold the ball during a post-evaluation test conducted for 100 trials.

Replication	1	2	3	4	5
Success rate (%)	77	74	69	66	50

5.2 Results

By analysing the obtained results we observed that in all replications of the experiment the evolved individuals display an ability to reach, grasp, and hold spherical objects located in varying positions (Table 2). In the case of the best replication of the experiment, the best individual displays a rather robust capability that allows it to successfully carry on the task in 77% of the trials. This represents a remarkable result in consideration of the rigidity of the robot body and of the difficulties of physically interacting with spherical objects that can easily roll away from the peripersonal space of the robot. The obtained solutions also represent progress with respect to the previous studies carried by some of the authors (see Massera et al. (2007)), in which the individuals were able to successfully accomplish a similar task but showed limited generalization capabilities with respect to variations of the object positions.

The visual inspection of the behavioural solutions displayed by these individuals (see http://laral.istc.cnr.it/res/reach-grasp/) can allow us to appreciate the importance played by the integration between the required elementary behaviours (i.e. reaching, grasping, and lifting) and by the way in which they are combined over time. Indeed, the way in which the best evolved individuals reach the object by bending the torso toward the table and by carefully pressing the ball over the table so as to block it, while the fingers are wrapped around the object, clearly demonstrate the importance of the fact that the reaching and the grasping abilities have been co-evolved to serve a common function.

Overall this demonstrates the potential advantages of acquiring the required elementary behavioural capacities through an adaptive process and of using methods that enable the co-development of multiple capacities. More specifically, for what concerns the experiments illustrated above, this suggests that the introduction of a fitness component that rewards the development of the required elementary capabilities and of components that reward the ability to appropriately combine and integrate the acquired elementary capabilities might be crucial for the development of general and effective solutions.

For instructions on how to replicate this experiment with FARSA and on how to analyse the evolved solutions see http://laral.istc.cnr.it/res/reach-grasp/.

6 Discussion and conclusion

The possibility to design adaptive agents able to develop their behavioural skills autonomously, while they interact with the physical and social environment in which they are situated, represented one of the most fascinating scientific landmarks of the end of the last century. Whether and how such methods can enable the synthesis of robots able to acquire complex behavioural and cognitive skills and able to progressively expand their behavioural and cognitive repertoire still represents an open question.

To achieve this challenging objective agents need to be able to first develop elementary capabilities and then more complex skills by recombining and integrating previously developed skills. However, the way in which this can be achieved still represents an open question.

A possible approach postulates a modular organization of the agents' control system in which different modules support the acquisition and production of the elementary capabilities and in which the elementary modules/capabilities are then combined to produce more complex skills (Mataric, 1998; Schaal, 2002; Wolpert & Kawato, 1998). The composition, however, is far from easy to achieve (Nemec & Ude, 2012), often employs very heuristic schemes (Reinhart, Lemme, & Steil, 2012), or needs in itself sophisticated modelling approaches (Kulic, Ott, Lee, Ishikawa, & Nakamura, 2012; Wrede et al., 2012).

An alternative approach postulates that multiple and complex capabilities can be obtained by recombining previously acquired behavioural and cognitive skills that do not necessarily correspond to different parts of the agent's control system (Nolfi, 2009; Yamashita & Tani, 2008). Within this approach, compositionality is seen as a property that arises from the acquisition and integration of multiple skills rather than a consequence of architectural constraints. The question of whether and how this approach can really lead to the progressive acquisition of a rich behavioural and cognitive repertoire, however, still remains to be answered.

In this paper we introduced a software framework that enables researchers to easily perform and analyse adaptive experiments involving relatively complex agents and tasks. We believe that the availability of tools of this type can significantly contribute to boost research in adaptive robotics by enabling the investigation of hard problems and the comparison of alternative models and methods.

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

Moreover, we reported the result of a series of experiments that demonstrate how a relatively complex humanoid robot provided with a simple non-modular controller can acquire multiple integrated behavioural capabilities. This is achieved through the use of multiple component fitness functions that enhance the evolvability of the system and channel the adaptive process toward promising directions. The question of whether and how this type of approach can scale to larger behavioural repertories constitutes an important research challenge for future research.

Funding

The authors gratefully acknowledge the financial support provided by the Italian National Research Council (CNR) through the EuroBioSAS Programme of the European Science Foundation.

References

Baranes, A., & Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, *61*(1), 69–73.

Barto, A. G. (2003). Reinforcement learning in motor control. In M. A. Arbib (Ed.), *Handbook of brain theory and neural networks* (pp. 968–972). Cambridge, MA: MIT Press.

Bernstein, N. (1967). *The co-ordination and regulation of movements*. Oxford, UK: Pergamo.

Berthier, N. E., Rosenstein, M. T., & Barto, A. G. (2005). Approximate optimal control as a model for motor learning. *Psychological Review*, *112*, 329–346.

Bonani, M., Longchamp, V., Magnenat, S., Rtornaz, P., Burnier, D., Roulet, G., …, Mondada, F. (2010). The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4187–4193). IEEE.

Carpin, S., Lewis, M., Wang, J., Balakirsky, S., & Scrapper, C. (2007). USARSim: A robot simulator for research and education. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 1400–1405). IEEE.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature PPSN VI* (pp. 849–858). Berlin: Springer.

Der, R., & Martius, G. (2012). The LpzRobots Simulator. In R. Der, & G. Martius (Eds.), *The playful machine: theoretical foundation and practical realization of self-organizing robots* (*Vol. 15*) (pp. 293–308). Berlin, Germany: Springer Verlag.

Jerez, J., & Suero, A. (2004). Newton game dynamics. Retrieved www.newtondynamics.com

Koenig, N., & Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2149–2154). IEEE.

Kulic, D., Ott, C., Lee, D., Ishikawa, J., & Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *International Journal of Robotics Research*, *31*(3), 330–345.

Massera, G., Cangelosi, A., & Nolfi, S. (2007). Evolution of prehension ability in an anthropomorphic neurorobotic arm. *Frontiers in Neurorobotics*, *1*(4).

Massera, G., Tuci, E., Ferrauto, T., & Nolfi, S. (2010). The facilitatory role of linguistic instructions on developing manipulation skills. *IEEE Compotational Intelligence Magazine*, *5*(3), 33–42.

Mataric, M. J. (1998). Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Sciences*, *2*(3), 82–86.

Metta, G., Fitzpatrick, P., & Natale, L. (2006). Yarp: Yet another robot platform. *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, *3*(1), 43–48.

Metta, G., Sandini, G., Vernon, D., Natale, L., & Nori, F. (2008). The iCub humanoid robot: An open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems* (pp. 50–56). New York, USA: ACM.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, *1*(1), 39–42.

Miyamoto, H., & Kawato, M. (1998). A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, *11*(7–8), 1331–1344.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., …, Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions* (*Vol. 1*, pp. 59–65). IPCB.

Mondada, F., Franzi, E., & Ienne, P. (1993). Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Proceedings of the 3rd international symposium on experimental robotics* (pp. 501–513). Tokyo: Springer.

Nemec, B., & Ude, A. (2012). Action sequencing using dynamic movement primitives. *Robotica*, *30*(5), 837–846.

Nolfi, S. (2009). Behavior and cognition as a complex adaptive system: Insights from robotic experiments. In C. Hooker (Ed.), *Handbook of the philosophy of science*. Amsterdam, The Netherlands: Elsevier.

Nolfi, S., & Floreano, D. (2000). *Evolutionary robotics*. Cambridge, MA: MIT Press.

Oztop, E., Bradley, N. S., & Arbib, M. A. (2004). Infant grasp learning: A computational model. *Experimental Brain Research*, *158*(4), 480–503.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., …, Dorigo, M. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, *6*(4), 271–295.

Reil, T., & Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, *6*(2), 159–168.

Reinhart, F., Lemme, A., & Steil, J. J. (2012). Representation and generalization of bi-manual skills from kinesthetic teaching. In *Proceedings of the IEEE-RAS international conference on humanoid robots*. IEEE.

Rolf, M., Steil, J. J., & Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, *2*(3), 216–229.

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909

Sandini, G., Metta, G., & Vernon, D. (2004). Robotcub: An open framework for research in embodied cognition. In *Proceedings of IEEE/RAS international conference on humanoid robots* (pp. 13–32). IEEE.

Savastano, P., & Nolfi, S. (2012). Incremental learning in a 14 DOF simulated iCub robot: Modeling infant reach/grasp development. In T. Prescott, N. Lepora, A. Mura, & P. Verschure (Eds.), *Biomimetic and biohybrid systems* (Vol. 7375, pp. 250–261). Berlin; Heidelberg: Springer.

Savastano, P., & Nolfi, S. (2013). A robotic model of reaching and grasping development. *IEEE Transactions on Autonomous Mental Development*, *5*(4), 326–336.

Schaal, S. (2002). Learning robot control. In M. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 983–987). 2nd edition.Cambridge, MA: MIT Press.

Schlesinger, M., Parisi, D., & Langer, J. (2000). Learning to reach by constraining the movement search space. *Developmental Science*, *3*, 67–80.

Sciavicco, L., & Siciliano, B. (2004). *Modelling and control of robot manipulators*. London; New York: Springer.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., & Nori, F. (2008). An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator. In *Proceedings of the 8th workshop on performance metrics for intelligent systems* (pp. 57–61). New York, NY: ACM.

Tuci, E., Massera, G., & Nolfi, S. (2010). Active categorical perception of object shapes in a simulated anthropomorphic robotic arm. *IEEE Transactions on Evolutionary Computation*, *14*(6), 885–899.

Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*(7–8), 1317–1329.

Wrede, B., Rohlfing, K. J., Steil, J. J., Wrede, S., Oudeyer, P. -Y., & Tani, J. (2012). Towards robots with teleological action and language understanding. In *IEEE humanoids, workshop on developmental robotics*. Osaka: IEEE.

Yamashita, Y., & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS Comput Biol*, *4*(11), e1000220.

Gianluca Massera, Tomassino Ferrauto, Onofrio Gigliotta and Stefano Nolfi, *Designing adaptive humanoid robots through the FARSA open-source framework*, Adaptive Behavior (volume: 22, issue: 4) pp. 255–265. Copyright ©[2014] (The Authors(s)). Reprinted by permission of SAGE Publications. DOI: https://doi.org/10.1177/1059712314536909