Development of Integrated Behaviour in a Simulated Humanoid Robot

Exploiting Language Assisted Training and Self Talk

Tobias Leugger
tobias.leugger@epfl.ch

Ecole Polytechnique Fédérale de Lausanne, Switzerland
School of Engineering
Laboratory of Intelligent Systems

Consiglio Nazionale delle Ricerche, Rome, Italy
Institute of Cognitive Sciences and Technologies
Laboratory of Autonomous Robotics and Artificial Life

Masters Thesis
School of Computer and Communication Sciences

March 15, 2012

Supervisor
Dr. Stefano Nolfi
CNR / LARAL

Professor
Prof. Dario Floreano
EPFL / LIS

Assistant
Dr. Steffen Wischmann
EPFL / LIS

Abstract

In this project, a simulated humanoid robot learns to reach for, grasp and move a spherical object to a desired target location. The trial and error training process is divided into multiple stages where the robot first develops lower-level behaviours that it can later integrate to achieve the higher-level goal. The robot is supported by a caretaker who provides linguistic labels indicating the action that should currently be performed. Multiple approaches to teaching the robot to become autonomous (no longer rely on the caretaker) are explored. One such technique is to let the robot engage in a form of self talk, where it can give itself the linguistic instructions used by the caretaker.

The robots trained with the proposed method are able to accomplish all the desired behaviours. The results of the experiments suggest that the lower-level skills serve as scaffolding for the integrated behaviour. The self talk mechanism improves the robustness of the autonomous behaviour, and the ability to adapt it to situations that are not experienced during training. Some individuals are even able to adapt better to new situations when they are acting autonomously than when they are guided by the caretaker.

Acknowledgements

I would like to thank my supervisor Dr. Stefano Nolfi of the Laboratory of Autonomous Robotics and Artificial Life for his continued guidance and for the insightful suggestions during times of difficulty and doubt. Thank you also to all the members of the lab for their support during my project, but beyond that, for welcoming me to the LARAL and making me feel at home in Rome. The friendly environment and the many interesting, fun, and educational conversations I have had with them have made my thesis work an enjoyable experience.

Thank you also to my supervisor at the EPFL Dr. Steffen Wischmann and my professor, Prof. Dario Floreano for giving me the opportunity to realise this project and for their support.

I am also grateful to my girlfriend, Kremena, who answered countless English questions and has been patient with me through the busy and stressful times. The gratitude is extended to my family and close friends, who have all had a positive influence on the outcome of this thesis, whether direct or not.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this thesis is to allow a humanoid robot to acquire relatively complex behaviours such as object manipulation skills. The hypothesis is that such complex behaviours can be developed by combining previously learned, simpler skills which may be referred to as motor primitives. The other objective is to verify whether language can facilitate the acquisition of the integrated behaviour skills. Concretely, the influence of the presence of a caretaker that instructs the robot on the sequence of motor primitives to use, and the ability of the robot to follow its own instructions by engaging in a form of self talk, is studied.

1.1 Background

The subject of this thesis is located at the intersection of human-centred robotics on one side and developmental and evolutionary robotics on the other. Human-centred robotics provides the overall vision, developmental robotics serves as an appropriate methodology for the design of the experiments and the formulation of the concrete goals, and evolutionary robotics contributes a well-established learning algorithm. Both developmental and evolutionary robotics share the view that robots should not be fully hand-designed but should acquire part of their characteristics through an adaptive process.

The goal of human-centred robotics is to develop robots that interact directly with humans in normal human environments. Possible application areas include care for the elderly, support for patients in physical recovery and general assistance in daily life. These robots will have to operate autonomously in normal human environments. Such environments are characterised by uncertainty, rapid changes and the presence of humans. Existing robotic systems work well when the environment is static and known in advance. No robot today has a skill set comparable to a human, so the number of tasks they are able to carry out is rather limited. In order to create robots that are helpful in everyday situations robots need to expand their skill set and learn to deal with the variability of our environment. These and other requirements and outstanding issues of human-centred robotics are outlined by Schaal in [1]. Successful development of such robots requires the cooperation of various disciplines, including psychology, ethics, neuroscience and classical robotics.

Developmental robotics, a field which partially overlaps with human-centred robotics, refers to the bi-directional exchange of ideas between researchers on ontogenetic development (development over the lifespan of a biological organism) and robotics. On the one hand, findings in psychology and neuroscience on human development are used as inspiration to create better robotic systems; on the other hand, robots are used to help test and verify models from developmental sciences [2]. Epigenetic robotics is closely related to developmental robotics and both can be seen part of biorobotics [2]. It is more and more accepted that all of these fields provide viable ways to design the next generation of robots [3].

A more intuitive account of the benefits of using humans as inspiration for the design of humanoid robots can be given by considering the evolution and continued existence of our species. Humans are obviously the most robust, adaptable and advanced humanoid system we have come across, capable of operating in uncertain and rapidly changing environments. It seems to make sense, therefore, to fashion the control system of humanoid robots after the functioning of the human.

Many of the components of ontogenetic development that have been subject to research (summarised in [2]) are used as guidelines for the experiments presented here. One key notion is *self-organisation*: structured behaviour can emerge from fine-grained local interactions among the components of a system and its environment. Another is that development is an *incremental process*: behavioural skills and cognitive functions evolve in multiple stages. More advanced structures build on previously established ones that are less efficient or incomplete. A related idea is that the presence of *changing constraints* can benefit development. For example, early limitations of the sensory and motor system decrease the amount of information to process and can make it easier to extract the relevant aspects. A final principle of development which should be mentioned here is the importance of *social-interaction* and language. Social peers of a developing individual can guide its attention and actions, which leads the individual to experience more meaningful stimuli.

Researchers have found evidence that cognition and language are closely linked in the human brain (findings summarised in [4, 5]). This idea has been used to show that language can help with motor control in robots [6, 7] and with classification of objects and postures [4, 8]. Exploration of the influence of language on learning and motor control in robots is an emerging field that is not yet very well understood [9]. Psychologist Lev Vygotsky was the first to develop the idea that language is an important tool in human cognition [5]. Inspired by this theory, Mirolli and Parisi [5] hypothesize that language comprehension and self-talk can facilitate the acquisition of action skills in robots. Vygotsky gives one example of children learning to solve a new task that illustrates this concept: after having been given instructions on how to solve a task, children repeat the instructions out loud to themselves when faced with the task again. This use of self talk helps them to internalise more and more complex behaviours. In the last part of our experiments we explore a similar process to learn the order and timing of a sequence of motor primitive. This is based on the first part of the experiments where the robot learns to respond to verbal instruction by performing a certain action.

The last of the three mentioned robotics fields is evolutionary robotics. It is a method for developing autonomous robots by letting them adapt to the

environment and the desired tasks without intervention of a human designer. The algorithms used are inspired by the evolution of biological organisms. An evolutionary algorithm is a process that makes random changes to the free parameters of the robot's control system (or the robot's morphology) and keeps or discards the changes based on the resulting behaviour. This can bee seen as the application of the self-organisation principle mentioned above. Note that in this thesis, an evolutionary algorithm is simply used as a learning method, it has nothing to do with the biological evolution (for a theoretical discussion of this approach see [10]).

After this high-level overview of related research areas, the following paragraphs describe the more concrete problems addressed by this thesis. In the experiments, a robot develops reaching and grasping skills which is an important research focus in humanoid robotics [11]. A proposed means to achieve general motor control, and therefore also reaching and grasping skills, is using motor primitives [1]. The idea is that complex behaviours can be divided into simpler parts which can then be used to form other behaviours. These action building blocks are called motor primitives, or simply primitives. They have been given different names in the literature, including behaviour primitives, motor schemas, control modules and movemes [12, 13].

Motor primitives are not only a topic in robotics research: evidence suggests that they also play a role in how humans and other animals control body movement [14, 15, 12]. One such hypothesis is called called *equilibrium point control* and gives a plausible explanation of how the *degrees of freedom problem* could be solved. This problem states that there are redundant degrees of freedom when planning body movements and that the planning of trajectories is therefore non-trivial.

Most of the research on controlling movement of robots with motor primitives uses some form of programming by demonstration [1]. An overview of programming by demonstration is given in [16] and concrete examples of learning motor primitives can be found in [17, 18, 19, 20]. Not only the question how motor primitives can be learned, but also the issue on how to use these primitives to form higher-level behaviour is a challenging task that is addressed in this thesis.

Tani et al. [18] explain the characteristics motor primitives need to have to be truly useful for controlling robots. The most important factor is what they call *organic compositionality*. This means that the primitives must be flexible enough to adapt to different environments: in terms of the object acted upon, its location, and the preceding and subsequent primitives. A motor primitive for grasping, for example, should be able to grasp an apple just as well as a banana. The grasp may also need to be adapted depending on whether the banana is going to be peeled or moved to another place.

1.2 Objectives

There are two main research questions behind the experimental setup. The first is whether it is possible to learn motor primitives with a goal-directed trial and error process. This encompasses the question whether the developed primitives have some of the properties of organic compositionality, especially if it is possible to assemble them to form more complex behaviours. By goal-directed learning,

we refer to a learning process that drives the robot toward the development of actions achieving a certain result. By trial and error, we mean a process that: (i) does not require information on how the goal-directed action should be realised, and (ii) leaves the robot free to determine how to act providing that its action achieves the given goal.

The second question is concerned with the influence of language on the robot's ability to learn the integrated behaviour. More specifically, the influence of the presence of a caretaker giving instructions is examined and compared to the possibility of the robot to talk to itself. The hypothesis behind this is that self talk can be a useful tool for the robot to find and memorise the order and timing of primitives when combining them to new behaviours.

1.3 Methods

In order to achieve the objectives of this thesis and verify the underlying hypotheses, an experimental scenario has been designed in which a relatively complex humanoid robot should develop elementary and integrated behaviours. The robot first learns four motor primitives while a corresponding linguistic instruction (a binary label) is given. Then, the robot is expected to solve a task for which it has to combine the primitives. Different conditions for learning the integrated behaviour are explored. The influence of the presence of continued instruction by a caretaker and the use of a self talk feedback loop allowing the robot to give itself the linguistic instructions it has been given during the initial training is studied.

The robot is controlled through a single neural network. The free parameters of the network (connection weights, biases and time constants) are found through an evolutionary algorithm: the behaviour of the robot individuals is evaluated in a series of trials where a fitness function assigns them a value expressing how close they got to achieving the goal of that trial. The best individuals are retained and are allowed to generate offspring. This algorithm has been chosen because it is a simple implementation of trial and error learning and it has proven useful in similar situations as we will see in the next paragraphs.

1.4 Related Research

This method is an extension of the approach used by Massera et al. [7] where an anthropomorphic robotic arm learns to reach for, grasp and lift objects located in front of it on a table. As in our setup, the robot is controlled by a neural network whose parameters are found through an evolutionary algorithm. The most important difference of their approach is that the task was learned without first learning elementary behaviours. The robot was directly expected to do the whole task, but in one of the experimental conditions, there were linguistic instructions indicating when the type of behaviour should be switched. The *reach* label, for example, was given until the robot's hand was placed above the object and then switched to *grasp* indicating that the robot should now wrap the fingers around the object. The results suggested that the use of these instructions makes it easier to learn the desired task. In fact, the full behaviour was only learned when the linguistic support was available. Their experiments

also showed that it is possible to incorporate different types of behaviours into a single neural network.

This result suggests that several motor primitives could be incorporated into the same control unit, at least if they have some commonalities. A lot of the research to develop motor primitives focuses on having well-defined subsystems that each perform one motor primitive (for an exception see for example [21]). We see an advantage in having a single neural network that can execute multiple primitives. In all of the primitives of our experiment, the robot has to learn to keep the hand close to a specified position. The reaching and remaining and this position has only to be learned once for all the primitives. Additionally, we expect the integration of the primitives to be smoother as the control system will most likely keep a similar posture for the different primitives.

Our approach of learning motor behaviours with a trial and error process has several advantages over learning by demonstration. The most important one is that the learning is by definition goal-directed. A problem that can arise in learning by demonstration is that it is difficult for the robot to know which parts of the demonstrated behaviour is important for achieving the goal and which parts are more coincidental. Another issue is that some crucial aspects of the behaviour might be missed altogether when observing the demonstrator. It is for example very difficult to observe how much force is applied to an object or what the stiffness of a body part is.

Another advantage of our approach is that a change in the robot's morphology is possible without much effort. A robot that is trained on reaching a certain location with its hand can essentially be trained with the same algorithm regardless of the number of joints involved (for example using only the arm or using the upper body too). This also means that an algorithm and fitness function that has proven useful on one robot might easily be adapted for other types of robots. However, as the number of degrees of freedom of the robot and the complexity of the task increases, the fitness calculation that evaluates an individual gets more complicated. The design of such a fitness function can be quite difficult and is not a straightforward process.

With these experiments, we hope to contribute to the ongoing research on whether and how motor primitives can be used to form a complete and adaptable control system for human-centred robots and how they can use language as a cognitive tool. If continued, this line of research might also give valuable feedback to other disciplines looking to find out how motor control is achieved in humans. As the control systems resulting from such research can be explored much more easily than the ones in humans, the workings can be studied and the research in humans can be guided by the findings. The idea that advances in robotics can help understanding humans better is outlined in detail in [22, 2].

Chapter 2

Experimental Setup

This thesis studies how a simulated iCub robot can acquire the ability to produce a complex action, i.e. moving an object to a desired location, after having learned to display more simple elementary actions such as moving the hand to a location or grasping an object. To this end, the experiment is divided into three stages. In the first two, the simplified primitive stage (*simple-prim* stage) and main primitive stage (*main-prim* stage), the robot learns to respond to symbolic linguistic instructions given by a caretaker. For every instruction, the robot learns to execute a subcomponent of the full behaviour with a clearly defined goal. These motor primitives are learned in similar circumstances as they will later be needed when assembled together. The four primitives are the following:

Reach Place the hand above the object

Open Open the hand and align the palm to face downwards

Grasp Close the hand around the object to grasp it

Move Move the grasped object to the specified target area

Once these motor primitives are learned, the experiment progresses to the last stage, the integrated behaviour stage (*integrated* stage). In this stage, the caretaker gives a new linguistic instruction and the robot has to learn to do the full integrated behaviour. Four experimental conditions are compared in this last stage. In some of these conditions, the robot is able to give itself instructions by producing linguistic symbols that have previously been used by the caretaker. The conditions serve to study the effects of this self talking mechanism and of the continued linguistic support by the caretaker on the robot's ability to learn and internalise the integrated behaviour.

In the rest of this chapter explains the detailed. First, the software and hardware used for the experiments are introduced. Then, the robot, its neural network and the evolutionary algorithm is explained. After that, a comprehensive description of the fitness calculation and all the experiment stages and conditions follows. Finally, the tests that are run to evaluate the learned behaviour are presented.

2.1 Simulation Software and Hardware

The experiments are run with the EvoICub software developed at LARAL[1] by Gianluca Massera and collaborators. All of the code is open source and is written in C++ with the help of the Qt framework.[2] There are three main components: the genetic algorithm, the neural network framework and the simulator of the world and the iCub. The last component relies on two external libraries: the *Newton Game Dynamics* physics engine[3] and the iKin kinematics library from the iCub repository[4] for kinematic calculations. EvoICub also has a graphical user interface in which individuals can be loaded and their actions can be inspected.

The simulation of the robot and its environment is divided into steps. Every step corresponds to 50 milliseconds of simulated time (1 second = 20 steps). At every time step the inputs of the neural networks are updated and propagated to the outputs. These are then applied to the robot, changing the velocity of the robot's joints. The calculations of the movement of the robot and the interaction with the environment are done in two different ways. In *kinematic mode* the kinematics library is used and in *dynamic mode* the physics engine. In dynamic mode all objects are simulated as rigid bodies. Gravity as well as the effect of collisions between objects is simulated realistically. The movement of an object is therefore determined by all the forces that are applied to it. In kinematic mode no forces are simulated and collisions have no effect. The movement of an object is simply determined by the velocity that has been set for it. The kinematic mode is of course not suited for a realistic simulation, but the advantage lays in the seven to eight times faster computation time. The use of the simulation modes is explained in the detailed descriptions of the experiment stages in Sections 2.7 to 2.9.

For every experiment that is run with EvoICub one needs to implement a class that sets up the world with all the necessary objects, sets the correct inputs to the neural network, applies the outputs of the network to the robot and determines how the fitness is calculated. This is the part of the code that had to be written for the experiments of this thesis.[5] Additionally, some minor changes to the core software had to be implemented. The specific settings of the genetic algorithm, the types of trials and the fitness functions are stored in a configuration file. There is a separate file that specifies the architecture of the neural network. The code of the experiments has been written to make it easy to quickly try out different settings. This means there are a lot of configuration parameters that can change the experiment in a myriad of ways.[6]

All the experiments are run on the LARAL computer cluster. Every of

[1] Laboratory of Autonomous Robotics and Artificial Life, Institute of Cognitive Sciences and Technologies at the Consiglio Nazionale delle Ricerche

[2] LARAL SVN repository at `http://laral.istc.cnr.it/svnrepos/laral/laral2`, documentation on `http://laral.istc.cnr.it/laral++/farsa/`

[3] See `http://www.newtondynamics.com`

[4] iCub SVN repository at `https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub`, installation instruction on `http://eris.liralab.it/wiki/ICub_Software_Installation`, documentation of the iKin library on `http://eris.liralab.it/iCub/main/dox/html/group__iKin.html`

[5] The code used for the final experiments can be found on `http://laral.istc.cnr.it/svnrepos/laral/laral2/evoicub/experiments/tobiasExperiments/exp4/`

[6] The configuration files used for the different stages and conditions are given on the website, see Appendix A.2

the 10 machines has two quad-core AMD Opteron 64-bit 2.2GHz CPUs and is running OpenSuse. It is possible to run the algorithm in parallel and most experiments are run with three to four threads, depending on the availabilities on the cluster.

2.2 iCub Robot

The robot used in the experiments is a realistic simulation of the humanoid iCub robot [23] shown in Figure 2.1.[7] The iCub is a research robot slightly over a meter in height; about the size of a three and a half year old child. It has a total of 53 degrees of freedom (DOF). During all the experiments only the torso, the left arm and the left hand of the robot is used. All the other parts stay immobile. The torso has three DOF of which the following two are used:

- Torso yaw: rotation of the torso (turning to the sides)
- Torso pitch: extension/flexion of the torso (leaning forwards and backwards)

These two DOF are used because of the rather short arms of the iCub that do not allow it to reach very far. The last DOF that would tilt the torso sideways is not used because the other two are sufficient for the robot to reach the whole area needed for the experiments.

All of the arm's seven DOF are used:

- Shoulder pitch: flexion/extension of the arm (front and back movement)
- Shoulder roll: abduction/adduction of the arm
- Shoulder yaw: rotation around the upper arm's principal axis
- Elbow: extension/flexion of the elbow
- Wrist pronsupination: rotation around the forearm's principal axis
- Wrist pitch: flexion/extension of the wrist
- Wrist yaw: abduction/adduction of the wrist

The hand consists of 5 fingers with 3 phalanges each. These can be moved through the following nine DOF:

- Spreading the index, middle, ring and little fingers apart (abduction/adduction)
- Opposing the thumb to the other fingers
- Extension/flexion of the proximal thumb phalanx
- Extension/flexion of the two distal thumb phalanges
- Extension/flexion of the proximal index finger phalanx
- Extension/flexion of the two distal index finger phalanges

[7] All information on the iCub can be found on `http://www.icub.org/`.

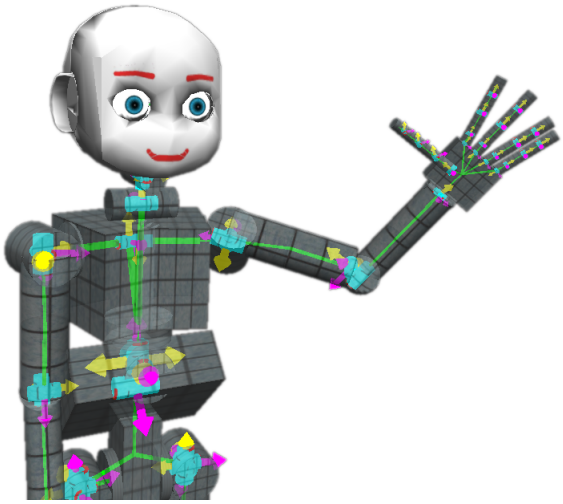


Figure 2.1: The simulated iCub robot

- Extension/flexion of the proximal middle finger phalanx
- Extension/flexion of the two distal middle finger phalanges
- Extension/flexion of all the ring and little finger phalanges

To simplify the learning of the grasp, all the finger joints are actuated through only three DOF. This is enough to ensure a good grasp of a spherical object. The actuated DOF are:

- Opposing the thumb to the other fingers
- Extension/flexion of the all the thumb phalanges
- Extension/flexion of the phalanges of the other fingers

The DOF for the abduction/adduction of the fingers is not used and fixed at maximum abduction.

In total, the robot that is used for the experiments has twelve actuated DOF. All of the arm and hand joints are allowed to move in the full range of motion possible by the physical iCub.[8] The two torso joints are both limited to a range of $[-10, 40]$ instead of $[-50, 50]$ (yaw) and $[-22, 70]$ (pitch). This simplifies the learning process because it eliminates a lot of undesirable positions.

To visualise the different movements of the human body, we recommend the animations on `http://davisplus.fadavis.com/dillon/animations.cfm`.

2.3 The Robot's Neural Network

The artificial neural networks used as the controller of the robot during the first two stages of the experiment is represented in Figure 2.2. The design is

[8] Defined in these files: `https://robotcub.svn.sourceforge.net/svnroot/robotcub/trunk/iCub/main/app/robots/iCubRome01/conf/`

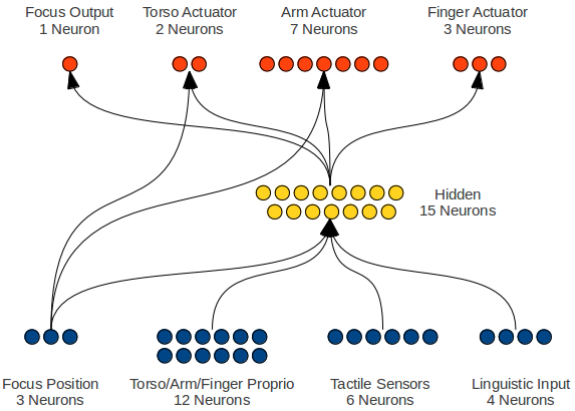


Figure 2.2: Neural network architecture used in the *simple-prim* and *main-prim* stage.

based on the network used by Massera et al. in[7]. The neurons are grouped into clusters. An arrow between two clusters indicates that the neurons of these clusters are fully connected. The architecture of this network can be divided into an input layer, a hidden layer and an output layer. All of the input neurons are connected with all of the hidden neurons, and these are in turn connected with all the output neurons. Additionally, there are two direct connection from the input layer to the output layer that are explained later in this section. The network is a pure feedforward network. All of the neurons in the hidden layer have a bias whose value is decided by the evolutionary algorithm along with all the connection weights. Both the biases and the connection weights take values in the range of $[-1, 1]$. The neurons have a sigmoid activation function: this means that the output value of a neuron is calculated with the help of the following formula:

$$O(t) = \sigma\left(-\beta + \sum_{j=1}^{J} w_j I_j(t)\right)$$
$$\sigma(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2.1}$$

Where β is the bias (0 for the neurons in the output layer), $I_j(t)$ is the output at time step t of the jth connected neuron with w_j the weight of that connection, J is the total number of incoming connections and $\sigma(x)$ is the logistic function with λ set to 1.5.

The output layer has a total of 13 neurons that can be divided into the following four clusters:

Torso Actuator 2 neurons. The desired position of the torso.

Arm Actuator 7 neurons. The desired position of the left arm.

Finger Actuator 3 neurons. The desired position of the fingers.

Focus Output 1 neuron. Binary output that determines whether the robot focuses its attention on the object or on the target area.

The input layer consists of 25 neurons in four distinct clusters:

Proprioception 12 neurons. Represent the current positions of all the used joints.

Tactile Sensors 6 neurons. Six binary inputs specifying which of the tactile sensors on the hand are in contact with an object.

Focus Position 3 neurons. Coordinates relative to the palm of the current focus (either the object or the target area).

Linguistic Input 4 neurons. Binary input neurons that encode whether the caretaker is producing the word *reach*, *grasp*, *open* or *move*.

The hidden layer consists of one cluster of 15 biased neurons.

In the *integrated* stage the neural network is expanded in different ways depending on the condition. This will be explained in Section (2.9). In the following sections, the use of the input and output clusters of the base network is described in more detail.

2.3.1 Torso, Arm and Finger Actuators

The joints are controlled by setting a velocity based on the difference between the current and the desired joint angle. There is one output neuron per actuated DOF. The output takes a value between 0 and 1 and is taken as the desired joint position. The current angle is then linearly mapped to the same range (0 meaning the minimum possible angle, 1 the maximum). The velocity is then calculated as $\alpha \times (desiredValue - currentValue)$ $[degrees/s]$ and the absolute value is not allowed to exceed 40 degrees/s. The parameter α has a value of 100 when in dynamic mode and 37 in kinematic mode. The value in kinematic mode was found empirically and chosen such that the resulting movement is as similar as possible to the simulation with the physics engine.[9]

2.3.2 Focus Output

This single neuron determines which of the possible positions is given as the focus position input. The output is between 0 and 1 but it is interpreted as a binary output: when the value is smaller than 0.5 the focus is on the object and otherwise on the target area.

The focus output can be conceptualised as moving the head to face towards an area and focusing the eyes on it. As this is not a central aspect of these experiments, the described simplification is used instead.

2.3.3 Proprioception Sensors

For every of the twelve actuated DOF of the robot, the neural controller has an input neuron that senses the current position of the corresponding DOF. All the values are linearly mapped between 0 and 1. For the DOF of the hand that

[9] Movement tends to be slower in dynamic mode because of inertia.

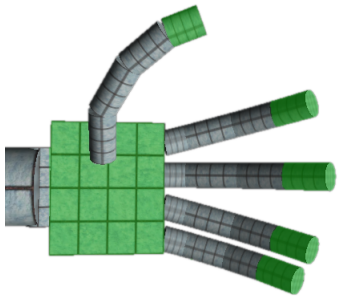


Figure 2.3: The green areas represent the locations of the six touch sensors

take together multiple actual DOF of the robot, the average of all the underlying positions is taken. Note that these in any case have almost the same value as long as there is no external force applied to the fingers. Brief tests have shown that the performance of the network is not improved if the individual positions of all DOF of the hand are given separately (results not shown).

2.3.4 Tactile Sensors

The physical iCub has six groups of tactile sensors: one on the inner tip of every finger and one distributed over the surface of the palm. The sensors on the fingertips respond to pressure, whereas the one on the palm detects contacts with electrical conductors. The tactile sensors are modelled as binary inputs in the simulation. The sensors on the physical robot have a higher resolution than that but the sensitivity is fairly low. Instead of modelling these sensors accurately, the simpler solution of binary sensors is chosen. This information is sufficient to carry out the grasping task. It would also make it more possible to use the controller on the physical robot as it would be easy to convert the actual output to something binary. To sum up, there are six neurons that take a value of either 0 or 1 if the corresponding part of the hand is in contact with another object. The exact locations of the touch sensors are shown in Figure 2.3.

2.3.5 Focus Position

These three neurons give the network the coordinates of the position it is currently focusing on relative to the position of the palm. This means that the input is (0, 0, 0) if the centre of the palm is exactly at the focus position. The coordinates are given in metres scaled by a factor of 20. So a distance of 5 centimetres results in an input of 1. For most parts of the experiment, this distance is usually not more than 30 centimetres. But in the beginning of the evolutionary process it can be more than twice that when the hand moves away from the focus position. The larger possible magnitude compared to the proprioception and tactile sensors was chosen to increase the impact of these neurons. The focus position is a very crucial input and the number of neurons is fewer compared to the proprioception and tactile sensors.

There are two possible positions that the robot can focus on: the object or the target area. Where the focus lies is decided by the *Focus Output* neuron described above (Section 2.3.2). When the object position is given as input it is the actual position of the object at that time step. If the object is moved, the input changes accordingly. The target area is a location that is fixed during the trial. The different target areas and object starting positions are explained in Section 2.5. There is no meaningful *Focus Output* at the first step of every trial and therefore (0, 0, 0) is given as input.

The *Focus Position* cluster has a direct connection to the *Torso* and *Arm Actuator* clusters. These connections make it easier to find a mapping between the position that the palm should be at (which is always near the *Focus Position*) and the position that the hand is moving towards (which is defined by the motor neurons of the torso and arm).

2.3.6 Linguistic Input

These neurons encode which of the instructions the caretaker is currently giving. There is one neuron for every of the four instructions *reach*, *open*, *grasp* and *move* corresponding to the primitives. The input is binary and scaled by a factor of 10 for the same reasons as the Focus Position input. When the *reach* instruction is given, for example, the input of the four linguistic input neurons is set to $(10, 0, 0, 0)$.

2.4 Training Algorithm

From the point of view of the evolutionary algorithm, a robot individual is represented as a sequence of bits. This sequence is called a genotype and it is divided into genes of 16 bits. Every gene represents one free parameter of the neural network and is decoded to a value in the range $[-1, 1]$. A group of genotypes is called a genome. In every cycle of the algorithm, all the genotypes are evaluated in a series of trials. The genotypes are assigned a fitness based on how well the robot controlled by the neural network achieves the desired behaviour. The genotypes with the highest fitness are allowed to continue to the next cycle and to generate offspring. A cycle of the algorithm is also referred to as a generation. The algorithm can be summarised as follows:

1. Initialise genome with 100 random genotypes (decided value of every bit with a fair coin flip)
2. Evaluate all genotypes to get their fitness
3. Stop if desired generation or termination condition reached
4. Take the 10 best individuals and insert them unchanged into the genome of the next generation
5. Generate mutated offspring of the 20 best individuals until the genome of the next generation contains 100 genotypes
6. Go to step 2.

Mutated offspring is created by taking the genotype of the parent and flipping every bit with a certain probability. This probability, the mutation rate, is set to 0.005 in the *simple-prim* and *main-prim* stage and to 0.04 in the *integrated* stage. A higher mutation rate is chosen for the last stage because the number of genes which are mutated is much smaller.

The termination condition in the *simple-prim* and *main-prim* stage is that five of the genotypes of the genome yield individuals that are successful in all of the trials. The evolution is also stopped when the two stages do not terminate within a total of 1500 generations. The *integrated* stage is always run for 100 generations.

In the *simple-prim* and *main-prim* stage, every robot individual has a genotype that is made up of 612 genes; 25×15 for the weights of the connections from input to hidden layer, 15×13 for the connections from hidden to output layer, 3×9 for the direct links from input to output layer and 15 for the biases of the hidden layer. This results in a total of 9792 bits per genotype. Because of the changing neural network in the *integrated* stage, new genes are added to the genotype before that stage is started. The details are explained in Section 2.9.2.

The experiment up to the end of the *main-prim* stage is repeated 10 times; each run with a different random starting genome and a new seed for the random number generator. In the *integrated* stage only the newly added parts of the network are evolved. This means that individuals that have not learned the primitives at the end of the *main-prim* stage will never learn them. In order to have as many individuals as possible that have learned the primitives when starting the *integrated* stage, a new genome is created consisting of the top 10 genotypes of all the runs. The experiment is then continued 10 times for each of the four conditions. The difference between these runs is the new random initialisation of the added genes of the genotypes.

2.5 The Environment

The environment consists of a table with an spherical object on it. The tabletop measures 50 x 80 cm and is placed on the height of the pelvis of the robot. Additionally, there is a designated target area where the object has to be moved to. The position of the object on the table, the target location and the posture of the robot varies from trial to trial. All the starting positions are changed in a round-robin manner. This means that the combination of the positions of the different elements is the same for every individual in every evaluation.

2.5.1 The Object and Its Initial Positions

The object is a sphere with a mass of 200 grams and a diameter of 7 cm with a flattened base to prevent it from rolling away easily. The base is a cylinder with a diameter of 3.5 cm whose bottom side is flush with the lowest point of the sphere. Every primitive is tested with four different object positions. The four positions are 10 cm to the front, back, left and right respectively of a point 30 cm in front and 10 cm to the left of the robot. During trials that start with the reach primitive there is an additional uniform random offset between -2 and 2 cm added to the x and y dimension (left/right and forward/backward). This

is not done for the other trials because the hand of the robot has to be in a well defined relative position to the object. The object position can be seen on the first row of Figure 2.4.

2.5.2 Target Area

There are two different target positions. Both are used twice for every primitive. They are 25 cm above the table and 25 cm in front of the robot. The height corresponds about to the height of the shoulder of the robot. One position is 10 cm to the left, the other to the right of the central axis of the robot. Additionally there is an offset between -2 and 2 cm added to all three dimensions that is determined uniform randomly for every trial. The target is considered reached if the object is within 6 cm of this point. The bright green circles in the last row of Figure 2.4 show the two target areas in context.

2.5.3 iCub Initial Postures

Every primitive requires different types of starting postures of the robot. There are four postures for every of the four primitives, one for every of the 16 trials of the *simple-prim* and *main-prim* stage.

Reach Four different arm postures away from the object, torso always completely centred and fingers always completely stretched out

Open In two of the trials the hand is almost grasping the object, in the other two the hand is a few centimetres above the object with almost completely closed hand. In all of the trials the hand is already above the object

Grasp The hand is a few centimetres above the object. In two of the trials the fingers are completely stretched out in the other two they are a bit less stretched out. In all of the trials the hand is already above the object

Move The hand has a good grasp of the object such that it does not fall out if the fingers are not moved.

The initial postures for the reach trials are also used for the trials of the *integrated* stage. Images of the starting postures are shown in Figure 2.4.

2.6 Fitness Calculation

As explained in Section 2.4, every individual is evaluated in a series of trials. These trials serve to establish the fitness based on which the selection of the best individuals of a generation is done. The calculation changes for the different experiment conditions, experiment stages, and trials. All of the fitness calculations are based on three components: the step fitness $S(t)$, the penalty $P(t)$ and the bonus B, where t is the time step indicating that this components is calculated and awarded in every time step of the simulation. $S(t)$ takes values between 0 and 1 depending on how close the robot is to the goal of that particular trial or how "correct" the current outputs of the neural network are. The function is mostly continuous, has only small discontinuities and it is almost impossible for the robot to be awarded the maximum or minimum value. That nature of the

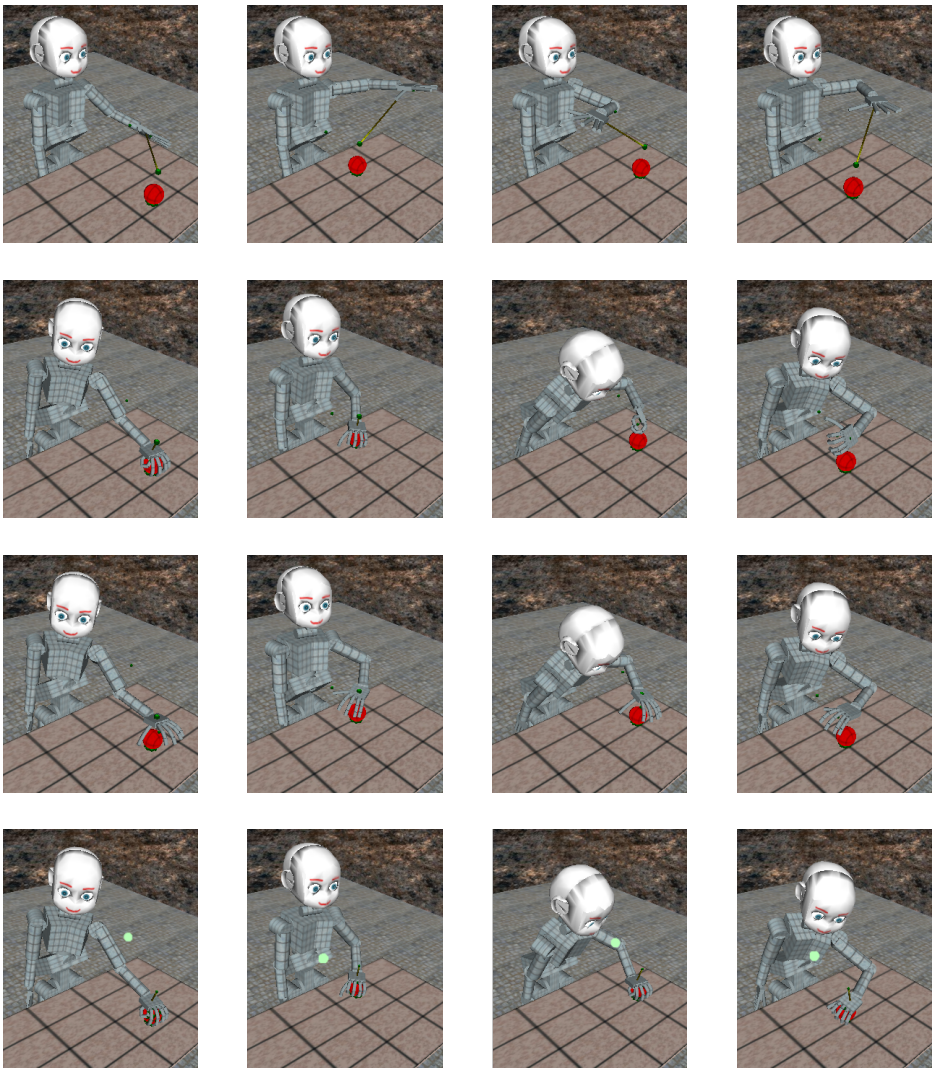


Figure 2.4: The 16 starting postures, each column corresponds to one of the primitives (reach, open, grasp, move).

step fitness function is important because it makes it difficult for two individuals to receive the same final fitness which is needed to make a sensible selection of the best individuals.

The penalty $P(t)$ takes values between 0 and 4, although the maximum possible penalty depends on the trial type. The penalty is subtracted from the fitness received in that time step. A penalty is given if the robot does something that it is not supposed to do in that trial but that does not necessarily have a direct effect on the outcome of the trial. The penalty function is discontinuous and its value is mostly 0.

The last component B is a bonus that is given based on the the overall behaviour in a trial. Fixed amounts of bonus are awarded if the robot fulfils certain conditions during the trial. The bonus is substantially higher than the fitness that can be received every time step to clearly distinguish between individuals that were successful in a trial from the ones that were not. It also makes sure that individuals that achieved the goal are selected even if their approach to do so was considered worse than the one of other individuals.

How these components are calculated exactly and how they are taken together for the total fitness is explained in Section 2.7 to 2.9.

2.7 The *Simple-Prim* Stage

In the *simple-prim* stage, the robot has to learn simplified forms of the actual primitives. These are primitives in which some of the constraints have been dropped to get the robot to learn roughly what it will be expected to do later.

This stage was introduced due to the lack of satisfactory results in preliminary experiments where the primitives were learned directly. The inspiration for this incremental learning process with changing constraints came from the discussed principles of ontogenetic development. The grasp and move behaviours were the primitives that seemed the most difficult in the preliminary experiments. That is why these were simplified by ignoring the interaction with the object. This is done by running the experiment in kinematic mode, meaning that there is no actual simulation of the physics and therefore of collisions. The only interaction between the robot and the environment that is already learned, is to avoid touching the table and the object during certain primitives. An additional advantage that the introduction of the *simple-prim* stage brought, was the reduction of the simulation time due to the faster kinematic calculations.

The fitness of the individuals is evaluated in 16 trials: four for each of the primitives. The trials differ in the starting posture of the robot, the position of the object, the location of the target area, the linguistic instruction given and of course the fitness function. Every trial lasts three simulated seconds (corresponding to 60 time steps). This was found to be enough time to complete any of the primitives from all of the starting positions.

The rest of this section explains the exact circumstances of every trial and gives the details on the fitness calculation.

2.7.1 Fitness Calculation

The fitness has four components, one for each of the four primitives. These values, denoted $F_p,\ p \in \{\hat{r}, \hat{o}, \hat{g}, \hat{m}\}$ (simplified reach, open, grasp and move),

represent how successful an individual has executed the individuals primitives. The final fitness is the harmonic mean of these four values:

$$F = \left(\sum_{p \in \{\hat{r}, \hat{o}, \hat{g}, \hat{m}\}} \frac{1}{F_p} \right)^{-1} \tag{2.2}$$

The harmonic mean is used rather than the normal arithmetic mean to make sure that the four primitives are learned in parallel. Tests comparing the two mean types showed that with the arithmetic mean it can happen that individuals get quickly very good at some of the primitives and are then too specialised to learn the other primitives. The harmonic mean gives more emphasis on the lowest component; a small improvement of the lowest value increases the average fitness to a greater extent than a bigger improvement of the higher values.

The components $S(t)$, $P(t)$ and B introduced in Section 2.6 are now written with a subscript p with $p \in \{\hat{r}, \hat{o}, \hat{g}, \hat{m}\}$ to indicate that their value depends on the primitive being examined. The total primitive fitness values are then calculated with the following formula:

$$F_p = B_p + \frac{100}{N_p} \sum_{t=1}^{N_p} (S_p(t) - P_p(t)) \tag{2.3}$$

Where N_p is the total number of time steps spent evaluating the given primitive. Described in word, the above formula means that the sum of the step fitness values is normalised by the number of steps that has been spent in this primitive and the bonus is added to this. For every primitive there is a success condition. If this condition is met, a bonus of 10 points is awarded. As every primitive is evaluated four times, there is a maximum of 40 bonus points. The success conditions are explained in the following section.

The theoretical maximum of F_p is therefore 140 (100 for receiving maximum step fitness in every step and no penalty as well as 40 for the maximum bonus). This value can however not be achieved in practice as the robot would have to be at the goal state from the beginning and stay there for the whole duration of all the trials.

In the following paragraphs, the different fitness components are explained in more detail.

2.7.1.1 Penalty Fitness $P_p(t)$

During all stages of the experiment there can also be a penalty at every time step. The penalty is a value that is subtracted from the fitness received in that step. In the *simple-prim* stage there are the following two kinds of penalties:

Table Touching 1 penalty point if any part of the left arm of the robot is in touch with the table.

Object Touching 1 penalty point if the robot touches the object.

The *Object Touching* penalty is not given during all the primitives. Which penalties are given and when they are applied is explained in the next section.

2.7.1.2 Trial Types and Step Fitness $S_p(t)$

The following values are used for the calculation of the fitness values (also in later stages):

$d_r(t)$ Distance in metres between the palm of the hand and a point 6 centimetres above the object.

$d_g(t)$ Distance in metres between the centre of the object and the centroid of the tip of the thumb, the tip of the pinkie and the centre of the palm.

$d_{\hat{m}}(t)$ Distance in metres between the palm and the centre of the target area.

$d_m(t)$ Distance in metres between the object and the centre of the target area.

$h(t)$ A measure of how open the hand is, 0 for completely closed and 1 for completely open.

$p(t)$ 1 when the palm is facing towards the table and 0 when it is facing away from it.

$f_f(t)$ A value expressing how correct the current focus is. It is bigger than 0.5 if it is correct and smaller if not.

In all of the trials the most crucial part is to select the correct focus for that primitive. If the focus is wrong, the goal of all the trials can only be achieved by chance. This is why $S_p(t)$ is calculated in the following way:

$$S_p(t) = \frac{1}{5} f_f(t) + \begin{cases} \frac{4}{5} s_p(t) & focusCorrect \\ 0 & \neg focusCorrect \end{cases} \tag{2.4}$$

Where $s_p(t)$ is the part of the step fitness that is variable between the different primitives. In the next paragraphs this and other details of the different trial types are explained.

Simplified Reach Trials

Goal Move the hand 6 cm above the object (meaning 9.5 cm above the object position because the sphere has a radius of 3.5 cm).

Correct Focus Object position

Linguistic Input REACH: (10, 0, 0, 0)

Step Fitness

$$s_{\hat{r}}(t) = e^{-10 d_r(t)} \tag{2.5}$$

Penalty *Table Touching*, *Object Touching*

Success Condition $d_r(t) < 0.03$ at any time during the trial

Simplified Open Trials

Goal Keep the hand 6 cm above the object while stretching out all the fingers and aligning the hand to face downwards.

Correct Focus Object position

Linguistic Input OPEN: (0, 10, 0, 0)

Step Fitness

$$s_{\hat{o}}(t) = \frac{1}{2}\left(\left(1 - \frac{|h(t) - 0.7|}{0.7}\right) + p(t)\right) e^{-10 d_r(t)} \tag{2.6}$$

Penalty *Table Touching*, *Object Touching*

Success Condition $d_r(t) < 0.03 \wedge h(t) > 0.7 \wedge p(t) > 0.9$ at any time during the trial

Simplified Grasp Trials

Goal Close the hand around the position where the object is.

Correct Focus Object position

Linguistic Input GRASP: (0, 0, 10, 0)

Step Fitness

$$s_{\hat{g}}(t) = \frac{1}{2}\left(\left(1 - \frac{|h(t) - 0.4|}{(1 - 0.4)}\right) + p(t)\right) e^{-10 d_g(t)} \tag{2.7}$$

Penalty *Table Touching*

Success Condition $d_g(t) < 0.03 \wedge h(t) < 0.4 \wedge p(t) > 0.9$ at any time during the trial

Simplified Move Trials

Goal Move the hand to the target position.

Correct Focus Target position

Linguistic Input MOVE: (0, 0, 0, 10)

Step Fitness

$$s_{\hat{m}}(t) = e^{-10 d_m(t)} \tag{2.8}$$

Penalty *Table Touching*

Success Condition $d_{\hat{m}}(t) < 0.03$ at any time during the trial

2.8 The *Main-Prim* Stage

In the *main-prim* stage, the learned simplified primitives are refined to the actual primitives. For the grasp and move primitives the changes are quite substantial, whereas the reach and open primitives stay basically the same. But because the simulation is now done in dynamic mode, adaptations are necessary for all of the primitives.

The trials in this stage are very similar to the ones in the *simple-prim* stage. The trial duration is chosen more carefully however, because of the slower simulation speed in the dynamic mode. The duration now depends on which primitive is evaluated. It is as short as possible while still allowing the primitive at hand to be completed. To reduce the simulation time further, some trials are terminated before the full duration if the behaviour either clearly failed or succeeded. Details on when trials are terminated prematurely are given further on in this section.

2.8.1 Fitness Calculation

The fitness is calculated according to the same formulas (2.2) and (2.3) as in the *simple-prim* stage except that the subscript p now stands for the non-simplified primitives, i.e. $p \in \{r, o, g, m\}$.

2.8.1.1 Penalty Fitness $P_p(t)$

Apart from the two components already explained in the *simple-prim* stage (Section 2.7.1.1), there is one new type of penalty:

Object Moving Up to 2 penalty points for moving the object from its starting position. The current distance in decimetres from the starting position is given as penalty, although movement by less than 2 cm is not punished.

This penalty is not used for the simplified primitives as the object can not be moved because collisions between objects have no effect.

2.8.1.2 Trial Types and Step Fitness $S_p(t)$

$S_p(t)$ is calculated with the same formula (2.4) as in the *simple-prim* stage. In the following paragraphs $s_p(t)$ for $p \in \{r, o, g, m\}$ and all the other differences to the corresponding simplified trials of the *simple-prim* stage are explained.

Reach Trials

Trial Duration 5 seconds

Premature Trial Termination If the object falls off the table (maximum penalty of -4 given for all remaining steps) or if the point above the object has been reached more than 20 steps (1 second) ago (maximum fitness of 1 given for all remaining steps).

Step Fitness

$$s_r(t) = s_{\hat{r}}(t) \text{ defined in (2.5)} \tag{2.9}$$

Penalty *Table Touching*, *Object Touching*, *Object Moving*

Open Trials

Trial Duration 3 seconds

Premature Trial Termination If object falls of the table (maximum penalty of -4 given for all remaining steps).

Step Fitness

$$s_o(t) = s_{\hat{o}}(t) \text{ defined in (2.6)} \tag{2.10}$$

Penalty *Table Touching*, *Object Touching*, *Object Moving*

Grasp Trials To test for a successful grasp, the collision between the object and the table is turned off after 40 steps (2 seconds). The trial lasts for another 2 seconds after that to make sure the robot is holding the object in a stable way.

Goal Grasp the object and hold it in place.

Trial Duration 4 seconds

Premature Trial Termination If object falls under the table surface (maximum penalty of -4 given for all remaining steps if this happens in the first 40 steps, otherwise no penalty given).

Step Fitness

$$s_g(t) = s_{\hat{g}}(t) \text{ defined in (2.7)} \tag{2.11}$$

Penalty *Table Touching*, *Object Moving* (only for the first 40 steps)

Success Condition If the centre of the object is still above the table at the end of the trial.

Move Trials After 10 steps the collision between the object and the table is turned off to make sure that the individuals that keep holding the object are clearly better than others.

Goal Move the object to the target location.

Trial Duration 3.5 seconds

Premature Trial Termination If object falls of the table (maximum penalty of -4 given for remaining steps).

Step Fitness

$$s_m(t) = \begin{cases} \frac{1}{10} & touching \\ 0 & \neg touching \end{cases} + \begin{cases} \frac{1}{10} + \frac{8}{10}e^{-10d_m(t)} & lifted \\ 0 & \neg lifted \end{cases} \tag{2.12}$$

Where *touching* is true if if one of the fingertips or the palm of the robot is in contact with the object and *lifted* is true if the object is touching the robot but not the table.

Penalty *Table Touching*

Success Condition $d_m(t) < 0.06$ at any time during the trial and the object is still *lifted* at the end of the trial

2.9 The *Integrated* Stage

The goal of the *integrated* stage is to produce individuals that respond to the new instruction *move-object-to-target* by completing the whole task of reaching for, grasping and moving the object to the target location without any external help. The experiment at this point is continued in four different conditions. In all of the conditions, new parts are added to the neural network and the part that was evolved so far is no longer changed.

In the simplest experimental condition, the *NI* condition (No Instructions), the robot is trained to perform the integrated behaviour without receiving any linguistic input (except for the mentioned *move-object-to-target* instruction). In this condition, the robot can rely on previously learned capabilities but does not receive any indication from the caretaker on how this capabilities should be used. An additional set of internal neurons with recurrent connection is added to the neural network. This can be seen as a form of memory unit that should provide the necessary computational resources to learn how to produce the integrated behaviour. The architecture of the network is represented in Figure 2.5b.

In half of the trials of the *CT* (Caretaker Instructions) condition, the robot perceives the language labels produced by the caretaker that indicate the action that should be performed in any particular moment and consequently also the conditions in which to switch from the execution of one action to the execution of the next action. The other half of the trials are the same as in the *NI* condition, as is the architecture of the neural controller.

In the *ST* condition (Self-Talk Instructions) the robot is provided with a different neural architecture shown in Figure 2.5a. There are four linguistic output units that can be used to self-generate the labels used by the caretaker. At every time step t a label is generated and it is given to the linguistic input unit at time $t + 1$. The robot can therefore learn to give itself the instructions that lead to a successful integrated behaviour.

In the fourth and last condition, the *ST-CT* condition (Self-Talk and Caretaker Instructions), the neural architecture is the same as in the *ST* condition. The self-generated labels are however only used in half of the trials. In the other half of the trials, the caretaker provides the instructions as in the *CT* condition. In these latter trials, the robot is not rewarded for producing the integrated behaviour, but for producing the linguistic label that corresponds to the next label given by the caretaker. The robot therefore learns the sequence of linguistic instructions and learns to predict an upcoming switch from one label to another.

Note how in all conditions the robot should be able to produce the integrated behaviour without the help of the caretaker during the second set of trials, or in all the trials in the *NI* and *ST-CT* conditions. In other words, the robot can in some conditions exploit the input of the caretaker to learn to generate the integrated behaviour. But it always also has to be able to act autonomously, without any support from the caretaker.

In this stage, individuals are evaluated in eight trials. Each of the four object positions are used twice, every time with a new random offset. The two target area locations are each used four times, also every time with a random offset. The robot starts the trials twice in each of the four reach starting postures described in Section 2.5.3. The *Integrated Behaviour Fitness* function is used in all trials of the *ST*, *CT* and *NI* conditions as well as in the odd trials of the

Condition	Neural Network	Fitness Function		Linguistic Input	
		Even Trials	Odd Trials	Even Trials	Odd Trials
ST-CT	Type A	Prediction	Integrated	Caretaker	Self Talk
ST	Type A	Integrated	Integrated	Self Talk	Self Talk
CT	Type B	Integrated	Integrated	Caretaker	None
NI	Type B	Integrated	Integrated	None	None

Table 2.1: Experiment conditions

ST-CT condition. The *Prediction Fitness* function is used in the even trials of the *ST-CT* condition.

All trials last for a maximum of twelve seconds. A trial is terminated before that if the object has been moved to the target more than 20 steps (1 second) ago and the object is still held in the hand. In this case, the maximum fitness is given for all the remaining steps in that trial.

A summary of the differences between the condition is given in Table 2.1.

2.9.1 Neural Network Architecture

The structure of the the two types of network architectures are shown in Figure 2.5. In both of the networks there is one new input neuron called *Secondary Linguistic Input*. It works like the other linguistic input neurons and its activation indicates that the caretaker is giving the *move-object-to-target* instruction. Whether or not this input is set does not change the experiment in a meaningful way. It was nevertheless included to show how one might have multiple integrated behaviours.

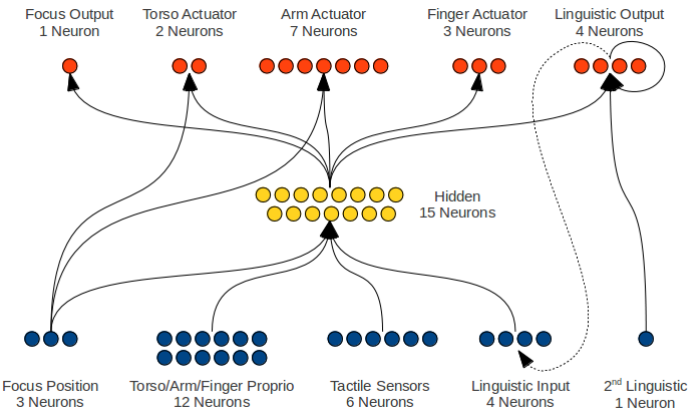
There is another new cluster of four neurons that serves a different purpose in the two types of networks. In architecture A, used in the *ST* and *ST-CT* conditions, these are neurons in the output layer called *Linguistic Output*. Each neuron corresponds to one of the linguistic labels for the primitives. The neuron with the highest output determines the label that is currently produced by the robot. This means the robot is producing exactly one label at every time step. In architecture B, used in the *NI* and *CT* conditions, these neurons represent a memory unit called *Hidden Linguistic* cluster, that has direct connections to and from the main *Hidden* cluster. In both types of networks this cluster has recurrent connections and there is an incoming connection from the *Hidden* cluster. Also, these neurons have a form of memory that is implemented by the following activation function:

$$O_m(t) = \gamma\left(\delta O(t) + (1-\delta)O(t-1)\right) \tag{2.13}$$

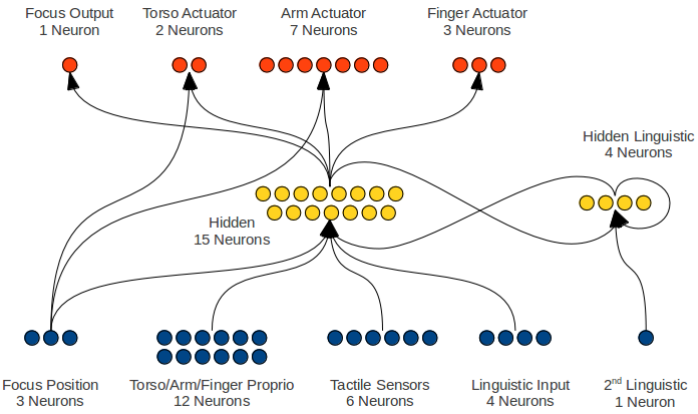
Where $O(t)$ is the function described in formula (2.1) and δ is a decay factor in the range $[0, 1]$ whose value is found through the evolutionary algorithm. The factor γ is 1 in the type A network and 10 in type B. The factor γ is used to give the *Hidden Linguistic* cluster the same weight as the *Linguistic Input* cluster.

2.9.2 Expanded Genotype

The genotypes are expanded in this stage because of the new parts in the neural network. In the *ST-CT* and *ST* conditions there are 84 new genes (1344 bits):



(a) Type A neural network architecture used in the *ST-CT* and *ST* conditions.



(b) Type B neural network architecture used in the *CT* and *NI* conditions.

Figure 2.5: The two neural network architectures in the *integrated* stage.

15×4 for the connections between the *Hidden* cluster and the *Linguistic Output* 4×4 for the recurrent connections of the *Linguistic Output*, 4 for the decay factor in the *Linguistic Output* neurons and 1×4 for the connections between the *Secondary Linguistic Input* and the *Linguistic Output*. In the *CT* and *NI* conditions, 144 new genes are added (2304 bits): the same as for the *ST-CT* and *ST* conditions plus another 15×4 for the connections between the *Hidden Linguistic* cluster and the *Hidden* cluster. Together with the 612 genes from the rest of the network, the genotype now consists of 696 or 756 genes respectively.

The newly added genes are initialised to a different random value in every individual of every run of the experiment. The genes encoding the weights from the *Hidden Linguistic* cluster to the main *Hidden* cluster in the network of the *CT* and *NI* conditions are initially set to zero (otherwise these connections would interfere with the learned primitives).

2.9.3 Linguistic Instructions

In all of the trials the new instruction *move-object-to-target* is given by the caretaker. In some of the trials of the *ST-CT*, *ST* and *CT* conditions the robot also gets the instructions from the previous stages. This can either be from the caretaker or by the robot itself through the explained self talk mechanism.

In even trials of the *ST-CT* and *CT* conditions, the caretaker gives a sequence of linguistic instructions that should lead to the successful completion of the integrated behaviour. The sequence of instructions given is *reach*, *open*, *grasp*, *move*. The label is switched to the next one as soon as the success condition used in the *main-prim* stage holds for the current primitive. The grasp success condition is changed because the above defined condition does not make sense here (the collision between the object and the table is never turned off). The grasp is now considered successful if three or more of the touch sensors are in contact with the object.

The caretaker can jump a label if the success condition is already fulfilled the first time that this label would have been given. In practice this only happens with the *open* instruction. Further, note that the caretaker never gives an instruction in the very first step of a trial. This is necessary to let the robot in the *ST-CT* condition learn that the correct self talking output in the beginning of a trial is *reach*.

2.9.4 Fitness Calculation

2.9.4.1 Integrated Behaviour Fitness

The integrated behaviour fitness is made up of components used in the *main-prim* stage to calculate the different primitive fitness values. The fitness function is built in a way that all individuals always receive some fitness and that makes minimal assumptions about how the goal of moving the object to the target location should be achieved. As in the previous stages, the most important thing for the network to learn is to focus on the right location at the right time. This is why the step fitness is again calculated with formula (2.4). The focus must be on the target if the object is *lifted* and on the object otherwise. $s_p(t)$ is the same in every trial and it is therefore written as $s_i(t)$. Its value is calculated with the following formula:

$$s_i(t) = \frac{1}{2}s_m(t) + \frac{1}{4}\begin{cases} s_o(t) & \neg opened \wedge \neg lifted \\ 1 + s_g(t) & opened \wedge \neg lifted \\ 2 & lifted \end{cases} \tag{2.14}$$

Where $s_o(t)$, $s_g(t)$ and $s_m(t)$ are defined in equations (2.10), (2.11) and (2.12) respectively. *Opened* is *true* if at any time during that trial the success condition of the open trial was fulfilled.

Penalty Fitness The *Table Touching* penalty is given during the whole trial. The *Object Moving* penalty is given as long as the object is not *lifted*. The total penalty received in a trial can not exceed the summed step fitness in that trial. That means it is not possible to get negative total fitness.

Bonus Fitness There are two types of bonus given in the integrated behaviour fitness. The first bonus is given if the object is *lifted* and the focus has been on the target area for at least three time steps. The second bonus is awarded when the object has been successfully moved to the target location (same condition as for the move primitive of the *main-prim* stage). Both bonuses are worth 10 points, resulting in a total possible bonus of 20 per trial.

No bonus is given for reaching to the object or opening the hand because this is not strictly necessary to achieve the goal. The object could be reached from the side with the hand only just as opened as it is needed to fit the object in the hand. In this situation, the hand might never cross the point above the object that was defined as a successful reach in the *simple-prim* and *main-prim* stage. Also no bonus is given for touching or grasping the object because it is difficult to define a success condition for the grasp that is always fulfilled on the way to achieving the goal.

First Generation Fitness A slightly modified version of the the *Integrated Behaviour Fitness* is used in the very first generation in all conditions but *NI*. The bonus fitness is increased by a factor of 10 and all the trials are done with the caretaker instructions. This small optimisation helps to select the individuals for which the combination of the primitives works best before starting to evaluate the capability of the individuals to perform the integrated behaviour on their own. In the *NI* conditions, this is not necessary as the individuals are never given any primitive linguistic instructions.

2.9.4.2 Prediction Fitness

The prediction fitness is of completely different nature than any of the other fitness functions. Instead of judging the robot based on the results of its actions, the output of the neural network is scrutinised more directly. Specifically, the values of the *Linguistic Output* neurons is analysed. The prediction fitness is used when the caretaker provides the instructions in the *ST-CT* condition. The goal of this fitness is to get the robot to correctly predict the next linguistic instruction that it will be given. An instruction is considered correctly predicted if the corresponding linguistic output neuron has the highest value in the previous time step. The linguistic instruction only changes a few times; most of the time the next instruction will be the same as the current one. This is

why the prediction fitness emphasises the time before an instruction switch the most. The three components that take part in the calculation are explained in the following paragraphs. For easier understanding, note that all functions called f are normalised in the range $[0, 1]$.

Step Prediction This part of the fitness is awarded at every time step. It rewards the correct prediction of the instruction of the next step. To get the maximum fitness in a step, the correct neuron must not only have the highest value but also be at least 0.25 higher than all the other outputs (all the outputs are between 0 and 1). This helps to make the dominant output more robust. Additionally, there is some fitness given to reward individuals that do not predict the correct instruction but that are close to it. The amount depends on the difference between the value of the neuron with the highest output and the value of the neuron that should have the highest output. The exact formula evaluated at every time step is the following:

$$f_{step}(t) = \begin{cases} \frac{3}{4} + min\left(\frac{1}{4}, o(p_c(t), t) - h_2(t)\right) & c(t) \\ \frac{1}{4} d_1(p_c(t), t) & \neg c(t) \end{cases} \tag{2.15}$$
$$d_1(p, t) = 1 - (h_1(t) - o(p, t))$$

Where t is the time step, $c(t)$ indicates whether the prediction is correct at time step t, $p_c(t)$ is the primitive that should be predicted at that step, $o(p, t)$ is the output of the neuron corresponding to primitive p, $h_1(t)$ is the value of the highest output and $h_2(t)$ of the second highest output. $d_1(p, t)$ is a measure on how close the output corresponding to primitive $p \in \{r, o, g, m\}$ is to being the highest output. The step prediction fitness for a trial is the average over all steps:

$$F_{step} = \frac{100}{N} \sum_{t=1}^{N} f_{step}(t) \tag{2.16}$$

Switch Prediction If only the step prediction fitness would be used, the robot would not learn to predict the switches from one instruction to another. The switch prediction fitness is responsible to change this. It analyses the linguistic outputs in the 20 steps (1 second) before an instruction switch and rewards the robot for anticipating an upcoming switch. In the very first step of a trial the only linguistic input that is given is *move-object-to-position*. The robot needs to learn that the first self talk instruction of a trial is *reach*. The first part of the switch prediction rewards the correct prediction of this first instruction.

$$f_{switch}(p|p = r) = \begin{cases} 1 & c(1) \\ \frac{1}{2} d_1(r, 0) & \neg c(1) \end{cases} \tag{2.17}$$

For all the other switches, fitness is given if the output of the neuron that corresponds to the upcoming instruction gets closer to being the highest output. As the output of the linguistic neurons before a switch does not usually change a lot, this change will be close to zero often. A sigmoid function is used to distinguish more between small changes:

$$f^i_{switch}(p) = \left(1 + exp\left(d_1(p, t_s(p)) - d_1(p, t_s(p) - 20)\right)\right)^{-1} \tag{2.18}$$

Where $t_s(p)$ is the time step just before the switch to primitive p happens. It is not enough however to increase the correct output relative to the highest output, but the prediction has to be correct at the switch step:

$$f^{ii}_{switch}(p) = \begin{cases} 1 & c(t_s(p)) \\ d_1(t_s(p)) & \neg c(t_s(p)) \end{cases} \tag{2.19}$$

Finally, the prediction should switch as close as possible to the actual switch step:

$$f^{iii}_{switch}(p) = \begin{cases} max\left(0, 1 - \frac{1}{20}(t_s(p) - t_{pr}(p))\right) & c(t_s(p)) \\ 0 & \neg c(t_s(p)) \end{cases} \tag{2.20}$$

Where $t_{pr}(p)$ is the time step in which the prediction switched to the correct one before $t_s(p)$. These three parts are taken together (for $p \in \{o, g, m\}$):

$$f_{switch}(p|p \in \{o, g, m\}) = 0.5 f^{i}_{switch}(p) + 0.1 f^{ii}_{switch}(p) + 0.4 f^{iii}_{switch}(p) \tag{2.21}$$

If a linguistic instruction is never given because that primitive was jumped, $f_{switch}(p)$ takes the maximum value of 1 for that primitive. This favours individuals that jump primitives, but is necessary to keep the maximum achievable fitness the same across individuals.

The total switch prediction fitness in every trial is a value in the range $[0, 100]$:

$$F_{switch} = 100 \sum_{p} \frac{1}{4} f_{switch}(p) \tag{2.22}$$

Switch Prediction Bonus There is also a bonus in the prediction fitness. A bonus of 10 is given for every switch that is considered successful and for the correct prediction in the first step.

$$B_{pred} = \begin{cases} 10 & c(0) \\ 0 & \neg c(0) \end{cases} + \sum_{p \in \{o,g,m\}} \begin{cases} 10 & c(t_s(p)) \wedge t_s(p) - t_{pr}(p) < 5 \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

Total Prediction Fitness Taking it all together, the total prediction fitness over all trials is (T being the total number of trials):

$$F_{pred} = B_{pred} + \frac{1}{T} \sum_{trials} \left(\frac{1}{3} F_{step} + \frac{2}{3} F_{switch} \right) \tag{2.24}$$

It it possible that the instructions given by the caretaker do not result in a successful integrated behaviour. In trials where the object is not successfully lifted to the target, no prediction fitness is given as the robot did not see a sequence of instructions that it should learn.

2.9.4.3 Maximum Fitness Values

In the *ST-CT* condition, the *Prediction Fitness* is used for the even trials and the *Integrated Behaviour Fitness* for the odd trials. For the final fitness, the two fitness values are simply added up. This means the maximum possible fitness

is 440 (100 for each of the two fitness types plus $4 \times 20 = 80$ bonus points from four trials with the *Integrated Behaviour Fitness* and $4 \times 40 = 160$ bonus points from four trials with the *Prediction Fitness*). In all the other conditions, the maximum fitness is 260 (100 plus $8 \times 20 = 160$ bonus points).

2.10 Post Evaluation

Several tests are run on the individuals resulting from the training process. The tests are described in the following sections.

2.10.1 Integration Test

After the *main-prim* stage a test is run to verify that the developed primitives can be combined to the desired integrated behaviour. The top 10 individuals of the final generation of every run are tested. This test consists of forty trials in which the caretaker produces the sequence of linguistic trials, as described in Section 2.9.3, that should result in the robot reaching above the object, opening its hand, grasping the object and then moving it to the target. The starting positions of the object, the robot and the location of the target are the same as in the *integrated* stage trials; each of the four types of positions are used 10 times. Trials are considered successful if the object is moved to the targets within twelve seconds.

2.10.2 Robustness Test

To compare the performance of the best individual of the last generation of every experiment run after the *integrated* stage, a test was performed with 40 trials. Each of the four types of object positions and initial postures of the robot was tested 10 times to also account for the fact that some individuals might be able to perform the behaviour more robustly. No support was given by the caretaker in any of the trials as the autonomous behaviour should be tested. This test serves to compare the performance of the individuals of the different conditions.

2.10.3 Generalisation Test

To test whether the learned integrated behaviour is not only working in the training conditions but is more general, the following post evaluation is run. The best individual in the last generation of every run is subjected to this test. Individuals that are not successful in any of the trials without the caretaker during training are excluded. The post evaluation tests the autonomous behaviour, meaning that there is no support by the caretaker. The generalisation of the behaviour with respect to the object position is the main focus and the robot's initial posture the secondary focus of the test. A rectangular area centred in the middle of the previous object positions of the size 35×35 cm is divided into a grid of 7×7 cells. The object is placed in the centre of every cell with a uniform random offset of ± 0.5 cm in both dimensions. The same four starting postures of the robot are used as in the *integrated* stage trials, only that every object position is now tested with every posture. The target area is always kept the same at a position in between the two trained ones. This results in 196 types

of starting situations. Each of those is tested 10 times with a different random offset for the object. The trial duration is increased to 15 seconds, to make sure the there is enough time to move the object for the new positions that are further away. The percentage of successful trials of an individual is taken as a measure of the quality of its integrated behaviour. A success rate of 0 is given to the individuals that were excluded from the post evaluation.

The results of this test help to compare the different conditions and show if any of the conditions yield individuals that are better at applying their learned skills to new situations. This same test is also run separately on all the final individuals of the *ST-CT* and *ST* conditions with the caretaker giving the instructions. These results (labelled as *ST-CT0* and *ST0*) allow to test the hypothesis that self talking individuals can generalise better to new situations than if a caretaker gives the instructions. Further, the time that is needed to move the object to the target will also be compared between these two situations.

Chapter 3

Results

3.1 The *Simple-Prim* and *Main-Prim* Stage

The 10 repetitions of the experiment in the *simple-prim* and *main-prim* stage are referred to as run 0 to 9. All but run 0 terminated successfully before the limit of 1500 generations. The fastest run terminated at generation 246 (run 7) and the slowest of the successful runs at generation 711 (run 8). Run 0 was successful in 15 of the 16 trials and the fitness of the best individual did almost not improve between generations 900 and 1500. The median number of generations needed to converge is 581 (lower and upper quartile at 426 and 700 generations). The *simple-prim* stage was completed by all of the runs, the fastest run got there at generation 177 (run 7), the slowest at generation 876 (run 0). The median for completing the *simple-prim* stage is 484 generations (lower and upper quartile: 363 and 630). The fitness values of the best genotype in every generation of run 2 and 7 are shown in Figure 3.1 . Apart from the total fitness, the summed up bonus fitness is also shown; every 10 points of bonus fitness corresponds to one successful trial. The shape of the fitness curve of run 2 is characteristic for most of the other runs: there is a fast increase in the beginning followed by a plateau and one or two more phases of faster growth when new trials are finished successfully. Then, there is a drop when proceeding to the *main-prim* stage. The fitness reaches almost zero because of the new kind of penalty *Object Moving* that is given in some of the trials of the *main-prim* stage. This penalty can be so big that is cancels all the received fitness.

Figure 3.2 depicts the best final individual of run 2 at the moment it completes the primitives in the 16 trials. A video of the robot executing the different primitives can be found on the website (Video 1, see Appendix A.2).

Additionally to these 10 incremental runs, where the primitives are learned in the two different stages, the experiment was also run five times starting directly in the *main-prim* stage (non-incremental runs). Four of the five runs completed in between 250 and 1146 generations (median 531, lower/upper quartile 454/1146), the other run is very similar to the incremental run 0 above, completing all but one trial. The shape of the development of the fitness of the top individual in every generation is very similar to the ones described above. A comparison of the number of generations needed to converge between the incre-

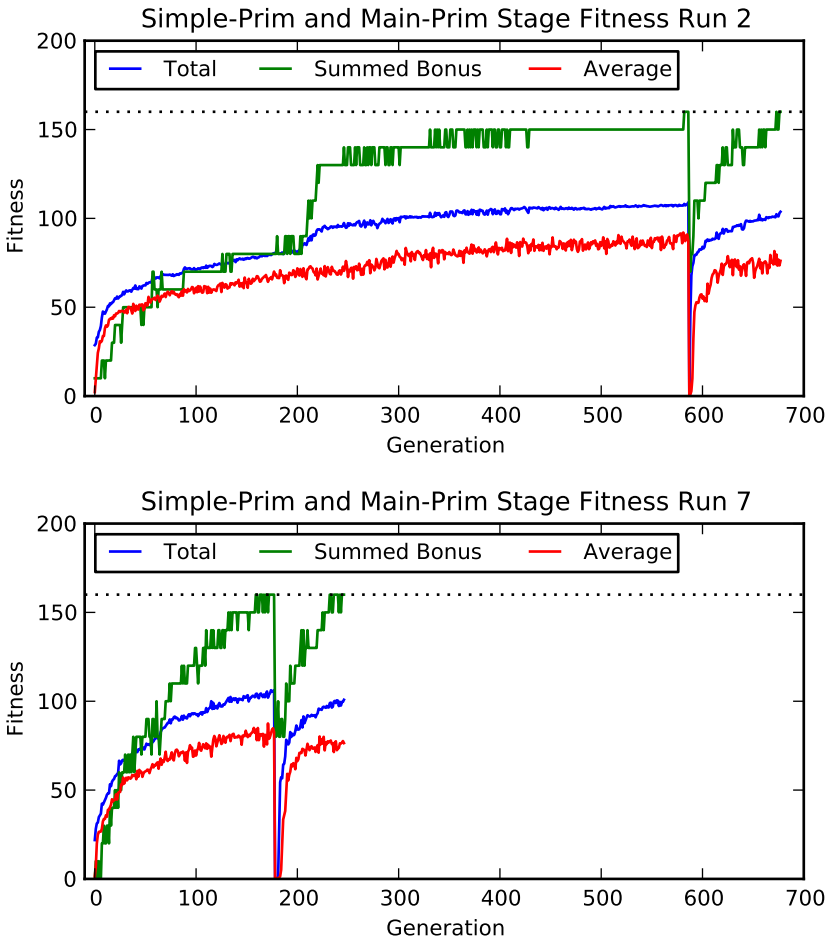


Figure 3.1: Main, bonus and average fitness during the *simple-prim* and *main-prim* stage of run 2 and 7 (top and bottom figures). Every 10 bonus points represent one successful trial. The dotted line shows the maximum possible bonus.

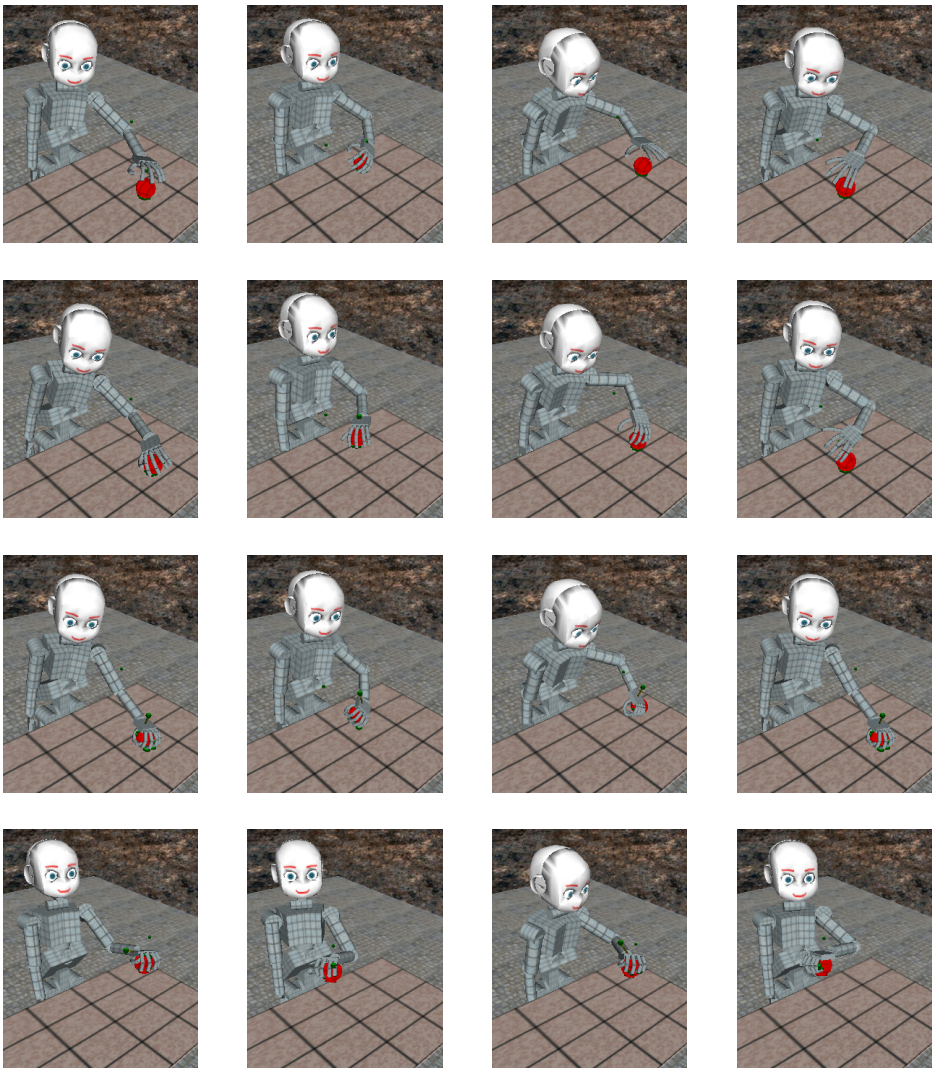


Figure 3.2: Positions when completing the primitives, every row corresponds to a primitive (reach, open, grasp, move).

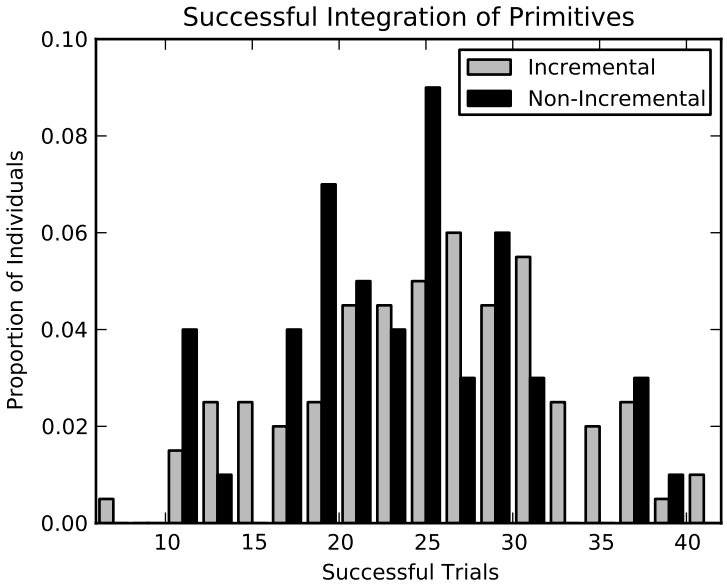


Figure 3.3: Comparison between the incremental (*simple-prim* and *main-prim* stage) and non-incremental (only *main-prim* stage) approach regarding the success of the primitive integration.

mental and non-incremental approach with a two-tailed Mann-Whitney U Test yielded a non-significant difference ($p = 0.90$).

The non-incremental runs show that the elementary behaviours can be learned in a single stage and therefore that the *simple-prim* stage is not necessary. However, to run the algorithm for 100 generations (with four parallel threads) takes about 2.5 hours in the *simple-prim* stage and 18.5 hours in the *main-prim* stage. This is due to the difference in simulation time between the kinematic and dynamic mode. It is also the reason why only five non-incremental experiment runs could be afforded.

3.1.1 Integration Test

The integration test described in Section 2.10.1 were run for the individuals from the incremental and from the non-incremental training process. Figure 3.3 shows how many individuals succeeded in how many trials. As we can see, the integration is possible in all off the examined individuals, although for most it is not reliably reproducible or not successful for all starting positions. The individuals that did better at the primitives do not have a clear advantage for the integrated behaviour (not shown in figure). Indeed, individuals that can not execute all the primitives perfectly, outperformed others that can in many cases.

The median of the amount of successful trials of all the individuals from the incremental approach is 25, the one of the non-incremental approach 23.5. This difference however is not statistically significant ($p = 0.10$, two-tailed Mann-Whitney U Test).

Figure 3.4: Comparison of the percentage of successful trials in the trained positions between the individuals coming from different conditions (the whiskers expand to the minimum and maximum with outliers marked as +).

An analysis of the caretaker instructions shows that the open primitive was jumped more often than not. The median over all the individuals of the number of trials in which it was jumped is 30 of the possible 40 trials (lower quartile at 20.25 and upper at 35).

Note that the way the caretaker decides to switch from one primitive to another is only one example of sensible guidelines. The results of this integration test might look quite different if other rules were chosen.

3.2 The *Integrated* Stage

In all of the conditions the algorithm produces individuals that learned the integrated behaviour at least in some cases. The results of the robustness test described in Section 2.10.2 are shown in Figure 3.4 (the data is also summarised in the Appendix in Table A.1. The figure depicts the percentage of successful trials of the tested individuals taken together by condition. The statistical significance of the difference between the groups is evaluated with a two-tailed Mann-Whitney U Test. The results of the test are shown in Table 3.1. The best performance is obtained in the *ST-CT* and *ST* conditions. There is no significant difference between the *CT* and *NI* conditions, but the best individual of the *CT* conditions is clearly better than the one of the *NI* condition.

The development of the fitness in the course of the algorithm and more detailed description of the results of the different conditions are explained in

	ST-CT	*ST*	*CT*	*NI*
ST-CT	-	24	9.5**	8.5**
ST	76	-	20*	16.5*
CT	90.5**	80*	-	51
NI	91.5**	83.5*	49	-

Table 3.1: Two-tailed Mann-Whitney U Test values of the results of the robustness test. The smaller the value the more likely the condition of that row is better than the one of that column. * indicates the value is significant for $\alpha = 0.05$ (critical U-value: 23) and ** for $\alpha = 0.01$ (critical U-value: 16).

the next sections.

3.2.1 Condition *ST-CT*

Nine of the runs yielded individuals that perform the integrated behaviour autonomously in all the four types of positions. Figure 3.5 show the fitness of the best agent in every generation for three of the experiment runs. The solid lines are the total fitness values, the dotted the total bonus and the dashed the total bonus received in self-talk trials. The curve of run three is representative of five runs that have a rapid increase in the beginning, corresponding to the learning of a self-talking strategy that results in a successful integrated behaviour, followed by a long slightly jagged plateau. Another four runs have fitness curve similar to the one of run 1 where the self-talk strategy is only learned sometime in the middle of the process. Run 8 is the only run where the fitness stays more or less flat the whole time and the integrated behaviour is not learned.

3.2.2 Condition *ST*

The best individual at the end of eight of the runs learned the integrated behaviour in almost all of the trials. For seven of these, the maximum bonus fitness was achieved. A representative fitness curve of this is the one of run 0 in Figure 3.6. The self-talking strategy is learned in the first half of the evolutionary process. In some runs, the strategy works in all trials within just three generations, in others it takes up to forty generations for the strategy to mature. The bonus fitness of run 6 indicates that no strategy was found that works in all of the trials. In the remaining two runs the fitness stays completely constant at a very low value and no integrated behaviour is learned. This is illustrated with the fitness curve of run 1.

3.2.3 Condition *CT*

Four of the runs of this condition resulted in individuals that are able to perform the integrated behaviour autonomously at least in some trials. Run 9, whose fitness progression is show in Figure 3.7, is one of the two runs that resulted in an individual that executes the integrated behaviour reliably in all trials. Two other runs, illustrated with run 8 in the figure, learned the integrated behaviour in most of the trials, but the behaviour is not very reliable. This can be seen by the strongly fluctuating fitness. In the other six runs the integrated behaviour

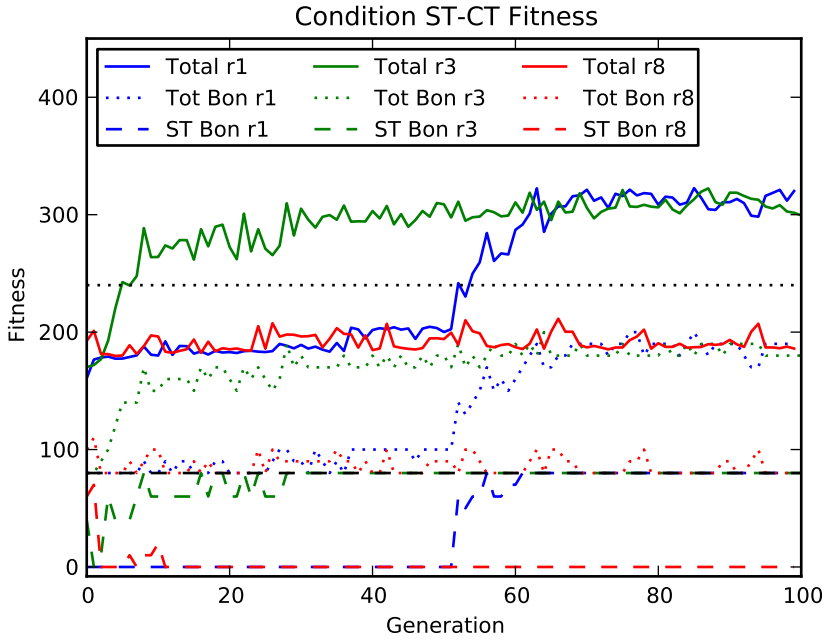


Figure 3.5: Fitness of the best individual in every generation of the *integrated* stage, *ST-CT* condition, runs 1, 3 and 8.

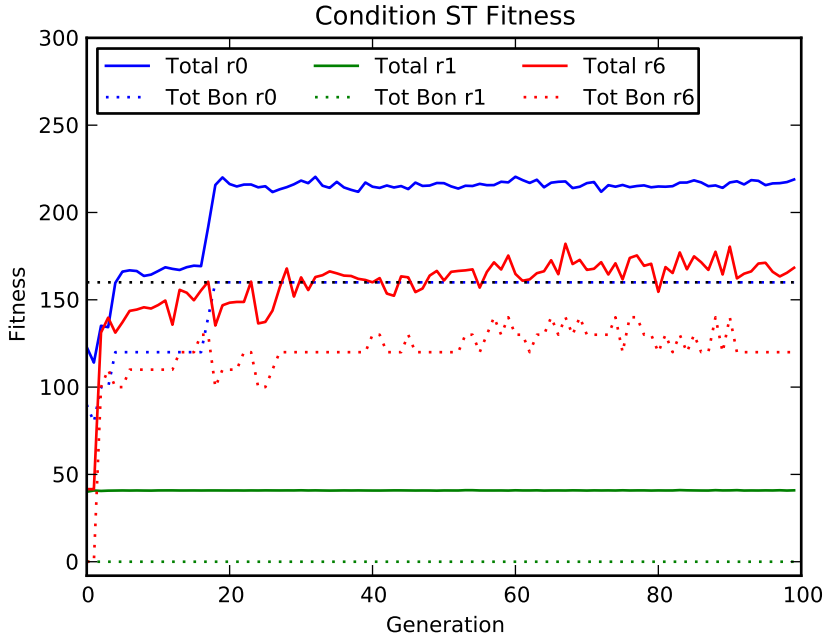


Figure 3.6: Fitness of the best individual in every generation of the *integrated* stage, *ST* condition, runs 0, 1 and 6.

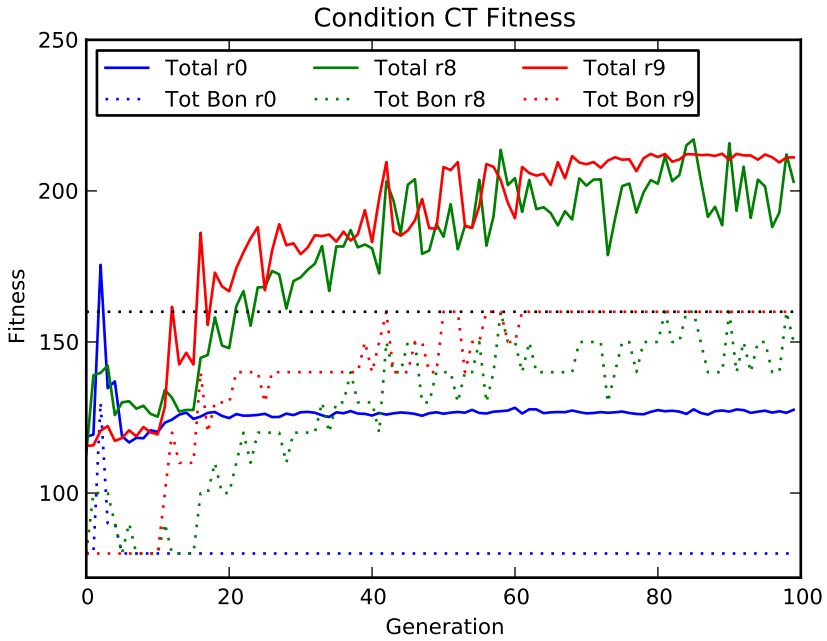


Figure 3.7: Fitness of the best individual in every generation of the *integrated* stage, *CT* condition, runs 0, 8 and 9.

is not learned at all and the fitness stays flat as in run 0. In some of these runs there are some spikes where the integrated behaviour was successful. But such individuals were only able to do the behaviour in very specific circumstances and were lost in later generations when they failed to perform.

Note how the best individual in every generation always had at least a bonus of 80. This is due to the successful integrated behaviour in the trials where the caretaker is giving instructions.

3.2.4 Condition *NI*

Five runs produced individuals that are able to do the behaviour in some of the trials. There are however no individuals that are able to reliably perform the integrated behaviour in all trials. The most promising run is run 8 shown in Figure 3.8. The fitness curve is jagged and continuously increasing over the 100 generations. In some generations the behaviour was successful in all trials. Three additional runs have a similar fitness curve but the highest bonus achieved is between 110 and 150. The only other run in which the behaviour is learned at least partially is run 2. In the other five runs nothing is learned except for a few lucky successes as in the *CT* condition. Run 0 is the run with the most such spikes.

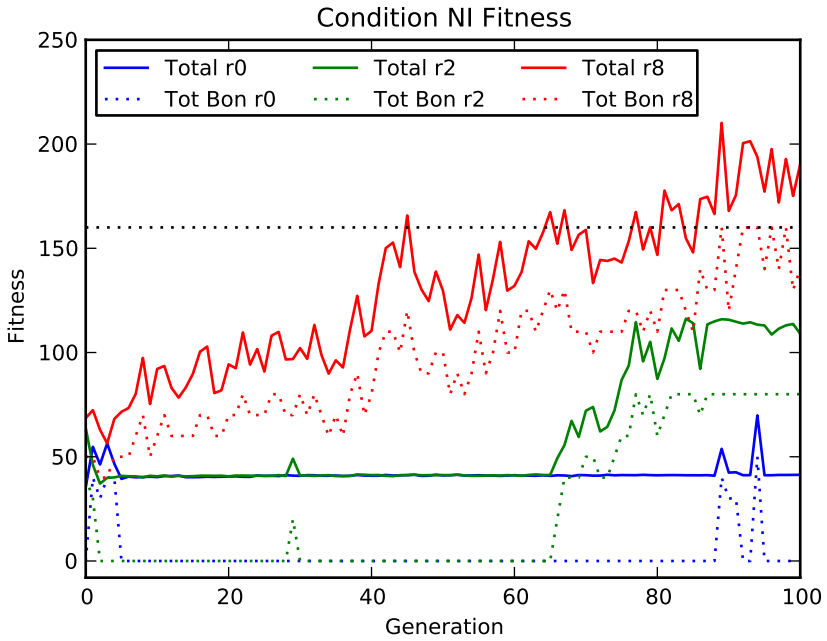


Figure 3.8: Fitness of the best individual in every generation of the *integrated* stage, *NI* condition, runs 0, 2 and 8.

3.2.5 Generalisation

Figure 3.9 shows a box plot comparing results of the different conditions of the generalisation test described in Section 2.9.3 (see Table A.2 for exact values). The data for *ST-CT0* and *ST0* comes from taking the same individuals as for the *ST-CT* and *ST* conditions respectively but instead of allowing the robot to self-talk, the instructions are given by the caretaker.

Table 3.2 shows the results of comparing the results of the conditions with a Mann-Whitney U Test. The *ST-CT* condition is significantly better than the *ST*, *CT* and *NI* condition and the *ST* condition than *NI*.

Figure 3.10 compares the results of the generalisation post evaluation of the best individual of every of the four conditions. For the *ST-CT* and *ST* condition, the results of the best individual when using the caretaker instructions (*ST-CT0*, *ST0*) are also shown.

The performance of the individuals from the *ST-CT* and *ST* conditions when they are using self talk instructions and when the caretaker provides the instructions are compared in Figure 3.11. A two-tailed Mann-Whitney U Test was used to compare the performance of the 10 repetitions of the 196 trials. It shows that for six of the individuals of the *ST-CT* condition and 3 of the *ST* condition, the performance is significantly better when they are following their own instructions. Furthermore, the time that it takes from the start of the trial until the object reaches the target is compared in the individuals that are significantly better when self talking. The durations in Table 3.3 show that for all of the individuals of the *ST-CT* condition and for two of the *ST*, this time

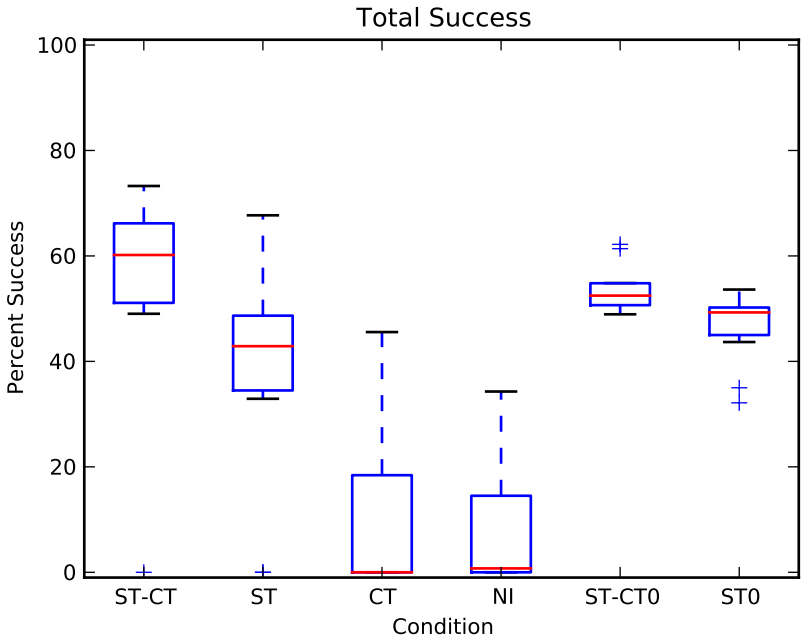


Figure 3.9: Comparison of the percentage of successful trials in the generalisation post evaluation test between the individuals coming from different conditions.

	ST-CT	*ST*	*CT*	*NI*	*ST-CT0*	*ST0*
ST-CT	-	22*	11**	7.5**	35	-
ST	78*	-	32	16**	-	63.5
CT	89**	68	-	45	-	-
NI	92.5**	84**	55	-	-	-
ST-CT0	65	-	-	-	-	-
ST0	-	36.5	-	-	-	-

Table 3.2: Two-tailed Mann-Whitney U Test values of the results of the post evaluation. The smaller the value the more likely the condition of that row is better than the one of that column. * indicates the value is significant for $\alpha = 0.05$ (critical U-value: 23) and ** for $\alpha = 0.01$ (critical U-value: 16).

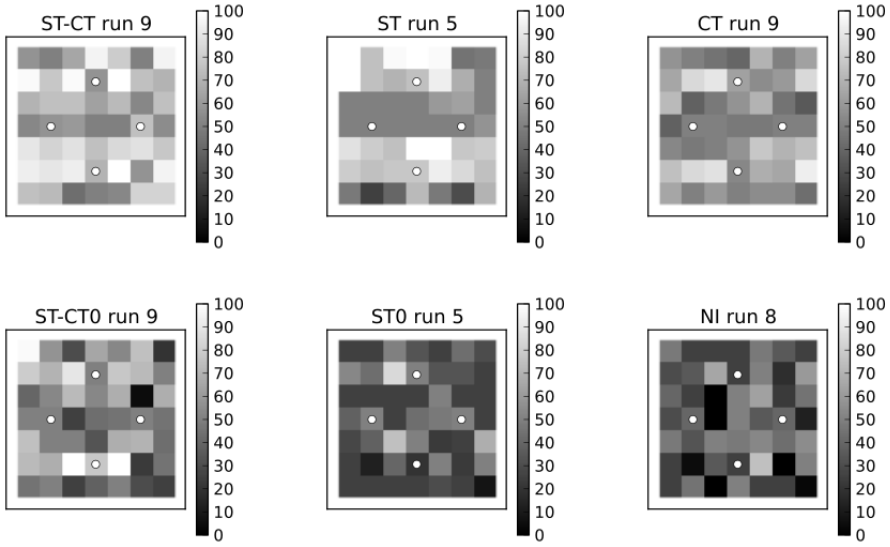


Figure 3.10: Percentage of successfully executed integrated behaviours when the object is located in each of the 49 cells of the generalisation test. The white dots represent the positions of the object during training.

Condition	*ST-CT*						*ST*		
Run	1	2	4	6	7	9	3	5	9
Self Talk	**8.77**	**7.40**	**7.06**	**8.51**	**8.87**	**7.87**	**8.18**	7.95	**8.12**
Caretaker	8.99	8.75	9.00	8.97	9.05	8.94	8.80	**7.89**	9.12

Table 3.3: Average time in seconds needed to move the object to the target location when using self talk or caretaker instructions. The average is taken over all the successful trials.

is lower.

3.2.6 Strategies

In this section the strategies of some individuals to complete the integrated behaviour are examined. Figure 3.12 shows the self talk and caretaker instructions in the four trial types of the *integrated* stage. The individuals were selected to show the important differences in the strategies. For the individual from the *ST-CT* condition run 7 and *ST* run 5 the self talk instructions are very similar to the caretaker instructions. There is some variation in the length of the primitives, but the sequence of primitives is the same. This is especially interesting for the *ST* condition as these individuals have not been given the caretaker instruction during training.

In the self talk instructions of the individual from the *ST-CT* condition run 7, there is rapid fluctuation between two instructions before the new instruction is established continuously. This behaviour can be seen in a few of the individuals from both conditions.

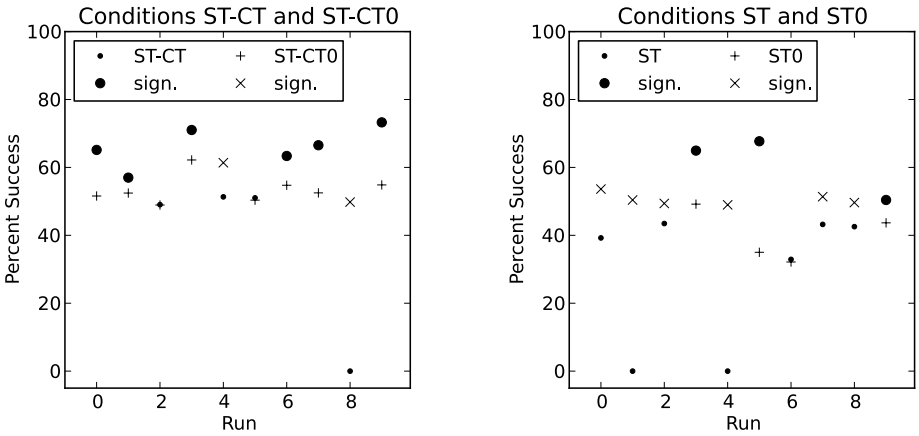


Figure 3.11: Comparison between self talk (*ST-CT*, *ST*) and caretaker (*ST-CT0*, *ST0*) instructions of the final best individual of every run. Big dots and plus markers indicate that the difference is statistically significant ($p > 0.01$).

Another interesting strategy is only seen in the *CT* condition. It is shown with the best individual from run 2: the *reach* instruction is shortened to almost nothing and the *grasp* starts much earlier. Just as all the other individuals, it never uses the *open* instruction, although the caretaker uses it in some cases. Both of these changes allow the individual to finish the behaviour faster, but reduce its ability to generalise with respect to the object position.

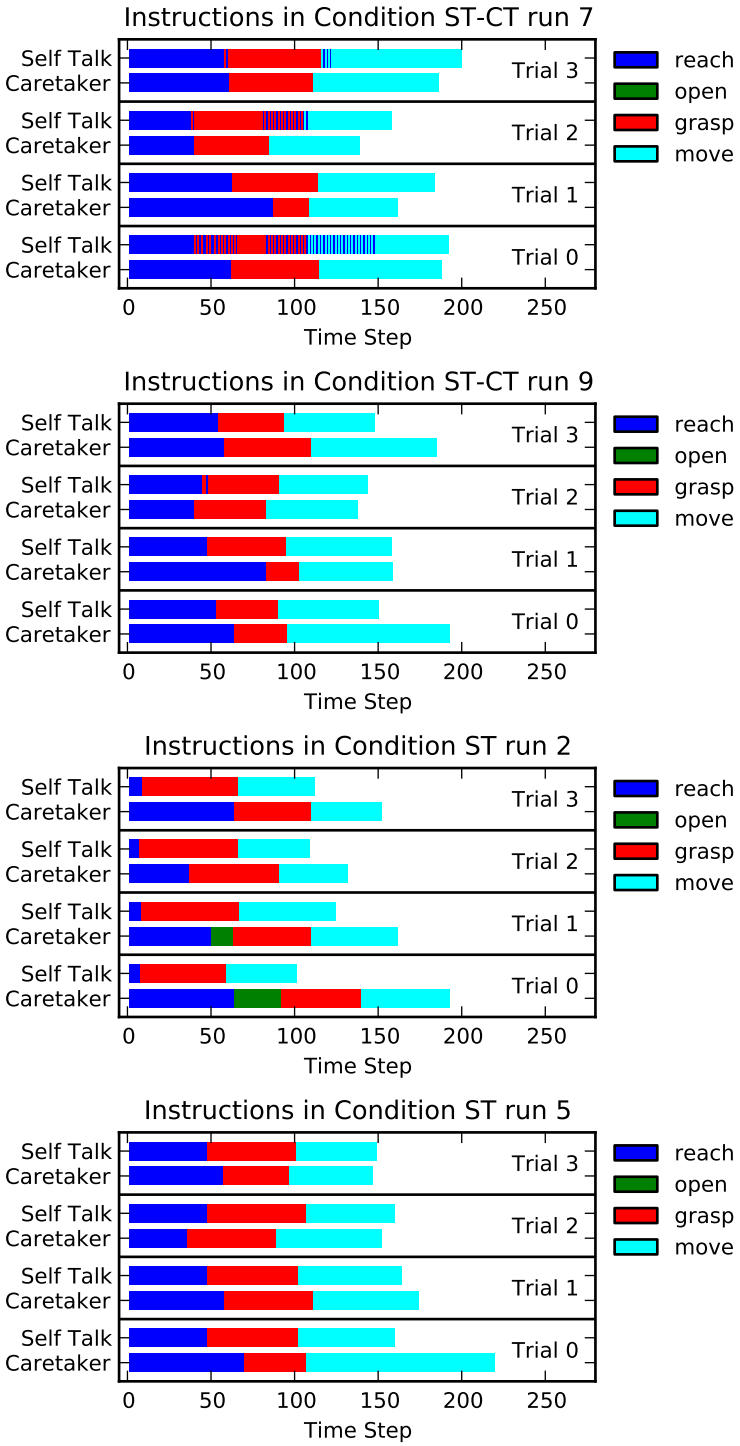


Figure 3.12: Self talk and caretaker instructions in the best individual of the *ST-CT* condition run 7 and 9 and *ST* run 2 and 5 during the four types of trials used in the *integrated* stage.

Chapter 4

Discussion

In this chapter, the motor primitives that were learned in the experiments are discussed first. Then, the different ways in which the integrated behaviour was learned is analysed. The last section concludes the discussion and presents possible future work that could expand and improve these experiments.

4.1 Motor Primitives

The results of the experiments show that the availability of linguistic instructions corresponding to goal-directed behaviour promotes the emergence of basic behavioural units, the motor primitives, that can be combined to produce higher-level behaviour. The emerging motor primitives possess several of the properties of organic compositionality presented in the introduction. The first is the already mentioned possibility to combine them to new behaviour. The integration tests after the *main-prim* stage demonstrate how well the individuals handle the transitions between the primitives without the need to learn them. Visual inspection of the integrated behaviour shows that the change from one primitive to another is often quite smooth (see Video 2, Appendix A.2). We attribute this capability to the fact that all the primitives are controlled by the same neural network and that the primitives were learned in similar circumstances as they are later needed.

The proposed mechanism to internalise the integrated behaviour through self talk is able to exploit the fact that the primitives fit together well and improve the smoothness of the movement. This is achieved through changes in the timing of the instructions. This can be seen by the fact that the total time needed to complete the behaviour is usually lower when using self talk and by visually inspecting the behaviour (see Video 3, Appendix A.2).

Other properties of organic compositionality are shown by the generalisation post evaluation: the primitives are to a good extent able to deal with variability in the location of the object and the initial posture of the robot. More detailed analysis of a similar control system used in a similar task in [24] suggests that the primitives would also be able to deal with different object sizes and shapes.

4.1.1 Incremental Primitive Learning

Whether the primitives are learned in one stage or incrementally by first learning simplified primitives and only then the actual primitives, does not seem to produce significantly different individuals. There is also no apparent difference in the number of generations needed for the acquisition of the skills nor in the chance that the algorithm ends in a local minimum. The only obvious advantage of one approach over the other is the reduced running time of the incremental algorithm.

As mentioned in Section 2.7, the incremental learning process was introduced because of the unsatisfactory results with the direct approach. Several other changes to the experiment setup were applied after the switch to incremental learning. The exact success conditions of the primitives were for example modified and the number of individuals in a genome was doubled. These modifications must have enabled the non-incremental approach to work as well. This was not noticed however until the final experiment was performed. There was little incentive to switch back because of the obvious practical advantages of the reduced simulation time.

We can deduce from the comparison with these previous experiments that the incremental approach might be necessary when the primitives are more complex or the parameters of the algorithm are less optimised.

4.2 Integrated Behaviour

All of the experimental conditions of the *integrated* stage produce individuals that are able to autonomously produce the integrated behaviour. The conditions in which there is no self talk (*CT* and *NI)* show that robots that have acquired the elementary behaviours are capable to learn a more complex skill that is composed of these behaviours. Massera et al. used very similar methods and an almost identical task in [7]. Their individuals did not learn the task with a network that was evolved without having learned any primitives beforehand and without linguistic support. This suggests that the learned motor primitives represent a prerequisite for the development of integrated behaviour of the level of complexity considered in this thesis. The difference in performance between the individuals with support from the caretaker (*CT*) and the ones without (*NI*) is not significant. However, if we ignore the runs in which the algorithm did not find a working solution, we can see that all of the individuals of the *CT* condition show a much better ability to generalise than any of the individuals from the *NI* condition. This means that the presence of the caretaker improves the resulting autonomous behaviour.

But we have found a better method to integrate the motor primitives. The self talk mechanism has been shown to significantly improve the robustness of the resulting behaviour and its ability to deal with new situations. Not only the quality of the behaviour is better, but also the chance of finding successful individuals is higher with the algorithm operating on self talking individuals. From a theoretical point of view, this result confirms the theory elaborated by Vygotsky. From a engineering point of view, it suggests that language mediated learning can enable the acquisition of robust and adaptable behavioural capabilities.

From visual comparisons between individuals that self talk and others that do not, we get an intuitive reason why the former have a behaviour that generalises better: individuals without self talk start closing their hand while they are reaching for the object. By the time the object is reached, it just fits in between the fingers. The activation of the tactile sensors then seems to trigger the switch of the focus to the target location and the further closing of the hand. The fact that the hand is closing from the beginning of the movement makes the behaviour more dependant on the initial position of the arm and the object. For example, if the object is further away than usual, the closing fingers might touch the object during the approach and make it roll away (see Video 4, Appendix A.2). In contrast to that, most self talking individuals first have a reaching phase, where the hand is kept fairly open until it is close to the object. Then, either triggered by the proximity to the object or the passing of time, they switch to the grasp primitive (see Video 5, Appendix A.2). We infer from this, that the self talking mechanism improves the individual's ability to choose the appropriate behaviour in situations where multiple actions are possible in similar sensory circumstances.

The self talking condition (condition *ST)* shows that the algorithm can find a working combination of primitives without any external help. This is remarkable as it does not only show that the self talking mechanism serves to internalise a sequence of instructions, but also, that the trial and error learning process can find such a sequence on its own. Of course the number of primitives that are available is very small and it is not obvious how this would work out for a larger set of primitives or for more complex integrated behaviours. In fact we have run preliminary experiments (with only a few runs per condition) where the integrated behaviour had to be executed starting with a closed hand. This made it necessary to use the open primitive before the object could be grasped, and therefore to use all four primitives. This was only learned in the condition where the caretaker helped with the acquisition of the self talking strategy (condition *ST-CT*).

Our experiments further show that the presence of the caretaker instructions during training improves the resulting behaviour in terms of stability and how well it generalises. In the *ST-CT* condition, the robot learns a mapping of the internal representation of the sensory input to the linguistic output corresponding to the next instruction given by the caretaker. This mapping is learned in parallel to letting the robot use its linguistic output to guide its actions, and verifying whether the desired behaviour is achieved. This is important as it does not force the robot to learn the exact caretaker instructions, but instead focuses on the extraction of the relevant aspects that allow the integrated behaviour to be completed. As is illustrated in Figure 3.5, none of the individuals ever learn to perfectly predict all the changes in the caretaker instructions, but are nevertheless able to perform the behaviour correctly. Furthermore, the parallel learning also makes it possible for the individuals to not only extract the essence of the instructions, but to optimise them to achieve the goal in a shorter time.

4.3 Future Work and Conclusions

For future work we propose to verify the results on the physical iCub robot. A lot of the necessary work has already been done as the EvoICub software was

built to control the physical robot in the same way as the simulated one. A vision module that finds the two-dimensional coordinate of a coloured object already exists. However, it would have to be expanded to get the third dimension by either assuming the object stays on the table or by using the two eyes of the iCub. Additionally, the multi-dimensional and continuous tactile sensor signal from the robot would have to be converted to a suitable binary signal.

Further, the quality and the amount of motor primitives should be increased in future work. For example, the grasp should be able to handle objects that need to be grasped from different directions and hold on to the objects more securely. It might be a good idea to separate the control of the focus into a separate network that is expanded to control the movement of the head and eyes to look at the current point of interest. Separate instructions could then be given to this focus network and the reach/grasp network which would make it possible to use compositional instructions such as "reach ball" and "reach cylinder". In this example, the focus network would look at the correct object and the reach/grasp network would reach that object.

Another interesting expansion of these experiments would be to let the self talking mechanism learn to correct errors. If the robot has grasped the object for example and starts moving it, but then the object is dropped, the self talk should again switch to the reach instruction. Such a behaviour has indeed already been observed in one of the preliminary experiments. It might be possible to achieve this correcting behaviour by simply exposing the robot to training situation in which the object is dropped, and rewarding it for starting to reach it again.

In this thesis we showed that adaptable and combinable motor primitives for a humanoid robot can be developed through a trial and error process that finds suitable values for the free parameters of the artificial neural network controlling the robot. The fact that these lower-level motor primitives are associated to a linguistic instruction that has been provided throughout the training process by a caretaker, allows the robot to guide its higher-level actions by learning to give itself these instructions. This self talking mechanism enables the robot to complete tasks that need the combination of the learned primitives either by loosely learning the sequence of instructions proposed by the caretaker or by finding its own strategy.

The model proposed in this thesis represents an interesting and effective way to combine goal-directed trial and error learning with supervised learning (a learning process that requires and exploits information that provides indication on how a given action should be produced). These two learning methods have complementary advantages and drawbacks that make them more or less suitable depending on the circumstances. On the one hand, trial and error learning is adequate in situations in which it is difficult from the point of view of the designer to specify how an action should be produced. This is indeed the case for the motor primitives of our experiments in which the robot has to determine the desired angular position of each joint over time. On the other hand, supervised learning techniques are suitable in situations in which the designer can specify the way in which an action has to be produced. In the context of our experiment, this is the case for the acquisition of the integrated behaviour in robots that already posses the relevant elementary skills. The good results obtained in the *ST-CT* condition indeed demonstrate that the combination of a trial and error learning for the training of the primitives, and of a language

mediated supervised learning for the integrated action, represent an effective learning method. This condition also shows that the combination of the two types of learning is benefitial at the level of the integrated action. The robot can exploit the supervision provided by the caretaker, but is also able to improve the caretaker's strategy thanks to a trial and error refinement of the timing of the used primitives.

Appendix A

Supplementary Results

A.1 Post Evaluation Test Results

	0	1	2	3	4	5	6	7	8	9
ST-CT	95	95	87.5	100	67.5	90	92.5	97.5	0	97.5
ST	67.5	0	95	90	0	90	75	80	87.5	82.5
CT	0	0	0	62.5	62.5	0	0	0	50	90
NI	0	0	32.5	0	0	32.5	62.5	60	75	0

Table A.1: Percent of successful trials of all individuals in the robustness test.

	0	1	2	3	4	5	6	7	8	9
ST-CT	65.2	57.0	49.0	71.0	51.3	51	63.4	66.5	0.0	73.3
ST	39.2	0.0	43.5	64.9	0.0	67.7	32.9	43.2	42.6	50.4
CT	0.0	0.0	0.0	40.6	44.6	0.0	0.0	0.0	45.6	59.3
NI	0.0	0.0	3.7	0.0	0.0	0.0	16.6	18.1	34.3	0.0
ST-CT0	51.6	52.4	48.9	62.2	61.4	50.4	54.7	52.5	49.8	54.8
ST0	53.6	50.4	49.4	49.2	49.0	35.0	32.1	51.4	49.6	43.7

Table A.2: Percent of successful trials of all individuals in the generalisation test.

A.2 Electronic Supplementary Material

Videos of various behaviours of the developed individuals can be found on the website `http://laral.istc.cnr.it/esm/selftalk-integration/`.

Bibliography

[1] S. Schaal, "The new robotics — towards human-centered machines," *HFSP Journal*, vol. 1, no. 2, pp. 115–126, 2007.

[2] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey," *Connection Science*, vol. 15, no. 4, pp. 151–190, 2003.

[3] R. Pfeifer, M. Lungarella, and F. Idia, "Self-organization, embodiment, and biologically inspired robotics," *Science*, vol. 318, no. 5853, pp. 1088–1093, 2007.

[4] V. Tikhanoff, J. Fontanari, A. Cangelosi, and L. Perlovsky, "Language and cognition integration through modeling field theory: Category formation for symbol grounding," in *Artificial Neural Networks — ICANN 2006*, vol. 4131 of *Lecture Notes in Computer Science*, pp. 376–385, 2006.

[5] M. Mirolli and D. Parisi, "Towards a vygotskyan cognitive robotics: The role of language as a cognitive tool," *New Ideas in Psychology*, vol. 29, no. 3, pp. 298–311, 2011.

[6] Y. Sugita and J. Tani, "Learning semantic combinatoriality from the interaction between linguistic and behavioral processes," *Adaptive Behavior*, vol. 13, no. 1, pp. 33–52, 2005.

[7] G. Massera, E. Tuci, T. Ferrauto, and S. Nolfi, "The facilatory role of linguistic instructions on developing manipulation skills," *IEEE Computational Intelligence Magazine*, vol. 5, no. 3, pp. 33–42, 2010.

[8] M. Mirolli and D. Parisi, "Talking to oneself as a selective pressure for the emergence of language," in *The Evolution of Language*, pp. 214–221, World Scientific Publishing, 2006.

[9] 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 Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 167–195, 2010.

[10] M. Schlesinger, "Evolving agents as a metaphor for the developing child," *Developmental Science*, vol. 7, no. 2, pp. 158–164, 2004.

[11] S. Schaal, "Arm and hand movement control," in *The handbook of brain theory and neural networks*, pp. 110–113, MIT Press, second ed., 2003.

[12] S. Flash and H. B., "Motor primitives in vertebrates and invertebrates," *Current Opinion in Neurobiology*, vol. 15, no. 6, pp. 1–7, 2005.

[13] M. A. Arbib, "Schema theory," in *The handbook of brain theory and neural networks*, pp. 993–998, MIT Press, second ed., 2003.

[14] K. A. Thoroughman and S. R., "Learning of action through adaptive combination of motor primitives," *Nature*, vol. 407, no. 6805, pp. 742–747, 2000.

[15] J. Konczak, "On the notion of motor primitives in humans and robots," vol. 123, pp. 47–53, Lund University Cognitive Studies, 2005.

[16] A. Billard, S. Calinon, R. Dillman, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, pp. 1371–1394, Springer Berlin / Heidelberg, 2008.

[17] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research*, vol. 15 of *Springer Tracts in Advanced Robotics*, pp. 561–572, Springer Berlin / Heidelberg, 2005.

[18] J. Tani, R. Nishimoto, and R. W. Paine, "Achieving organic compositionality through self-organization: Reviews on brain-inspired robotics experiments," *Neural Networks*, vol. 21, no. 4, pp. 584–603, 2008.

[19] J. Tani, R. Nishimoto, J. Namikawa, and M. Ito, "Codevelopmental learning between human and humanoid robot using a dynamic neural-network model," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 1, pp. 43–59, 2008.

[20] T. Inamura, I. Thosima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *The International Journal of Robotics Research*, vol. 23, no. 4–5, pp. 363–377, 2004.

[21] Y. Yamashita and J. Tani, "Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment," *PLoS Computational Biology*, vol. 4, no. 11, pp. 1–18, 2008.

[22] A. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi, "Cognitive developmental robotics as a new paradigm for the design of humanoid robots," *Robotics and Autonomous Systems*, vol. 37, no. 2–3, pp. 185–193, 2001.

[23] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The icub humanoid robot: an open platform for research in embodied cognition," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pp. 50–56, ACM, 2008.

[24] G. Massera, A. Cangelosi, and S. Nolfi, "Evolution of prehension ability in an anthropomorphic neurorobotic arm," *Frontiers in Neurorobotics*, vol. 1, no. 4, pp. 1–7, 2007.