Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping

Lorenzo Natale

LIRA-Lab, DIST, University Of Genoa

This work has been carried out by Lorenzo Natale, during his Ph.D. course in Robotics under the supervision of Prof. Giulio Sandini at LIRA-Lab, Department of Telecommunication, Computer and System Sciences, University of Genoa, Italy. ©2004 LIRA-Lab

The research described in this book has been supported by grants from the Italian Ministry of Education, University and Research (MIUR), the European Union (projects, MIRROR, COGVIS, ADAPT) and by the Italian Space Agency (ASI).

All rights reserved. No part of this book ay be reproduced, in any form or by any means, without the permission in writing from the authors.

Printed in Italy.

Copyright notice:
LIRA-Lab, DIST, University of Genoa, Italy, © 2004 LIRA-Lab.
URL: http://www.liralab.it
URL: http://nat.liralab.it

Abstract

In this thesis we propose a developmental approach to the design of a humanoid robot. We present a possible sequence of developmental stages which starting from limited knowledge enables the robot to autonomously learn to perform goal directed actions on objects (reaching, pushing, and a simple form of grasping). The robot initial knowledge consists in a few visual algorithms (disparity, tracking, motion detection) and motor synergies providing a rudimentary form of sensorimotor coordination useful to begin interaction with the environment. During the initial steps of development the robot learns to recognize and control its own body (gazing, localization of the hand); based on these abilities it moves afterward to the exploration of the external world (reaching and grasping).

We stress the importance of the physical interaction between the robot's body and the environment and the advantage of exploiting actions to simplify and learn perceptual as well as motor tasks (e.g. distinguish the hand from the background, recognize objects based on tactile experience, pushing/pulling objects on a table).

This approach is inspired by the observation of how mature behaviors emerge in infants during development and by recent theories in neural sciences proposing that the link between action and perception might be at the basis of higher level, abstract functions like action recognition, imitation and language. These considerations and the experimental results reported in the thesis support the conviction that our approach is indeed worth pursuing as it is perhaps the only route toward the realization of cognitive abilities in an artificial system.

Ringraziamenti

Vorrei esprimere la mia gratitudine al Prof. Giulio Sandini perché questi tre anni al LIRA-Lab sono stati un'esperienza davvero interessante e importante. Il suo entusiasmo per la ricerca è stato di stimolo e insegnamento durante tutto il mio lavoro. Sono infinitamente obbligato nei confronti di Giorgio Metta che è stato una guida in questi anni; le nostre lunghe discussioni e i suoi consigli – di persona e via messenger – hanno contribuito in maniera sostanziale a questo lavoro.

Un grazie a tutti i colleghi del LIRA-Lab per aver creato un ambiente di lavoro divertente e stimolante. In particolare, Matteo Brunettini ha tenuto in piedi il laboratorio durante gli ultimi mesi, nei quali mi sono reso praticamente irreperibile per la scrittura della tesi. Fabio "Le so tutte" Berton perché le sa davvero tutte e Carlos Beltran che, con occhio particolarmente severo e attento, ha contributo alla correzione finale della tesi. Eventuali errori nel testo sono dovuti alla mia esplicita volontà di non seguire tutti i suoi suggerimenti. Me ne prendo la responsabilità ☺.

Un ringraziamento "agli MRG" per avermi tenuto compagnia nei non pochi week-end e nelle serate di lavoro: Claudio, Marco, Jacopo, Luca, Massimo e Armando. In particolare Luca e Massimo per le lunghe conversazioni, alcune cene ma soprattutto… lo sporco mercato nero dei numeri ☺. Tac per le lunghe discussioni sui temi di ricerca, vita, futuro (e non-futuro ☺). Marco perché dalla California mi ha tenuto aggiornato sulle feste che mi stavo perdendo mentre scrivevo la tesi...

Non so come avrei fatto senza le varie feste e birrate organizzate dal Bio-lab (non che non facciano anche cose serie…). Vorrei esprimere la mia gratitudine a Marco che ha sempre avuto una parola di incoraggiamento quando le cose non funzionavano. Grazie a Federica, Barbara, Silvia, Elisa e Ivan per le divertenti (un po' lunghe ?) pause caffè. Ho perso il conto dei caffè che mi sono stati preparati da Federica (vorrei solo aver avuto più monete per poterli pagare tutti ☺).

Un grazie ai miei amici storici: Matteo, Carlo, Gabriele, Marco, Lorenzo, Fancesca e Corrado, che di tanto in tanto mi hanno ricordato che c'è vita oltre le mura del laboratorio. Corrado come compagno di squash ha reso possibile l'unica attività fisica degli ultimi mesi. Colgo l'occasione per ringraziare finalmente anche Massi "Dinobirdo", per alcune lezioni gratuite su networking, TCP/IP e Multicast (chi può dirlo, magari un giorno entrerò nel Campo ☺).

Infine, un pensiero particolare a tutti i membri della mia famiglia per il supporto e l'affetto che mi hanno dato in tutti questi anni. Questa tesi è dedicata a loro.

Acknowledgments

I would like express my gratitude to Prof. Giulio Sandini because these three years at LIRA-Lab have been a really interesting and important experience. His enthusiasm for research has been an incentive and example for my work. I'm profoundly indebted to Giorgio Metta who has been a guide along these years; our long discussions and his advices – in person and on messenger – have given a substantial contribution to this work.

I would like to thank all my colleagues at LIRA-Lab for creating an interesting and stimulating environment to work in. In particular Matteo Brunettini took care of the lab in the last few months when I made myself almost unreachable while writing the thesis. Fabio "I know it all" Berton, because he does know it all and Carlos Beltran who, with particular attention and rigorousness, contributed to the review of this thesis. Those mistakes you might find in the text can only be due to my explicit will to not follow all his advices. I take full responsibility for it ☺.

Thanks to the guys at MRG-Lab for keeping me company during quite a few working week-ends and nights: Claudio, Marco, Jacopo, Luca, Massimo and Armando. In particular Luca and Massimo for the long conversations, some dinners but most of all… the black dirty market of numbers ☺. Tac for the long discussions about research, life and future (or no-future ☺). Marco because, from California, kept me up to date on the parties I was missing while writing the thesis…

I really don't know how I could have done without the parties and beer happenings organized by the Bio-Lab (not that they don't do serious things…). My gratitude to Marco who always had an encouraging word when things did not work out. Thanks to Federica, Barbara, Silvia, Elisa and Ivan for the entertaining (a bit too long ?) coffee-breaks. I lost count of the coffees that Federica prepared me (if only I'd had enough change to pay all of them ☺) .

I would like to thank my historical friends: Matteo, Carlo, Gabriele, Marco, Lorenzo, Fancesca and Corrado who from time to time reminded me that there is life beyond the walls of the lab. Corrado as a squash mate made possible the only physical activity in the last few months. I take here the opportunity to thank also Massi "Dinobirdo", for some free lessons about networking, TCP/IP and Multicast (who knows, maybe on day I will join the Field ☺).

Finally, a special thought to the members of my family for the support and love they have given me all along these years. This thesis is dedicated to them.

To my family

Contents

1. Introduction 15

1.1. Theories of cognition 17
1.2. Embodiment 18
1.3. Cognitive developmental robotics 20
1.4. Developement...21
1.5. Self-supervised learning (what do we need manipulation for?) 23
1.6. A developing robot 26
1.7. Outline..28

2. System's architecture...31

2.1. Babybot's body 31
2.2. Interface cards 35
2.3. Software architecture 37
  2.3.1. Communication protocols...38
  2.3.2. Hiding the Operating System..39
  2.3.3. Robot independent code 41
  2.3.4. Robot specific interface..42
2.4. Learning architecture...43

3. A biologically inspired elastic actuator 47

3.1. Biology and motor control 48
3.2. The mechanical prototype...52
3.3. Mathematical model 52
3.4. Experiments with a single actuator 54
3.5. Experiments with coupled actuators..54
3.6. Open loop control 56
  3.6.1. The force-displacement plane..56
3.7. Conclusions...63

4. Eye movements...65

4.1. Retina-like visual system 65
  4.1.1. Some maths 67
4.2. Gaze control 68
4.3. Visual stabilization...69
4.4. Voluntary eye control 69
4.5. Vergence control 70
4.6. Eye-head coordination 71

4.7. Attentional system 72

5. Learning a body map from experience 75

5.1. Learning gravity compensation 76
  5.1.1. Discussion 79
5.2. Learning to localize the robot's hand 79
  5.2.1. Segmentation and prediction 82
  5.2.2. Exploiting the hand prediction 87
  5.2.3. Discussion 88

6. Learning to act on objects 95

6.1. Reaching 95
6.2. Learning to act on objects 100
  6.2.1. Description of the experiment 100
  6.2.2. Results 103
  6.2.3. Testing the learned maps 104
  6.2.4. Discussion 105
6.3. Learning about objects' shapes 106
  6.3.1. Hand calibration 107
  6.3.2. Touch-elicited grasp 108
  6.3.3. Description of the experiments 110
  6.3.4. Experiment 1 110
  6.3.5. Experiment 2 111
  6.3.6. Discussion 112

7. Conclusions 115

7.1. Motor theories of perception 115
7.2. Objects' affordances and action 117
7.3. Linking action to perception 118

8. Appendix 121

8.1. Motion algorithm 121
8.2. Ellipse Fitting 122
8.3. Self organizing feature map (SOM) 125

List of figures 127

References 137

Index 147

Joe said, "But Edgar Mahan proved that a synthetic life form can't come into existence, 'Life has to come from life, and therefore, in the construction of self-programming mechanisms – ' "
"Well you're looking at twenty of them," Mali said.
"Why were we told they couldn't be made?" Joe asked her.
"Because there're too many unemployed people on Earth as it is. The government faked scientific evidence and documentation to say robots couldn't be done. They are rare, however. They are hard to build and costly. I'm surprised to see this many. It is all he has, I'm sure. This is a – " She searched for the word. "For our benefit. A display. To impress us".

Philip K. Dick, "Galactic Pot-Healer", 1969

Chapter 1

Introduction

The ultimate goal of Artificial Intelligence is perhaps one of the most intriguing of science, being that of building a machine which embeds some sort of intelligence. The problem is significant because it implicitly subsumes that of understanding what intelligence actually is and how to design an artificial system which can be called intelligent. The former is not a trivial question because it delineates the approach to be pursued and provides insights on the possible ways to determine achievements. For this reason, it is useful to begin the discussion by trying to give a definition of artificial intelligence.

Defining artificial intelligence is not easy; indeed there is little agreement among scientists about this point. According to Russell and Norvik (Russell and Norvig, 1995) definitions of artificial intelligence can be organized into four categories: they may be concerned with the notion of *reasoning* versus *behavior* and measure success based on *human* performance or an "ideal concept of intelligence" that the authors call *rationality*. Accordingly these four categories give AI four different goals:

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

Table 1-A. Goals of Artificial Intelligence (adapted from (Russell and Norvig, 1995)).

The existence of these different approaches is already interesting, as traditionally the focus has been on building computer programs to solve problems that could naturally be formed in term of symbols like theorem proving, logic, or chess play. This approach sees an intelligent system as a rational reasoning device and is often called Classical AI (or Symbolic, Knowledge-based AI). Classical AI uses symbols to represent knowledge so that a machine can work with them to derive some additional knowledge. This approach has been successful in particular domains, but has run into at least two fundamental problems. Firstly, symbolic systems lacked the ability to detect and use context information; even within their domain of operation they often failed to provide the correct answer in ambiguous situations where humans would easily be successful (consider for instance the problem of pattern recognition or language understanding). On the other hand these techniques seemed unable to generalize knowledge and use it across different domains. As a result it was impossible to scale up from limited, tractable domains to more complicated situations; the second fundamental problem is thus how to apply symbolic AI to real-life problems, usually encountered for instance in the field of computer vision and robotics (consider for example navigation and surveillance). In this case the most difficult problem is perhaps how to synthesize a symbolic representation of the real world (especially when the latter it is not 100% deterministic), and relate it with the information received from the sensory system. Moreover, real-life problems, when reduced to formal logic, easily become computationally intractable.

On the other extreme, Behavioral Based AI has given more emphasis on the capacity of the system (at this point something more than a computer) to interact with the environment. Somewhat related to this philosophy, the Turing test was initially put forward by Alan Turing as a means to evaluate artificial intelligence: requirement for an intelligent agent is to behave in such a way to fool a human interrogator (judge). Several points have been raised against the validity of this test (the Chinese room argument see (Russell and Norvig , 1995) for a review[1]). However, besides its historical importance, the Turing test is significant because focuses on the behavior of the system and puts human cognition as a reference for intelligence. Indeed, if on the one hand it is hard to give a satisfactory definition of intelligence, on the other nobody would argue about the fact that humans are intelligent. The Turing test does not yet introduce the concept of body; intelligence is measured by the *interaction* between a computer and a human being, but this interaction takes place via messages typed on a keyboard and, hence, is not physical. In the Behavioral Based AI approach intelligence is seen as a feature unique to biological systems and the focus is on their capacity to interact with the world (Brooks, 1990; Pfeifer, 1996). Behavioral complexity emerges from basic sensorimotor coordination and adaptation. The latter seen not only as the ability to tune predefined behaviors to maintain their functionality or to improve performance, but also – and especially – as the capacity to form new behaviors based on those already available (learning, evolution, development). Learning and adaptation are made possible by the continuous interaction between the agent's body and the environment where the agent is embedded in (situatedness, embodiment).

---

[1] The Chinese room argument was originally raised by Searle.

In setting humans as a reference point, the goal changes radically; we are no longer concerned with getting the *correct* result out of our system but rather with getting the same answer a human would give to the same problem. Embodied AI is hence an interdisciplinary field linking together computer science, robotics, brain and cognitive sciences like physiology and psychology.

1.1. Theories of cognition

Closely related to the just mentioned approach to AI, theories about cognition have been proposed. Following Vernon (Vernon, 2003) two possible groups can be identified: theories which see cognition as representational and theories which see cognition as emergent.

According to the representational approach cognition is mainly a computational process. Cognitive behavior is the result of this computation which is carried out on symbolic representations instantiated by the system through sensing and reasoning about these representations. Cognitivism is very close to Symbolic AI which tries to imitate intelligence by means of algorithms working on symbols to produce other symbols. The world is abstracted to a formal representation which is manipulated with syntactic rules; the result of the computation (reasoning) is then used to obtain a solution to a problem or to plan an action.

Connectionism and dynamical approaches are rather different theories (for a review see: (Beer, 2000)). In both these cases cognition is conceived as a property emerging from the structural organization of sub-elements. Representations are not explicitly defined in the system (as opposed to the symbols in the cognitivistic approach) but are intrinsically defined by the internal architecture as the result of its particular history (experience). Any given state of the system implicitly code a representation, symbols may be associated by an external observer to each particular state, but they are neither part of the system nor are they required for its proper functioning. An example of a connectionist system is a neural network which has learnt to associate input patterns to certain outputs. The representation the network codes is represented by the weights corresponding to the neural connections established during learning. This representation is hence said to be distributed across the whole network architecture. The dynamical approach tends to describe a cognitive system as a process whose state evolves over time, as the result of the influences of internal as well as external forces. An input by itself does not produce a definite single state, as the time course of the system depends also on its internal state. Mathematically such a system may be represented with a set of differential equations (in the continuous case) defining the evolution of the state of the system over time. Thus the state at time $t+1$ depends on both the input and the state at time t. Finally it is important to stress that the distinction between Cognitivism, connectionist and dynamical system approaches is not at the implementation level, as dynamical and connectionist models are often simulated on digital computers and recurrent neural networks have been employed to model dynamical systems. The distinction between the different approaches is more on the point of view they take in the modeling phase and on the insights they offer to understand the cognitive phenomenon they describe (Beer, 2000).

For the purpose of this discussion it is worth stressing the fact that in emergent theories the interaction between agent and environment is very important. It is by acting in the world that the system can change/shape its internal structural organization. As the implicit representation resulting by this interaction depends on the coupled effect of the body and the environment, it is also impossible to separate the representation from the body that has generated it. Moreover, a dynamical system exists only in presence of an environment which allows (*enacts*) it to evolve over time. So *enaction* (Maturana and Varela, 1998) is another approach to cognitive system which puts even more emphasis on the interaction between the body and the environment. Cognition is *"effective action"* or the process that allows an agent to take appropriate actions according to its goals and its internal as well as external state: agent and environment are coupled dynamical systems which evolve together (Chiel and Beer, 1997).

1.2. Embodiment

According to both Behavioral Based AI and the most recent theories about cognition, the existence of the body and its consequent interaction with the environment are a necessary condition for the emergence of intelligence and cognition. Imagine, however, that we want to reproduce the exact functioning of the brain from a purely computational point of view; even in this case it is not possible to avoid considering the role of the body (Chiel and Beer, 1997) . Usually the nervous system is thought of as a black box receiving input from the environment and, based on its internal state, providing motor commands for the body. However, the body carries out a great deal of processing on the input signals received by the nervous system as well as on its output. It is possible to see the body as a sort of interface for the nervous system. For example the morphology of the visual system provides an optimal spatial sampling of the light impinging the retina. Photoreceptors in the retina are much denser in its central part (fovea) and sparser at the periphery. As a result the brain receives and processes a lesser amount of visual information, but the maximum resolution is maintained where required. Another example can be found in the auditory system of many species. Two dimensional localization of sounds is achieved thanks to the spatial filtering carried out at different frequencies by the head and the outer ears (for a review: (Blauert, 1983; Middlebrooks et al., 1989)). The Barn Owl in particular exploits a peculiar asymmetry in the feathers which make up its facial ruff. Removal of the feathers drastically reduces the accuracy of these animals in localizing auditory targets (Knudsen, 1981; Knudsen and Knudsen, 1985). Muscles act as low-pass filters on the output of motor neurons; their intrinsic elasticity and variable stiffness simplify the problem of motor control, especially in presence of an unpredictable environment ((Bizzi and Mussa-Ivaldi, 1993), Section 3.1 in this thesis). Intelligence in biological system seems to be distributed across the whole body rather than being an exclusive characteristic of the nervous system alone: to understand intelligence it is not possible separate out the role of the body.

Part of the research in robotics has been devoted to the study of morphology in both natural and artificial systems; the goal is to understand the mechanisms used by biological systems to perceive and act and, possibly, to design more efficient and robust artificial systems (Pfeifer, 2000). These studies have been successfully applied to the design of bio-morphic sensors for vision, audition, touch and olfaction (for a review see: (Barth et al., 2003; Dario et al., 1993)) and complete robotic artifacts (Beer et al., 1998; Brooks, 1990). The same approach has been used to understand human brain functions in general and resulted in a cross-fertilization between the fields of biology, brain sciences, and robotics. In this case robots can be used as platforms to test computational models of the nervous system. Physical models might be preferable, for instance if compared to numerical simulations, because they offer a "living proof" to the existence of a solution to the specific problem they address. Besides, in several cases they are more accurate and realistic descriptions of the system and of the environment, especially considered that the latter might not even be completely simulable when it involves humans and other autonomous agents (*"the world is its own best model"* (Brooks, 1991)). For these reasons it is not surprising that the majority of such applications have been in the field of motor control and locomotion. The study of orienting behavior is an example. Robotic heads have been built to study ocular movements driven by visual (Berthouze and Kuniyoshi, 1998; Capurro et al., 1995; Capurro et al., 1997; Grosso et al., 1995) as well as auditory cues (Irie, 1995; Natale et al., 2002a; Rucci et al., 1999). Inertial sensors were employed to simulate the human vestibular organ and realize inertially driven eye movements for visual stabilization (Panerai et al., 2000; Panerai et al., 2002). Inspired by the observation that in many cases neurons in the brain exhibit multi-modal response (Stein and Meredith, 1993) vision has been integrated with information from other sensory modalities. For example, Panerai et al. integrated visual and inertial information to improve visual stabilization (Panerai et al., 2002) whereas Natale and colleagues proposed the integration between visual and auditory cues to learn auditory elicited eye movement in a binocular head (Natale et al., 2002a); finally, a multi-cue approach for the control of a humanoid robot is proposed by (Cheng and Kuniyoshi, 2000).

Computational theories of motor control have been a point of contact between neuroscience and robotics. Several models have been proposed to explain the neural mechanisms converting sensory information (i.e. visual input) into motor commands (for a comprehensive description see: (Desmurget et al., 1998)). The minimum torque and minimum jerk models provide mathematical descriptions concerning torque generation for limb motion (for a review see (Jordan, 1996)). On the same line the equilibrium point hypothesis is an attractive model for the control of posture and movement (Hogan, 1985; Mussa-Ivaldi and Giszter, 1992; Mussa-Ivaldi et al., 1993). Although Gomi and Kawato (Gomi and Kawato, 1997) questioned the biological plausibility of this model, it offers a simple solution to the inverse kinematics and dynamics problems for robotic manipulators (Mussa-Ivaldi and Bizzi, 1993; Mussa-Ivaldi and Hogan, 1991). The controller of Babybot's arm was initially implemented using this approach (Metta et al., 1999).

Mechanical properties of muscles inspired the design of elastic actuators to simplify the control of robots in an unconstructed environment (Pratt and Williamson, 1995; Robinson, 2000). These actuators, and low-impedance control in general, are popular solutions to control humanoid robots (Hirai et al., 1998; Robinson et al., 1999; Williamson, 1996). A prototype of a biomorphic actuator mimicking muscle properties has been built during this thesis and is described in Chapter 3.

1.3. Cognitive developmental robotics

The paradigm of embodiment and situatedness capitalize the importance to have complete systems embedded in a real environment. For intelligence and cognition to emerge sensorimotor coordination is required; this means that the system must be able to plan meaningful actions based on sensory information. The classical approach considers planning the result of a centralized process; if several modules exist to process different inputs, coherent behavior of the overall system is achieved by combining the result of these different processing units. To some extent this is close to Descartes' view of the brain where decisions are taken by a centralized unit (homunculus). Brooks (Brooks, 1990) strongly rejects this point by proposing an architecture (subsumption architecture) where several processing units are organized in different modules working concurrently to achieve proper behavior.

Perception is directly linked to action at the level of each module; conflicts between different modules are avoided by mechanisms of inhibition and suppression. This architecture was initially employed as a design principle for insect-like robots and, later, more complex humanoid robots (Brooks et al., 1999). In the COG project, for example, the focus was on the possibility to increase flexibility, performance and reliability with adaptive behavior (Brooks et al., 1998).

Another important aspect of biological systems is that they continuously interact with the environment and adapt their behavior as a result of this interaction. In other words, the current behavior results form both perception and past experience. Inspired by studies on human development researchers have started to study the problem of adaptive behavior with a new perspective. Growing interest has been given to the new fields of developmental and epigenetic robotics (Asada et al., 2001; Lungarella et al., 2003) which aim at simulating cognitive development in artificial systems. On the one hand, this seems a constructive approach to learn how to design efficient and reliable robotic systems. On the other hand, robots can be employed by neuroscientists and developmental psychologists as "synthetic" tools to study and test models of human development.

From an engineering point of view it is useful to look at the solution adopted by nature to solve the problem of integration. In fact assembling something as complex as a humanoid robot as a collection of modules that are built separately can be very hard if not virtually impossible. Instead, the different parts composing the body and the neural circuits in the brain develop simultaneously as the result of predefined rules (phylogeny) and the individual experience (ontogeny). Learning of motor skills and acquisition of perceptual abilities in general, are not completely free but seem guided by predefined mechanisms (reflexes, motor synergies) which allow and drive exploration. This process is particularly important at birth and during the first years of life, but it is maintained active all life long to keep the system calibrated and adapt it to changes in the environment. Let us now review the most salient aspects of infants' sensorimotor and cognitive development.

1.4. Developement

One of the central issues in development is the role of phylogeny and ontogeny: that is the role played by genes and evolution in the maturation of individuals. Ontogeny considers the processes that take place during the life of an individual and is somewhat opposed to phylogeny which is more concerned with evolution and the information coded by genes. To what extent phylogenesis influences ontogenesis is not yet clear. Two opposing theories may be formulated. The first one reduces ontogenesis to a mere execution of rules specified within one's genetic inheritance, whereas the second one suggests that the external world is the main source of information shaping the human mind. It is important to notice that according to both approaches the information defining the structure of every individual would be pre-determined. It is commonly accepted that the latter is the result of a complex, dynamical interaction between genes and environment (Johnson, 1997). The nature of this interaction is not clear and still debated as it can originate at different levels (e.g. molecular) and before/after birth. It is, however, frequent to distinguish innate from acquired components, the former being characteristics common to all individuals within a specie, whereas the latter being the result of the experience unique to each of them.

In particular, innate behaviors characterize humans during the first phases of development and allow them to start interacting with the world from the early moments of their lives. Newborns are able to first interact with the environment by shifting gaze (Streri, 1993). Their attention is attracted towards interesting objects such as human faces, sounds or moving stimuli. As a crude form of social interaction babies can communicate emotions (like pain and hunger) and show imitative behavior of facial gestures (lip, mouth opening and tongue protrusion). Innate motor schemas allow newborns to perform ballistic arm movements to bring their hand to the mouth or, occasionally, to reach for objects (von Hofsten, 1982); in other cases arm motion can be visually controlled to maintain the view of the hand (Van der Meer et al., 1995).

Overall it seems that these reflexes and motor synergies constitute a raw form of sensorimotor coordination that has a twofold purpose: it allows babies to survive and feed themselves (e.g. the sucking reflex) and it provides them the ability to start gathering information about the environment and their own body. For example it has been proposed that such movements allow infants to visually tune an internal model of their own body and progressively improve reaching. At the beginning in fact motor abilities are rather limited: newborns cannot grasp or perform smooth tracking of objects. The same is true for perceptual capabilities: visual acuity is poor (objects beyond 50 cm are probably not seen clearly) and depth estimation has not yet developed. Nevertheless these capabilities emerge during the first months of life. At 4 months of age infants become successful at reaching and grasping objects and can easily move the eyes to track moving objects. These achievements are the result of several improvements at the level of the motor as well as the perceptual system. From the point of view of perception visual activity has improved and stereoscopic vision often developed whereas, at the same time, postural control is more mature, and arm muscles stronger. Thus neural maturation gradually allows for reflexive behaviors to be inhibited and makes it possible to perform independent, voluntary movements with arm, head, and hand. These abilities show that at this age the brain has already acquired an internal model of the body including its kinematics (length of body segments and their relative position) and dynamics (moments of inertia, weight, viscosity, stiffness).

Another important process taking place during development is the detection of regularities in the sensory streams which correspond to stable and constant properties of the world. This process allows children to create expectations about the events they attend to. Examples are the ability to extrapolate object motion and object persistence over occlusions. Experiments show that infants at this age start predicting reappearance of both linearly and circularly moving objects (Rosander and von Hofsten, 2003). This has been shown to be crucial as grasping for instance is facilitated by predictive abilities (the hand anticipates the timing of the reaching by flexing the fingers before tactile contact with the object). Accordingly, the ability of the brain to compute speed and trajectory to catch fast moving targets has been observed. Von Hofsten reported that infants can anticipate the meeting point between their hand and the target with an error of few degrees (von Hofsten, 1983). Expectations thus improve the infant's motor competencies and allow them to extend their understanding of external events.

Cognitive abilities at this point further improve from the interaction with objects. At 9 months of age infants use differentiated finger movements and a wider range of grasp types (Ronnqvist and von Hofsten, 1994). By repetitive trials children learn different ways of grasping objects and acquire tactile information about them. These experiences provide tactile, visual and kinesthetic information which contributes to form the infant's representation of objects. Later on (one year old) this representation is enriched when children start to explore object-object interaction (for instance how an object fits into a hole).

1.5. Self-supervised learning (what do we need manipulation for?)

Learning in both artificial and natural systems requires the exploration of the state space. The latter is defined by all possible combinations of the state variables defining the problem to be solved. The dimensionality of the state space increases quickly with the size of the state (the curse of dimensionality); in most cases an exhaustive search across all possible values is impossible or requires too much time. The issue then becomes how to explore this space so that learning is effective (the training set is large enough) but the system does not spend too much time in the exploration phase. In fact if learning is online the system has to decide when to stop exploration and start exploiting the knowledge it has previously acquired.

We want to stress here that if the learner is a physical agent situated in a real environment (e.g. the world) then learning might be simpler in this respect. In fact, the interaction with the world and physical constraints between the limbs and other parts of the body can narrow the effective state space. At the same time, the active agent can extend exploration to other regions of the state space if required; this may happen if there are ambiguities (for instance large variance in the training set) or if a particular region is more important/critical. An example is a robot that learns to manipulate objects on a table; if the robot can move, it might follow the advantageous strategy of placing itself so that the table is always on the same relative position with respect to the its body. In this case exploration during learning is limited to the space in front of the robot (the table).

Another advantage concerns the possibility to guide learning in a self-supervised fashion. By acting on the environment an agent has the ability to actively probe its properties and focus the attention on events that happen at particular instants of time (e.g. when the actions take place). Manipulation, in addition, allows the agent to merge and link sensory cues perceived from different modalities (e.g. vision and touch). If properly integrated this information draws a coherent picture of the world – and objects – that can simplify the problems of interpretation, categorization and recognition.

The traditional approach to learning in computer vision has tried to solve these tasks by using vision alone, but it has failed to create artificial systems able to work in a complicated environment (the real world). Even problems that are straightforward for humans proved to be challenging for computers (e.g. object segmentation). It seems that somehow our brain can dispel all possible ambiguities and provide us with a consistent picture of the visual world. The overall process that makes this possible is far from being understood although it has been extensively investigated by neuroscientists, physiologists, roboticists, and computer scientists. Many agree on the fact that the brain takes advantage not only of visual cues, but also of the wealth of multimodal information from other senses and from the kinesthetic experience derived from the interaction of the body with the environment. The representation of the world in adults is the result of an active process of collecting information which starts in infancy and continues all along our life. In this process manipulation is remarkably important. It enables us to access properties that otherwise would not be available (like weight, roughness or softness) while on the other hand it gives us the possibility to actively control the investigation of these modalities (active touch). Once an object has been grasped, in fact, it is possible to exert explorative strategies (squeezing, weighing, rotating, to mention a few) and autonomously carry out the investigation of the object's properties.

To summarize, manipulation establishes a link between action and perception that facilitates learning and enable the acquisition of a multimodal representation of the world. In the brain there is probably more than a single area responsible for coding this representation. For instance, two main pathways have been individuated which have complementary roles (Milner and Goodale, 1995); these two streams (the *dorsal* and *ventral* streams) code visual information depending on the task to be performed. More in general, according to Jannerod (Jeannerod, 1994) the brain has a pragmatic representation of the attributes relevant to action. This is somehow different from the semantic representation grouping together all information necessary for object recognition and categorization. The former includes parameters relevant for shaping the hand according to the size, weight and orientation of the object we are going to grasp. The latter has the function of forming a perceptual image of the object in order to identify it. In dealing with an object the brain has to solve the following questions: *what* the object is, *where* it is and *how* to handle it. The representation of *where* and *how* constitutes the pragmatic representation which is directly related to action. The representation of *what* is related to the conscious perception of the object and corresponds to its semantic representation.

The *where* representation is completely different and does not directly involve knowledge of objects. The representation of *what* the object is and *how* it can be manipulated are normally integrated but under certain conditions can be dissociated. This was proven by behavioral studies of reaction times in humans, by anatomical studies performed in monkeys, and from the observation of patients with lesions in the posterior parietal cortex (for a review see: (Jeannerod, 1994)).

Although separated, both representations are based on knowledge that is acquired (learned) by interacting with objects. Even when answering the *what* question, information about shape, size and weight might prove helpful to bias the recognition in cases when only ambiguous cues are available. Similarly, the same cues are used during grasp to anticipate the shape of the hand thus to achieve a stable grip. Visual information in this case activates the brain circuitry responsible for the pragmatic representation of the object to be grasped which controls the orientation of the hand, its maximum aperture and the opposition space.

Recent studies on the monkey premotor cortex have revealed the existence of neurons which code a similar pragmatic representation of objects (Gallese et al., 1996). A group of neurons located in the monkey premotor cortex (area F5) is activated both when producing a motor response to drive an object-directed grasping action and when only fixating a graspable object. This population of neurons seems to constitute a vocabulary of motor actions that could be applied to a particular object. This response is somewhat reminiscent of Gibsonian affordances because it represents the ensemble of grasping actions that an object affords ((Gibson, 1977) see also Section 7.2).

Finally, the link between action and perception is important because it may be involved in the process of understanding the actions performed by others. This is supported by the discovery of another class of neurons (Fadiga et al., 2000) which not only fire when the monkey performs an action directed to an object, but also when the monkey sees another conspecific (or the experimenter in this case) performing the same action on the same object (*mirror neurons*). Clearly knowing in advance the range of affordances given the object facilitates the interpretation of the observed gesture by constraining the space of possibilities to those suited for the context.

1.6. A developing robot

Motivated by these observations Giorgio Metta (Metta, 2000) addressed the problem of building a humanoid robot mimicking at least some aspects of infant sensorimotor development. Metta focused on eye-head and eye-hand coordination and the integration between vision and vestibular information. The robot starts from a very limited set of competencies; more complex behaviors are built (emerge) as the result of the interaction with the environment. The initial competencies consist in few perceptual abilities and a set of reflexes providing the robot an early form of sensorimotor coordination. Immature motor system is simulated by adding noise to the executed commands. Initial reflexes and noise thus bootstrap the system and permit the exploration of its state space. Developmental rules guide the system through learning of more and more complex behaviors.

For example, in its initial state the robot cannot control the eyes, the neck and the arm in a purposive way. Development proceeds as follows. Small eye movements are performed randomly to estimate an internal model of the ocular system; this information is later on used to control the eye in a goal directed mode (tracking). As soon as the robot can track moving targets an ocular map is learnt to perform rapid eye movements (saccades) to increase tracking performances. The probability to move the additional degrees of freedom in the neck is increased with time. So the neck does not move at the beginning to facilitate learning of the eye movements; however as the latter progresses the robot starts controlling gaze by moving both eyes and neck simultaneously. At the same time the robots learns how to integrate vestibulo-ocular reflex (VOR) and opto-kinetic response (OKR) for better stabilization. The development of reaching exploits an initial rough coordination between gaze and arm; owing to a reflexive moment the arm follows the direction of gaze to keep the hand within the robot’s visual field. This reflexive behavior together with a noise component endows the robot with an early form of reaching. By fixating the arm end-point after each reaching trial the robot is able to match head (i.e. fixation point) and arm posture and eventually fill a motor-motor map. The latter is afterwards used to substitute the reflexes.

Let us summarize the key aspects of the approach. The noisy initial configuration coupled with basic reflexes starts exploration (eye movements and reaching). The initial behavior is very simple (only the eyes move), more complex modules are added as development progresses (the neck and the arm). New modules are built on top of the previous ones to achieve integration. The goal of each modules is to solve/learn a particular task (i.e. eye movements, visual

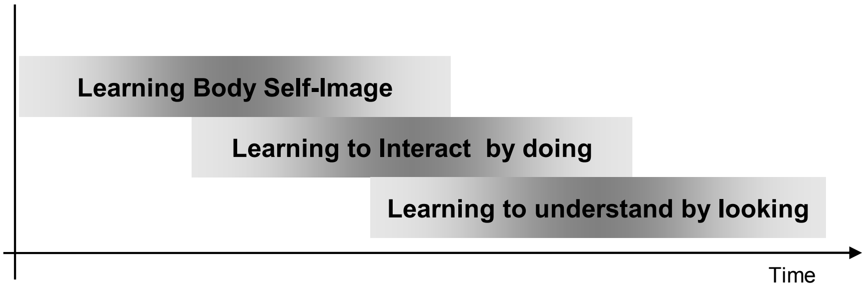

Learning Body Self-Image
Learning to Interact by doing
Learning to understand by looking
Time

Figure 1.1. The development of the robot takes place by following this simplified schema. The first stage involves learning about the robot's own body (limb size and dynamics). The second one concerns learning to interact with the environment whereas the third (hypothetical) stage is devoted to learning event interpretation. Stages are not completely separated as they evolve together; consequent stages rely on the competencies acquired in the previous ones (compare to Table 1-B).

stabilization, reaching), the developmental program guides learning by controlling "external" parameters (the amount of noise, the probability of motion of the neck) and by enabling or inhibiting the initial reflexes. It is finally important to stress that there are not two separate phases of learning/calibration and functioning. Conversely, development and learning happen online during normal system operation.

We propose here some additional steps to continue the work of Metta. The main idea of the approach is that development moves from the exploration of the robot's body to the interpretation of the outer world. A first broad classification divides this process in three stages (Figure 1.1). The first stage is devoted to learning a *body-map*: the robot learns physical properties of its own body and to distinguish it from the rest of the world. This happens while basic motor and perceptual skills are acquired. For example the robot learns gaze control, eye-head coordination and reaching to touch an object. Based on these competencies in a second stage the interaction with the external world is investigated. We call this process *learning to interact* as it involves finding out how to act on objects and handle them. Initially the robot tries simple actions, like power grasp (approaching the object with the hand full open) or pushing/pulling an object. This interaction enables it to start acquiring information about the entities acted upon. As a result, more efficient and complicated explorative strategies are developed. Differentiated grasping can then be applied to objects, depending on the goal of the action; for example the robot can discover that small objects are more suitable to be grasped by using precision grip (the thumb opposing the index finger) whereas power grasp is more effective for big, heavy objects. The third stage involves *learning to understand/interpret events*.

This happens by learning association between what the robot sees and what it has done in the past. The resulting link enables the robot to associate meaning to the events it perceives on the basis of its own previous experience. Further, new ways to handle objects may be discovered by observation (imitation). Table 1-B summarizes this developmental path; for each phase it details the goal of learning, the goal of the system and the link that is established between action and perception. The rightmost column reports the delay that exists between the two. Notice that there is a strict connection with the three stages hypothesized in Figure 1.1; the delay gets longer as development progresses and cognitive abilities emerge. Shaded cells in Table 1-B represents those aspects that at least in part were addressed in this thesis. The final discussion will come back to these issues to better delineate future work and its connection with what was done.

1.7. Outline

We have just given an overview of the motivations behind the approach we followed. The remainder of the thesis is organized as follows. Chapter 2 describes the robotic setup that was used to carry out the experiments. The same chapter details also the software and hardware architecture of the robot; given the complexity of the system both of them have been an important aspect of this research. In particular, the software architecture allows the system "to grow" as new modules (sensory as well as motoric) will be introduced. For a humanoid robot this aspect is fundamental as it is very common to reach a level of complexity that jeopardizes manageability. The hardware architecture includes aspect of mechanical design. A biomorphic actuator with spring-like properties was realized during the thesis and its realization and test is reported in Chapter 3. The following chapters describe the developmental path of the robot. Chapter 4 deals with the robot's visual system and the implementation of the motor behaviors which enable the robot to visually explore the world. Chapter 5 and Chapter 6 are concerned with the exploration of the robot's own body and the external environment respectively. They correspond to the second stage of the developmental process described above. In the last section we draw the conclusions and discuss future work. Finally, more technical details about specific algorithmic implementations that were purposely overlooked in the thesis are reported in the Appendix.

Development	**Goal of learning**	**Goal of the system**	**Link between perception and action**	**Timing between events**
Gazing	Head-eye coordination	Look around	Control gaze based on visual input (smooth pursuit)	Immediate effect
Pre-reaching	Approach an object	Touch	Controlling arm and hand movements in space	
Power grasping	Eye-hand coordination based on object position and object motion	Grasp	Anticipatory closing of the hand	Short delay between action onset and consequences
Differentiated grasping	Adjustment to object shape and size	Grasp appropriately	Eye-arm-hand coordination based on objects' shape	
Object manipulation	Objects' affordances	Handle objects appropriately (use)	Eye-arm-hand coordination based on actions to be executed on objects	
Imitate acts on objects	Associate what is seen with what the system can do	Action's interpretation	What I do looks like what I see	Long delay between action and perception
Act to communicate	Associate what is seen (perceived) with "meaning"	Action's meaning	What I do generates some reactions	

Table 1-B. A possible developmental path for the robot. Cells represent successive motor and perceptual competencies acquired during development. The table details for each phase the goal of the learning, the goal of the system and the link that is established between action and perception. The rightmost column reports the delay occurring between action and perception; the latter is related to the time course of development and the stages as reported in Figure 1.1. Shaded cells are topics that in part were addressed in this thesis.

Chapter 2

System's architecture

Humanoid robots are usually quite complex; their design is hence critical from several points of view. The computational power required to control a robot very often exceeds that of a single machine and a cluster of computers has to be used. As new behaviors and perceptual abilities are added, the software design can make a difference: it constitutes the basis for building a complicated architecture, where behaviors coexist and cooperate in a coherent and meaningful way. This chapter deals with the software and hardware aspects more closely related to the engineering and design of the robot used for this work. Section 2.1 and 2.2 describe the mechanical components of the robot and its sensors; the description will give enough details to let the reader understand the remaining part of the thesis. Section 2.3 describes the software architecture of the robot as the result of more than a few efforts to have a comfortable and manageable platform to work with. Finally, Section 2.4 describes aspects related to the learning architecture.

2.1. Babybot's body

Babybot is an upper torso humanoid robot with a head, a manipulator arm and a hand. The head has five degrees of freedom (d.o.f.). Three of them are associated with the two cameras to achieve independent panning and coupled tilting. The two remaining d.o.f. allow panning and tilting respectively, at the level of the neck. The manipulator is a 6 d.o.f. Unimation PUMA 260, mounted with the shoulder horizontal to better resemble a human arm kinematics (Figure 2.1).

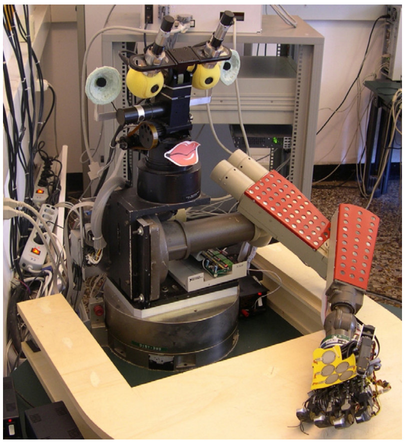


Figure 2.1. The robotic setup, the Babybot.

A 5 fingered robot hand is attached to the arm end point. Each finger has 3 phalanges; the thumb can also rotate toward the palm. Overall the number of degrees of freedom is hence 16. Since for reasons of size and space it is practically impossible to actuate the 16 joints independently, only six motors were used. Two motors control the rotation and the flexion of the thumb. The first and the second phalanx of the index finger can be controlled independently. Medium, ring and little finger are linked mechanically thus to form a single virtual finger controlled by the two remaining motors. No motors are connected to the fingertips; they are mechanically coupled to the preceding phalanges in order to bend in a natural way as explained in Figure 2.2.

The mechanical coupling between gears and links is realized with springs. This has the following advantages:

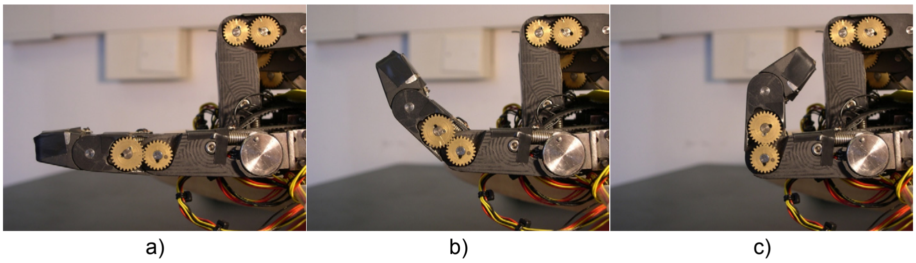


Figure 2.2. Mechanical coupling between phalanges. The second phalanx of the index finger is directly actuated by a motor. Two gears transmit the motion to the third phalange. The movement is respectively of 90 and 45 degrees.

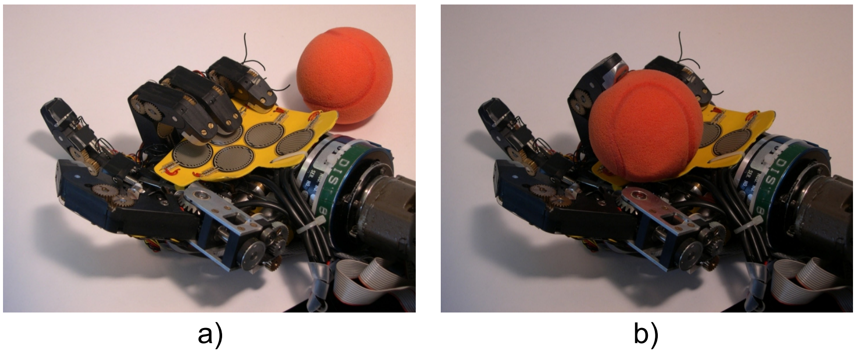


Figure 2.3. Elastic coupling. (a) and (b) show two different postures of the hand. Note however that in both cases the position of the motor shafts is the same. In (b) the intrinsic compliance of the medium finger allows the hand to adapt to the shape of the object.

- The coupling between medium, ring, and small finger is not rigid. The action of the external environment (the object the hand is grasping) can result in different hand postures (see Figure 2.3).
- Low impedance, intrinsic elasticity. Same motor position results in different hand postures depending on the object being grasped.
- Force control: by measuring the spring displacement it is possible to gauge the force exerted by each joint.

The robot's sensory systems include vision, audition, touch, proprioception, and inertial sensing. Proprioceptive feedback is achieved by means of the motor optical encoders. Two cameras rotating with the eyes and two microphones attached to the head respectively provide visual and auditory feedback. During the acquisition, images are sampled non-uniformly to mimic the distribution of receptors of the human retina. More pixels are acquired in the central part of the image (fovea) and less in the periphery (mathematically the distribution is approximated by a log-polar function, see Section 4.1 for a more detailed description). The head mounts a three axis gyroscope that provides the robot with an artificial equivalent of the human vestibular system (in Figure 2.4). This sensor measures inertial information consisting of angular velocity along three orthogonal axes. It can be used for stabilizing the visual world efficiently and coordinating the movement of the head with that of the eyes. For the hand, Hall-effect encoders at each joint measure the strain of the hand's joint coupling spring. This information jointly with that provided by the motor optical encoders allows estimating the posture of the hand and the tension at each joint. In addition, force sensing resistor sensors (FSR) are mounted on the hand to give the robot tactile feedback. These commercially available sensors exhibit a change in conductance in response to a change of

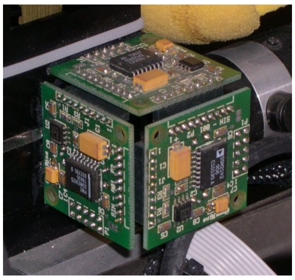


Figure 2.4. The inertial sensor of the Babybot developed at LIRA-Lab. It consists of three mono-axial sensors arranged along three orthogonal axes.

pressure. Although not suitable for precise measurements, their response can be used to detect contact and measure to some extent the force exerted to the object surface. Five sensors have been placed in the palm and three in each finger (apart from the little finger) (see Figure 2.5). Further proprioceptive information is provided to the robot by a strain gauge torque/force sensor mounted at the link between the hand and the manipulator's wrist. This device is a standard JR3 sensor designed specifically for the PUMA flange. It can measure forces and torques along three orthogonal axes (Figure 2.4).

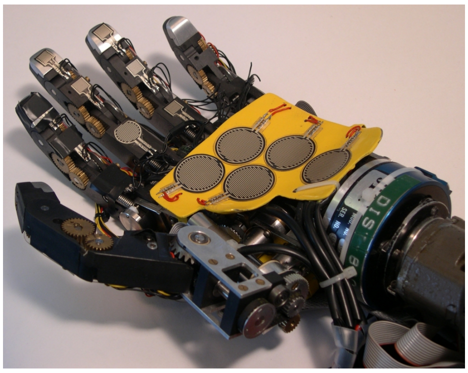


Figure 2.5. Tactile sensors. 17 Sensors have been placed: five in the palm, three on each finger apart the little finger. In this picture the sensors in the thumb are hidden. The short blue cylinder that links the PUMA wrist to the hand is the JR3 force sensor.

2.2. Interface cards

The robot sensing includes some digitizing interfaces, special signal conversion and conditioning modules. The link between the hardware and the robot is provided mainly through standard PCI/ISA cards and serial/parallel ports. Motor control also requires special hardware to generate the appropriate signal to drive the motors. At this level the Babybot follows a very traditional approach, as it is actuated by DC motors. All of them have their specific control cards and power amplifier. In the case of the arm (PUMA), the original Unimation linear amplifier was modified and interfaced to the control card on standard PC. The head and hand joints are controlled through a bank of switching amplifiers (PWM). Each control card has a DSP which can be programmed to some extent to generate the desired control strategies. For example the head is controlled with a high gain controller while for the arm we employed a low-stiffness control schema (Section 5.1 and 6.1). Encoder signals are collected by the same control cards. Motor control card have also analog inputs which can be used when necessary. For example this solution was used for the inertial sensor.

Images are provided by standard CCD color cameras and they are sampled at full frame rate by frame grabbers with the common BT848 chipset. The original images are sub-sampled as early into the processing as possible to the desired resolution and format (the log-polar format, see Section 4.1). Auditory signals are sampled at 44 KHz by a standard sound card. The signal coming from the microphones is amplified and conditioned appropriately before sampling. Tactile sensors have their own microcontroller and AD converter. Digital values are sent to a PC though a serial line. Hall-effect analog signals are sampled by another card with a bank of AD converters.

The hardware is heterogeneous since it evolved from previous implementation of the Babybot. Control cards have different CPUs, sampling rates, DSP and software interface. The same applies to the set of PCs where the hardware is interfaced to. They range from older Pentium to the latest generation PIV. Presently the robot is controlled by 14 machines connected via two separate 100 Mbits Ethernet networks. One network is dedicated to control signals, the other mostly to visual processing (Figure 2.6).

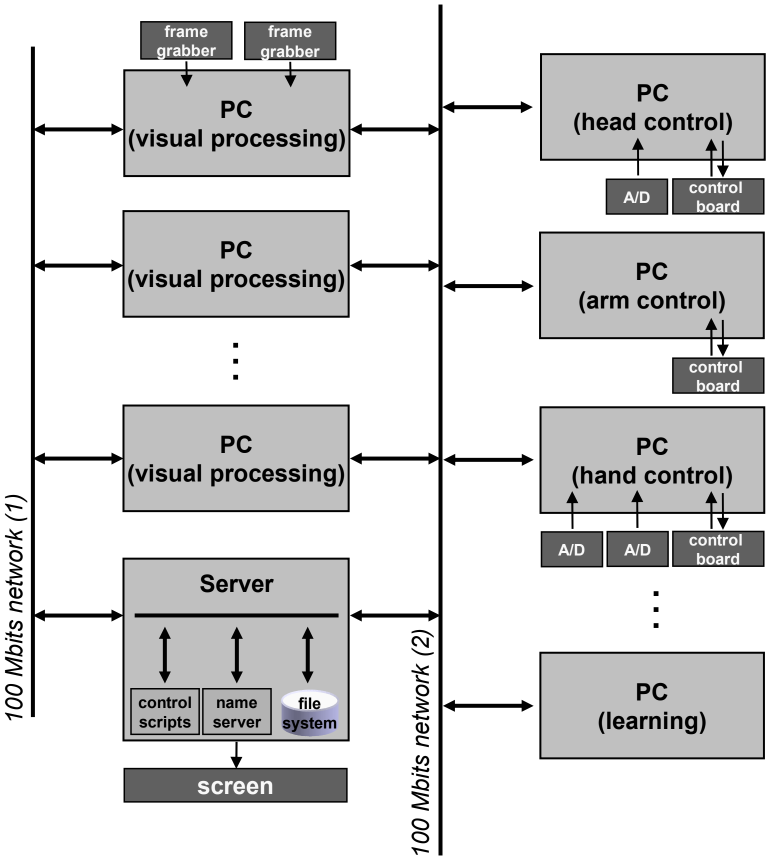

frame grabber
frame grabber
PC (visual processing)
PC (visual processing)
PC (visual processing)
Server
control scripts
name server
file system
screen
100 Mbits network (1)
100 Mbits network (2)
PC (head control)
A/D
control board
PC (arm control)
control board
PC (hand control)
A/D
A/D
control board
PC (learning)

Figure 2.6. Hardware architecture. It consists of two separated switched networks. One network is dedicated to visual processing; the other to control signals and other data. The nodes are computers (from 2.4Ghz PIV to 750 Ghz PIII) connected either to one or both networks. The machine equipped with frame grabbers acquires the images and broadcast them across network 1. Nodes with motor cards drive the robot and receive position feedback (head, arm and head); in some cases supplementary cards may be used (e.g. in the case of the hand the acquisition of the magnetic encoders and the tactile sensors requires additional A/D converters). Other machines can be connected at will to perform other processing (e.g. learning). The server provides access to a shared file system and runs the name service. From the server console it is possible to launch control scripts, which remotely execute new processes and manage the running ones (this includes termination, connection and disconnection).

2.3. Software architecture

The first important aspect concerns the operating system. The choice is among more standard solutions such as Microsoft Windows NT and Linux or operating systems specifically designed with performance issues in mind (real time operating systems such as Linux RT, VxWorks or QNX). Microsoft Windows (NT at the beginning and XP more recently) has been preferred in general because of the hardware supported; device drivers on Windows are often reliable and tested, especially in the case of general purpose, cheap hardware like network cards, video adapters and frame grabbers. On the other hand Windows is not optimized for performance nor is it designed for real-time applications. QNX (http://www.qnx.com/) is an operating system specifically designed to achieve real-time performances. Features include fast interrupt latency, low overhead context switches and preemptive priority-based scheduling. For instance QNX can run a closed loop with period of about 1ms, whereas Windows NT hardly handles loops below 10 ms. For motor control, reducing the control loop cycle time by an order of magnitude can make the difference; in other cases, however, fast response is not strictly required. Vision for instance is often bound to the PAL standard (25 Hz) and, in any case, to computational limits which hardly allow decreasing computations below 30/20 ms.

The goal of the software architecture is to make it easier to develop new modules and to integrate them in the system. Moreover, as software engineering rules suggest, the software should be divided in modules (objects) implementing different function which can be reused as necessary. This avoids programming errors and allows sparing a good amount of time as well-tested modules are recycled within the system. This is not something that can be achieved easily in a system like the one that was described above. Most of the modules are directly interfaced to the hardware; this is especially true given that, as it is virtually impossible to use a single CPU, some kind of protocol is required to connect different processing units together. As a result the software makes extensive use of operating system facilities (to access the hardware, schedule processes, achieve synchronization and communication) and gets easily intermingled with the particular platform and cards used. To improve manageability it is in general convenient to separate the details of the low-level hardware (usually the interface with the device driver) from the software implementing a particular algorithm.

At the moment this thesis is being written, most of the architecture runs Windows (NT, 2000 and XP). Anyway the software was designed with the idea to have a heterogeneous architecture employing the operating system more suitable for a given purpose. Thus, the architecture allows a software module to be compiled and ran on different architectures irrespective of the operating system (QNX, Windows and Linux were successfully tested). Not all the modules are completely implemented on all the architecture (for instance motor control does not run on

Linux). However the software architecture is designed to minimize the effort required to adapt the high level modules (signal processing, communication as well as motor control) to different operating systems and hardware platforms. Let us now have a look at the implementation details.

2.3.1. Communication protocols

The software is designed with the goal to achieve two levels of transparency. *Access transparency* means that different modules access to data independently of the underlying hardware; thus, for instance, differences in the way data are stored and organized in memory are handled by low-level classes implementing communication. The same classes realize *location transparency*; this means that modules running on different machines are not concerned with communication details. The library offers a unified communication layer that is used by all processes. The library automatically makes use of the more appropriate protocol depending on the situation. For instance if two processes require a connection and are running on the same machine, the library establishes a connection through shared memory. Compatibility between different machines is achieved by using the Internet protocol suite which is a *de facto* standard for communication. In particular we used TCP/IP, UDP and Multicast (MCAST). TCP/IP offers a reliable connection where a transport layer guarantees that all sent packets are correctly received by the client. UDP is a connectionless protocol. It is more efficient than TCP/IP and does not require that the connection is established in advance. UDP does not guarantee that packets are not lost on the way between sender and receiver. If the network load is not too high and the CPUs are not completely busy this happens rarely. Anyway UDP is more suitable for those cases where loss of packets is not a problem (for instance in data streams). Multicast is another connectionless protocol which enables a single sender to transmit data to multiple receivers at the same time. It is similar to UDP, with the difference that the same packet can reach multiple clients (by means of a sort of subscription mechanism). Besides TCP/IP, QNX implements a proprietary protocol specifically designed for real-time (QNET). Among QNX machines this is obviously the most efficient solution and for this reason its support was included in the library. However TCP/IP, Multicast and UDP were used more often because they allow the connection between heterogeneous machines.

The last piece to achieve *location transparency* is naming. Each communication channel is assigned a name that is registered on a shared database; a name service handles name queries and keeps tracks of the TCP/UDP and Multicast ports used on each machine. In this way it is possible to instantiate and destroy channels at run-time without the need to keep a static list of the ports used on each machine.

2.3.2. Hiding the Operating System

For the task of encapsulating the operating system we relied on existing software. In particular we found it convenient to base our implementation on ACE (Adaptive Communication Environment), an open-source library that among many things provides a tiny object-oriented OS wrapper. For more information about ACE see (Schmidt, 2003; Schmidt and Huston, 2002). ACE runs on Windows, Linux, and QNX that were also our target operating systems. Basing our implementation on ACE allowed running all our code on any of these operating systems. From our point of view ACE provided a common C++ class interface for the communication code and the OS wrapper. Advanced ACE functionalities were not fully exploited. We preferred to take a minimalist approach and rely on the smallest subset of ACE that allowed solving our tasks. The library is called YARP (Yet Another Robotic Platform); following the open-source philosophy, it was made freely available on SourceForge (http://yarp0.sourceforge.net/).

Most part of the communication code is profoundly inspired (and recycled) from a previous version developed at MIT (Fitzpatrick, 2003) and was tested extensively on the humanoid robot Cog on QNX 4.25. The latest implementation has been completely rewritten (using ACE) but it maintains the same high level interface. The communication code is a C++ templatized set of classes contained in a specific static library. The main abstraction for inter-process communication is called a "port". A port template class can be specialized to send any data type across an IP-network

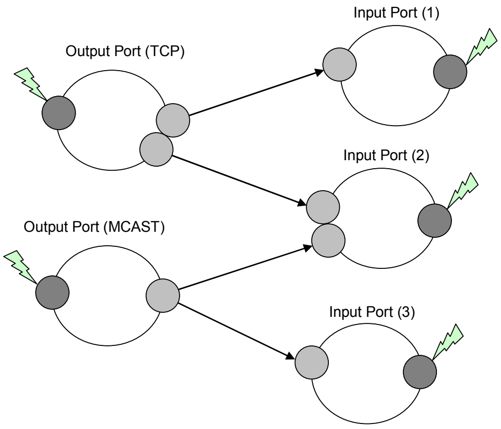


Figure 2.7. The YARP communication architecture, simplified schema. Five ports are represented in a hypothetical configuration; each port consists in a command receiver (dark gray) and one or more portlets (light gray). Portlets are active objects instantiated to handle connections: the TCP port requires a portlet for each connection whereas the Multicast port instantiates only one. Notice also that input ports may receive from different protocols.

relying on a set of different protocols. Depending on the protocol different behaviors can be obtained – as described in the previous section the implemented protocols include TCP, UDP, Multicast (MCAST), QNET, and shared memory. A port can either send to many target ports or receive simultaneously from many other ports. A port is an active object: a thread continuously services the port object. Being an active object allows responding to external events at run time, and for example it is possible to send commands to port objects to change their behavior. Commands include connecting to another remote port or receiving an incoming request for connection and since all this can be done at run-time it naturally enables connecting/disconnecting parts of the control system on the fly.

Figure 2.7 shows an exemplar structure of the port abstraction. Each port is, in practice, a complex object managing many communication channels of the same data type. Each port is potentially both an input and output device although for simplicity of use only one modality is actually allowed in practice. This is enforced by the class definition and the C++ type check. Each communication channel is managed by a "portlet" object within the main port. Different situations are illustrated in Figure 2.7: for example an MCAST port relies on the protocol itself to send to multiple targets while on the contrary a TCP port has to instantiate multiple portlets to connect to multiple targets. In cases where the code detects that two ports are running on the same machine the IP protocol is replaced by a shared memory connection. In Figure 2.7 a special portlet is shown (dark gray): it is called a "command receiver". As already mentioned its function is that of receiving commands to connect, disconnect, or generically operating on the port. Further ports can run independently without blocking the calling process (if desired) or they can wake up the calling process on the occurrence of new data. In some cases synchronous communication is allowed (TCP protocol).

Protocols can be intermixed following certain rules. Different operating systems can communicate to each other. QNET protocol is an exception and it is only valid within a QNX network. YARP communication code leads to a componentization of the control architecture into many cooperating modules. The data sent through ports can range from simple integral types to complex objects such as arrays of data (images) or vectors. Thus controlling a robot becomes something like writing a distributed network of such modules (the layer we called "Experiments").

In addition, YARP contains supporting libraries for mathematics and robot type computation (kinematics, matrices, vectors, etc.), image processing (compatible with the Intel IPL library), and general purpose utility classes. We also designed a few modules based on existing Microsoft technology to allow remote controlling Windows machines (this support comes naturally on QNX). In short, these scriptable modules complete seamlessly the architecture allowing the design of scripts to bring up the whole control structure and connect many modules together.

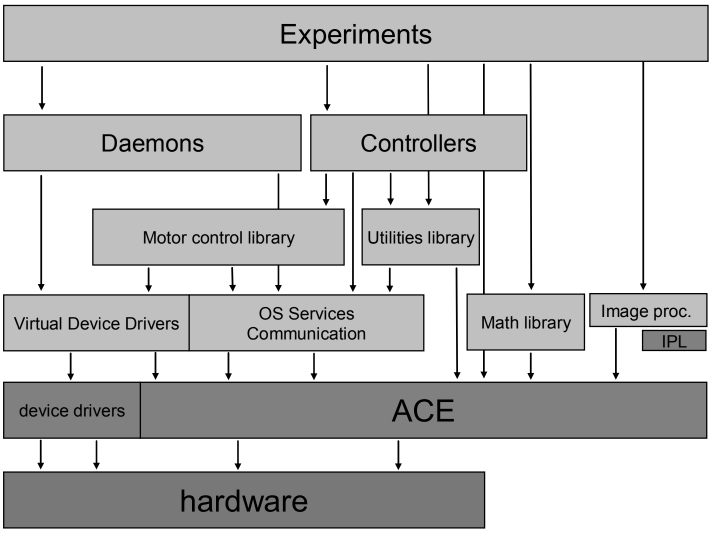

Experiments
Daemons
Controllers
Motor control library
Utilities library
Virtual Device Drivers
OS Services Communication
Math library
Image proc.
IPL
device drivers
ACE
hardware

Figure 2.8. YARP libraries: dependence chart. Dark gray represents third part software and libraries; light gray are libraries and software modules that are part of YARP. All blocks (excepted virtual device drivers) use ACE to make the software platform independent. Details about specific blocks are reported in the text.

As an aside a Matlab interface to ports has been implemented. This allows building Matlab modules (e.g. .m files) that connect to the robot to read/write data. There are basically two advantages: i) complex algorithms can be quickly implemented and tested relying on Matlab existing toolboxes, ii) an additional level of scripting can be realized within Matlab. Matlab provides a relatively efficient and easy to use display library that can be used to visualize the functioning and performance of an ongoing experiment.

In summary, Figure 2.8 presents schematically the link and dependences between the YARP libraries.

2.3.3. Robot independent code

One of the goals in writing our control architecture has been that of simplifying the programming of a complex robotic structure such as a humanoid robot. As described in Section 2.2, control cards come in many different flavors and programming them is usually painful. It would be much better if a standardized interface, or even a suitable abstraction, were available.

To solve the first problem we defined a "virtual" device driver interface into YARP. To solve the second, we encapsulated the control of parts of the robot (head, arm, frame grabbers, etc.) into a standardized template class hierarchy.

In short, the virtual device drivers bear much of their structure from the UNIX device drivers. Each card’s driver class contains three main methods: Open, Close, and IOCtl. The latter is the core of the interface. Each device accepts a set of messages (with parameters) through the IOCtl call. Each message accomplishes a specific function. Two different control cards supporting roughly the same commands can be easily (as it was done in our setup) mapped into exactly the same virtual device driver structure, although clearly the implementation might differ.

The next layer is a C++ hierarchy of classes which through templates includes both the specification of the controlling device driver (e.g. the head is controlled through a certain control card) and the idiosyncrasies of the particular setup (e.g. wiring of the robot might differ, or initialization might require different calibration procedures). This hierarchy is shown in Figure 2.9.

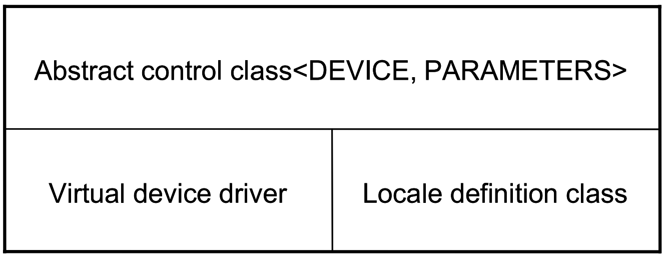


Figure 2.9. The structure of a control class for a generic device. The virtual device driver provides a generic interface to the hardware. Idiosyncrasies of the particular setup (wiring of the robot, initialization procedure) are implemented in a separate class (“Local definition class”).

2.3.4. Robot specific interface

The real “communication” with the robot is carried out through a set of binary modules that use the device driver structure described in the previous section. Module customization is at this stage accomplished through configuration files. In the YARP language these modules are called daemons (a term borrowed from UNIX). The daemons directly interact with the remainder of the robot software through YARP ports, and in general they export very specialized communication channels. For example the frame grabber has an output port of type “image” and the head control daemon an input port that accepts velocity commands. There are no specific restrictions on the type of ports exported by a daemon, since any type of state information about the modules might be required.

Further, some of the daemons accept or send commands of a special type that are generally used to communicate status information. A bus structure based on the Multicast protocol has been implemented to transmit and receive these special messages (called “bottles”). YARP bottles may contain any type of data or even a group of heterogeneous elements of different types. The structure contains identifiers to properly decode messages and interpret the data. YARP bottles create a network within the network of behaviors to realize a high-level control and coordinate a large number of modules.

2.4. Learning architecture

This layer describes an arrangement of YARP modules that tends to repeat across our robotic architecture. This is not formally into YARP proper but simply an implementation of a particular experiment relying on YARP libraries. Conceptually it forms a layer where to build more sophisticate experiments since for example it provides simple motor control and sensorimotor coordinative behaviors. Overall they could be seen as very high level commands that support positioning, gazing, reaching for visually identified objects, and grasping them.

Grossly speaking, autonomous learning requires a slightly different approach from classical supervised paradigms where data is presegmented and simply fed into a function approximator. Autonomous learning is perhaps closer to reinforcement learning in that it requires action and proper behaviors (exploratory) to gather the training set. Necessarily our architecture will require bootstrapping behaviors supporting the construction of the training set. The question of how much explore and how to get quickly to a solution is an open one in reinforcement learning and unfortunately reinforcement learning itself tend to be difficult, requiring a very large number of samples. In addition, in the case of a real robot we should not allow "spurious" or random control values to get to the low-level controllers; at the basis of any control strategy we should probably have a reasonable "safe" explorative procedure and certainly not a complete random one. Self-supervised procedures can be identified (similar in spirit to feedback error learning) and given the appropriate amount of exploration they can quickly approximate the desired sensorimotor coordination pattern.

When data samples are available in sufficient number with respect to the size of the parameter space of the function approximator of choice, the system can start learning and using what has been learnt up to date; necessarily in the long run the influence of explorative behaviors should be reduced. At least two possibilities exist here: learning could be implemented either in batches or fully online. The specific strategy is mostly a function of the algorithm and specific implementation of the function approximation. Inhibition or a functional equivalent should take care of reducing or mixing up exploration with actual "exploitation" of the acquired behavior.

Our discussion is only focused here on the function approximation problem since a good part of the sensorimotor behaviors can be actually well implemented by mapping sensory values onto motor commands or the opposite (e.g. feedback error learning or distal learning). Another constraint on the design of explorative behaviors is that they should mostly "explore" the space that will be used in the future. Failure to do so might result in very poor performance.

The learning algorithm can be conceptually divided in two parts: the one providing the "learning signals" sometimes called the "critic", and the one doing the behavior called the "actor". This distinction is important in motor control problems since the actor must be extremely fast and should work in a small delay regime. On the other hand, the critic could take even seconds or minutes to process the training data and provide infrequent adjustments to the actor's parameters. We maintained as much as possible (apart from trivial cases) this distinction within our system. This division is to some extent compatible with biological mechanisms of learning being these, for example, the rates at which synaptic changes and growth processes develop in the brain compared to actual spikes' travel times.

Figure 2.10 sketches the modules required for each actual behavior acquisition. At the moment of writing we have only conducted a few experiments with the combination and definition of modules presented here. Examples of explorative components are (at the moment) bounded random behaviors (used when training the hand localization map) or early motor synergies connecting and generating motion of different joints and even different limbs. In learning to reach, these synergies can be exploited to bias the exploration space and avoid random movements. Whenever learning relies on multiple cues, such as visual and motor, having an initial coordination (although imprecise) can be advantageous. One net effect would be the reduction of the learning space that needs to be explored before getting to a reasonable behavior. This strategy was used in our previous work (Sandini et al., 1997).

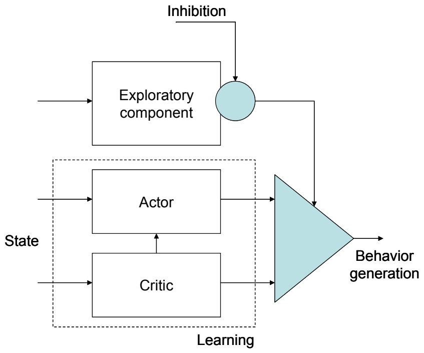


Figure 2.10: A module for learning sensorimotor coordination.

The actor and critic modules in our experiment consisted of a simple batch learning backpropagation neural network. Although not the best, it proved to be quite reliable so far (in particular we adapted a fast C implementation of the backpropagation (Anguita et al., 1994)). Backpropagation has been extensively tested and its behavior very well characterized in literature. Consequently, it is much easier to understand especially when things do not go as expected. The implementation maintains the separation of actor and critic to the point of having a slow batch learning method as critic, and a distinct process providing the behavior. Naturally, given the overall robot architecture, the two modules can be even running on two different machines.

Inhibition and the control of activation and coordination of many behaviors is still argument of further research and no definite implementation has been reached yet. Figure 2.11 shows the combination of many blocks of this type. In this case too, the realization is completely hypothetical since testing has not been performed yet.

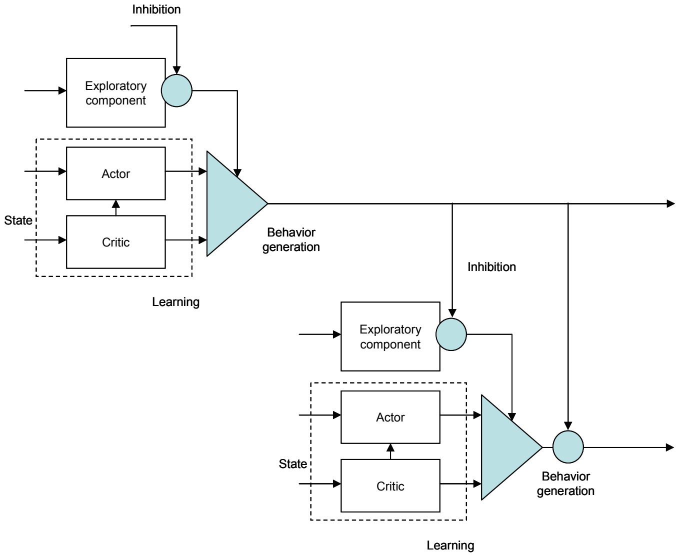

Inhibition
Exploratory component
Actor
State
Critic
Behavior generation
Learning
Inhibition
Exploratory component
Actor
State
Critic
Behavior generation
Learning

Figure 2.11. The combination of learning modules in a hypothetical subsumption arrangement.

Chapter 3

A biologically inspired elastic actuator

Although usually we are tempted to think of intelligence as the outcome of some software algorithm the physical architecture of the agent plays a critical role. The intrinsic soft compliance of the skin, for instance, allows humans to successfully grasp an object without the need to place the fingers exactly at the required position around it. Indeed, the study of biological systems tells us a very important story and suggests that a suitable physical structure can turn a difficult problem into an easy one. Evolution rarely found solutions to the problems it had to face by improving neural computations alone, but rather went through physical adaptation. As discussed in the introduction the study of these aspects has been of paramount interest in robotics; among the others, the problem of motor control is one of the most important and constitutes the main point of contact between neuroscience and robotics. If on the one hand robots are getting more and more precise and reliable in performing specific tasks, on the other they are still far from achieving good results in more general cases. The contrary is true for biological systems which are not precise and reliable in particular situations but can successfully accomplish tasks they were not specifically designed for. As one of the most striking differences between artificial and natural systems lays in the physical properties of their actuators, part of the research described in this thesis was devoted to the study and the realization of an innovative actuator, mimicking the characteristics of human muscles. This chapter provides a description of the prototype and presents the results that were achieved.

3.1. Biology and motor control

Usually the desired trajectory of either a limb or a robotic manipulator is expressed in a convenient extrinsic coordinate system, often referred to as task-coordinate frame. Most goals are in fact naturally expressed in a Cartesian coordinate system. In the case of a robot, the system has to convert this trajectory into joint-coordinates (for example joint angles) and compute the torques that must be applied by the motors to achieve the predefined motion. The former problem is called the inverse kinematics, whereas the latter is commonly referred to as the inverse dynamics. Both problems are thought to be solved by the central nervous system (CNS) of humans and animals, although we do not know the exact mechanisms that are used and the reference-frames in which this computation is carried out. The computational solutions developed and largely used in robotics have not proven to be of much help in this regard. For the same reason robots are very efficient at solving clearly defined tasks in structured and well known environment, but they perform poorly in all those cases when a part of the task is not exactly defined and the interaction with the external environment is somehow uncertain. Examples of such tasks are walking, running, and manipulation (including grasping and catching). Humans are far better in those tasks than robots; thus a nine-month old baby can successfully reach out for an unknown object and grasp it, whereas robots cannot manipulate an object if it is not in a specified position, and an accurate three dimensional model of it is not available. More surprisingly human limbs are kinematically redundant; the same is rarely true for the artificial manipulators used in robotics. The inverse kinematics and dynamics problems in this case would be even more challenging.

This difference between biology and robotics resides in part on the structural differences of the mechanical actuators usually employed in robots as opposed to our muscles and their physical properties. Electric motors are almost ideal force generators, meaning that they can generate a required force independently of their position and the effect of the external environment. Several experiments have been conducted to study the mechanical properties of skeletal muscles and the mechanism used by the nervous system for controlling movement. A detailed description of the internal structure of muscles and the cellular mechanisms by which neural signals are converted into mechanical forces is beyond the scope of this discussion (a description can be found in (Ghez, 1991). Taken as a whole, the most evident property of muscles is that they exhibit a spring-like behavior; the contraction force of a muscle depends not only on the level of activation of its afferent motor neurons, but also on its length (Ghez, 1991). A spring is a mechanical device that absorbs energy when stretched and responds to the increase of length with a restoring force. This force represents the system tendency to move toward a state with minimum energy which corresponds to its *resting length*. Zero force is produced when this minimum length is exceeded; beyond this limit the force increases linearly (Hooke's law). Mathematically this behavior can be so described:

$$\begin{cases} F = k\left(l - l_0\right) & \text{if } l > l_0 \\ F = 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The ratio between the amount of force developed as a result of a given elongation defines the spring stiffness:

$$k = \frac{dF}{dl} \tag{3.2}$$

The force-length characteristic of a muscle is however more complicated. If the length of the muscle is forcedly changed – for instance by means of a motor connected to its extremity – it is possible to measure the resulting restoring force (its *tension*) and to derive the muscle force-length characteristic. The result in Figure 3.1 (left) shows that the slope of the characteristic is not constant and that the stiffness of the muscle varies with its length. Within a certain range, however, the constant stiffness spring law is a good approximation. The central nervous system can actively change the length-tension curve, in particular the stiffness of the muscle has been shown to vary with neural activation Figure 3.1 (right).

The resting length of the muscle defines its equilibrium point that is an intrinsically stable state toward which the system is spontaneously driven if an external disturbance (force) is applied. Given a certain force the resulting displacement depends on the stiffness of the muscle. As soon as the external force is removed the system is free to return to a configuration with minimum energy (it

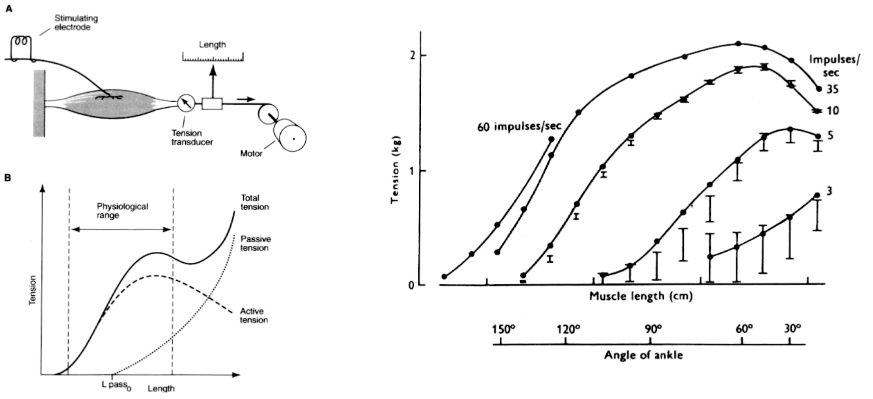


Figure 3.1. Muscle tension-length characteristic. Left plot: muscle stiffness varies according to neural activity (adapted from (Ghez, 1991)). Right plot: Length-tension curves measured in the cat's soleus muscle at different activation rates. The initial part of the curves is linear, stiffness increases with the activation (adapted from (Mussa-Ivaldi and Bizzi, 1993)).

can be easily shown that the equilibrium point corresponds to a minimum in the potential energy of the spring). This mechanism would be used by the CNS to control movement. The activation level of each muscle would be actively varied to shift its equilibrium point and produce a torque suitable to drive the attached limbs according to the desired trajectory (*equilibrium point hypothesis*, *(Bizzi et al., 1991; Hogan, 1985; Mussa-Ivaldi and Bizzi, 1993)*).

The sequence of equilibrium positions does not correspond to the actual trajectory followed by the limb and constitutes the so called *virtual trajectory*. During motion the intrinsic elasticity of the system would be responsible to generate restoring force to compensate for errors in the trajectory and external disturbances by the environment. The nice result of this idea is that it avoids the necessity to explicitly solve the inverse dynamics problem as the limb stiffness is responsible for generating the force required to compensate for the effect of inertia, viscosity and friction. Small errors in the movement could be compensated for by actively increasing the stiffness of the structure.

However, this last point has been questioned. In fact it was shown that the level of stiffness that would be required to produce accurate movements is not in accordance with experimental results on humans (Katayama and Kawato, 1993). Apparently the nervous system would have to take into account discrepancies between desired and actual motion in the computation of the virtual trajectory. In this case the computational advantages of the equilibrium hypothesis seem to vanish, because the calculation of the virtual trajectory would implicitly need to solve the inverse kinematics problem (Katayama and Kawato, 1993). In any case, the equilibrium hypothesis is important because it puts control of movement and posture under the same perspective. Besides, muscle spring-like properties are still considered to simplify the problem solved by the CNS in controlling the limbs.

Usually position control is used to perform link motion in robotics. This is very effective when the interaction with the environment is limited or controlled; the torques exerted by the motors are computed to produce a desired motion of the end-effector. Several control strategies can be employed, from very simple (PID) to more complicated (adaptive control, computed torque control (Fu et al., 1987)); to different extent all these strategies are effective when the robot is free to move. If physical interaction with an external surface or an unexpected object occurs, it may be impossible to apply the desired motion. In trying to compensate for the error in the motor trajectory the controller would increase the motor torque with the risk to break either the robot or the object (or both). For this reason recently, and particularly for applications where robots operate in the proximity of humans, the concept of "intrinsic safety" was introduced. This refers to robotic systems where unpredictable failures will produce only limited damage (if any). One possibility is to add some level of compliance in order for the robot to adapt to unpredicted obstacles. This can be achieved in the following ways:

- active control
- passive compliance

In the first case, the controller is designed thus to actively simulate a compliant behavior. This is usually obtained by controlling the force applied by the end-effector instead of its position; examples are stiffness/impedance control and pure force control. In the second case, the manipulator is position controlled and material with elastic properties is used to obtain the required level of compliance. This solution includes covering the end-effector with soft substance – which is usually the part of the manipulator that more often comes into contact with the environment. – or using conventional springs in series with the motors, as in the series elastic actuators (Pratt and Williamson, 1995; Robinson, 2000). An alternative way to obtain a certain degree of low stiffness is also to reduce the gains of the controller so that it responds with a relatively small force to a position error. A similar control was actually implemented for the motion of the PUMA arm mounted in the robot (see Section 5.1).

The basic idea underlying active force control is to compensate the inertia of each link and to simulate a damped spring-mass system. Although it is in principle possible to simulate any kind of spring and to vary its stiffness, this approach is technically difficult because the controller needs to compute first and second derivative of the state of the system.

A possible solution to the problem is to employ actuators specifically designed to be physically compliant. Hydraulic actuators are essentially position controlled because of the fluid (usually oil) high stiffness and difficult backdrivability; pneumatic actuators may have lower stiffness, but exhibits a very slow bandwidth response. Low impedance control with DC motors is difficult because of inertia and friction added by the reduction boxes.

Another approach is to add a linear elastic component to traditional DC or hydraulic actuators (series elastic actuators see (Pratt and Williamson, 1995; Robinson, 2000)). This solution, for example, has been successful for the control of walking robots (Robinson et al., 1999) and humanoids robot in general (Brooks et al., 1999). A linear spring is placed in series with the electric motor to obtain a low stiffness actuator. Force feedback is provided by measuring the spring displacement (according to Hooke's law (3.1)). A similar solution was adopted in the finger joints of the hand (see Section 2.1). Although more suited for force control than standard actuators, series elastic actuators have constant stiffness determined by the embedded elastic element and are not good models of human muscles.

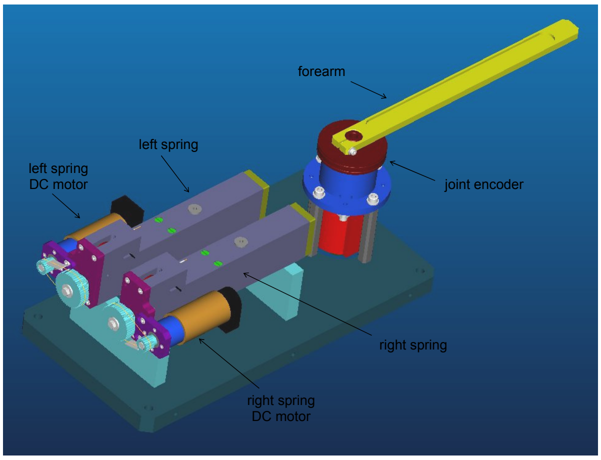


Figure 3.2. CAD model of the realized prototype. See text for a detailed description.

3.2. The mechanical prototype

Motivated by these considerations we have designed and realized a single joint elastic actuator with variable stiffness. A cad model of the prototype is represented in Figure 3.2. Motion is produced by two elastic elements (two identical helical springs) connected in a push-pull configuration to a rotary joint by means of metal tendons. Two electrical DC motors control the stiffness of the springs by changing the number of coils that are free to move (hereafter we will refer to them as *active coils*); to obtain this, the motor shafts are shaped as screws with a thread equal to the one of the springs. When an actuator rotates, it screws in (or out) the spring reducing (or increasing) the number of coils that are effectively pulling the tendon connected to the "free end" of the spring. In this way the elastic property of the spring changes as a function of the rotational position of the actuator. It is worth noting that this actuator must be employed for a rotational joint, as the mechanism only exerts a "pulling" force and that each joint has to be controlled by a pair of actuators (similarly to how muscles works in humans and other animals). As far as sensors are concerned, optic encoders provide position feedback of the two electrical motors as well as the actuator joint, whilst strain gauges measure the tension of the two cables to provide force feedback (Figure 3.3).

3.3. Mathematical model

Let us consider a spring made of a single coil. Starting from Hooke's law it is possible to derive the force that is exerted as a result of stretch:

$$F = -\frac{1}{c}\left(x - x_{00}\right) \tag{3.3}$$

where c is the compliance of a single coil, x_{00} its resting length.

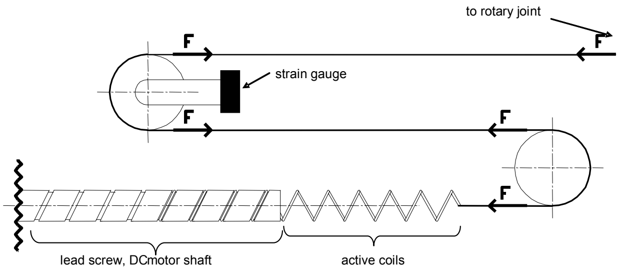


Figure 3.3. Detail of a single elastic actuator. A lead screw controlled by a DC motor varies the number of coils of the springs. The spring stiffness changes as the motor rotates. A strain gauge provides force feedback (F) whereas digital encoders (not shown) measure the position of the motor shaft as well as the position of the rotary joint.

Let now be n the number of coils of the spring. Imagine the spring as consisting of n single coil elastic elements (springs) in series, with a compliance of c and zero resting length each. If we now suppose to apply a force F to the spring, at equilibrium each single coil element would exert a force F to its neighbor; the displacement of each element would then be $d = F \cdot c$, and the overall displacement $X - X_{00} = n\left(x - x_{00}\right)$. We can now rewrite Hooke's law to obtain the stiffness of the new n-turns spring as follows:

$$K = \frac{\Delta F}{\Delta X} = \frac{d / c}{nd} = \frac{1}{n \cdot c} \tag{3.4}$$

From this equation it is clear that by changing the number of active turns, it is possible to regulate the stiffness of the spring.

Let us now connect the two elastic actuators together (Figure 3.4). If the tension of the cable is not zero, the positions of the springs are constrained by the length of the cable itself ($X_1 = X_2 = X_m / 2$). The net torque that acts on the joint is:

$$T = \left(F_1 - F_2\right) \cdot r \tag{3.5}$$

Putting together the force law of each spring and the cable constraint leads to the following equation:

$$T = -\widehat{K}\left[X - \widehat{X}_0\right] \tag{3.6}$$

Where:

$$\begin{aligned} \widehat{K}\left(n_1, n_2\right) &= \frac{1}{c \cdot n_1} + \frac{1}{c \cdot n_2} \\ \widehat{X}_0\left(n_1, n_2\right) &= X_m \frac{n_1}{n_1 + n_2} \end{aligned} \tag{3.7}$$

Equation (3.6) represents Hooke's law for a single spring, where both stiffness and resting length depend on the controlled input (n_1, n_2) (see Figure 3.4). By controlling the DC motors it is possible to act on the input (n_1, n_2) to change the spring resting length and stiffness.

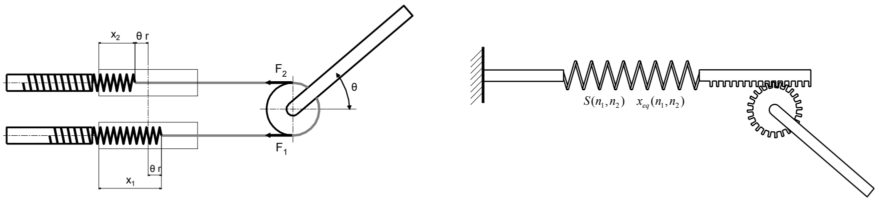


Figure 3.4. Left: two actuators linked together. Tendons connect two springs to the joint in push-pull configuration. Right: the effect of the two springs is equivalent to a single spring whose stiffness and resting length depends on the number of active coils.

3.4. Experiments with a single actuator

A first experiment was conducted on a single spring. The number of coils was varied to sample an interval between 5 and 10. For each of these values the position of the joint was passively varied in a range between +/- 50 degrees. The output of the strain gauge was recorded to estimate the force-length characteristics; a line was fitted to the characteristic when force values were above zero and its slope taken as an estimation of the stiffness.

The plot reported in (Figure 3.5) shows the results of this experiment. The force-length curve behaves as theoretically predicted showing that the input of the system controls the stiffness of the spring. The stiffness decrease as the number of coils increases according to the inverse proportional law of equation (3.4).

3.5. Experiments with coupled actuators

A similar experiment was conducted with the two actuators linked together. The link was passively moved to sample different positions ranging from -50 to +50 degrees. We recorded the net restoring torque exerted by the two actuators; in this

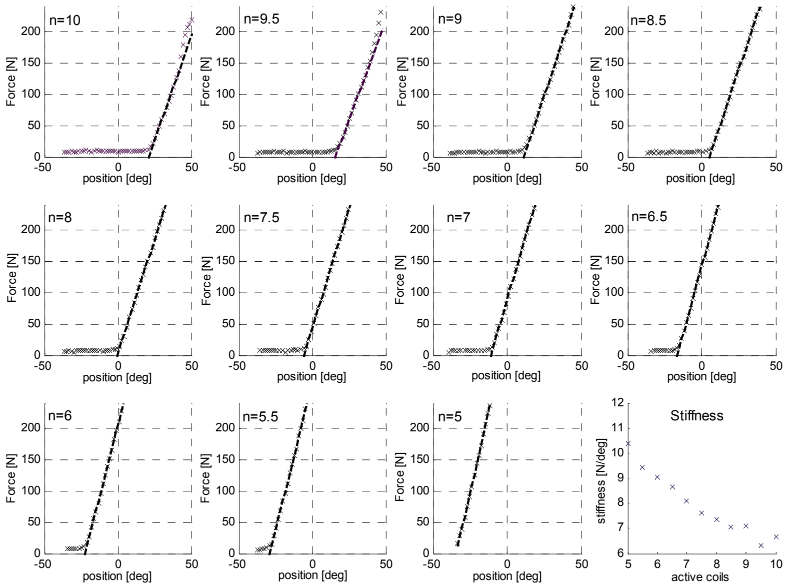


Figure 3.5. Force-length curves of a single actuator. The number of active coils varies from 10 (top-left) to 5 (bottom-right). In each plot a line was fitted on the data for F>0 and its angular coefficient taken as an estimation of the stiffness. These measures are reported in the last plot which reports the variation of the stiffness with respect to the number of active coils.

case we varied the number of active turns of both springs (range from 5 to 9) in order to keep a symmetric configuration; this means that in all cases the equilibrium point of the system (restoring force = 0) was the midpoint (0 degrees). This experiment tests the spring-like properties of the actuator as a whole (Figure 3.4). The results reported in Figure 3.6 show that the slope of the curve force-displacement increases as the number of coils decreases.

However, it is not possible to define a single value for the stiffness; the restoring force is in fact the net result of the forces exerted by the two springs. The central part of the curve results from the coupled action of the two springs unless one is slack. In this case only a single spring is actually exerting force and therefore contributing to its variation – the stiffness. Mathematically this effect is represented in equation (3.1); in the experiments it is well visible by the sudden changes in the slopes.

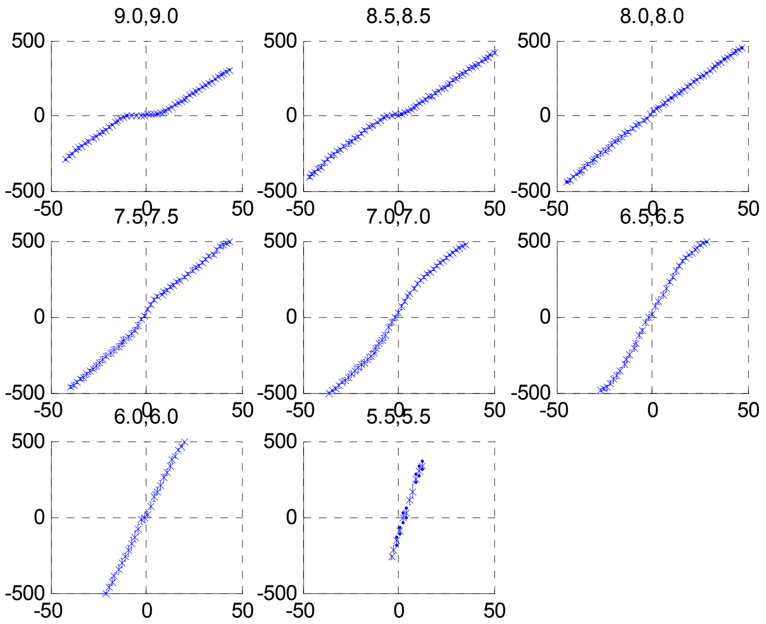


Figure 3.6. Force-length characteristics for the linked actuators. Abscissa represents the position of the link (roughly in the range +/- 50 degrees), ordinate reports the net restoring force acting on the link (+/- 500 N). Plots are reported for different values of active coils from 9 (top left) to 5.5 (bottom right). The stiffness follows an inverse proportional law with respect to the number of coils of the springs.

3.6. Open loop control

The actuator was also tested in a simple open-loop configuration. Open-loop here refers to the fact that we did not use the feedback information provided by the strain gauges and the encoder on the joint. The input to the system (n_1, n_2) was varied by controlling the position of the motor shafts with a PID controller. Figure 3.7 describes the control schema in more details.

3.6.1. The force-displacement plane

The net force of the actuators is described by three variables: the number of active coils of the springs (n_1 and n_2 both ranging from 1 to 15) and the position of the joint.

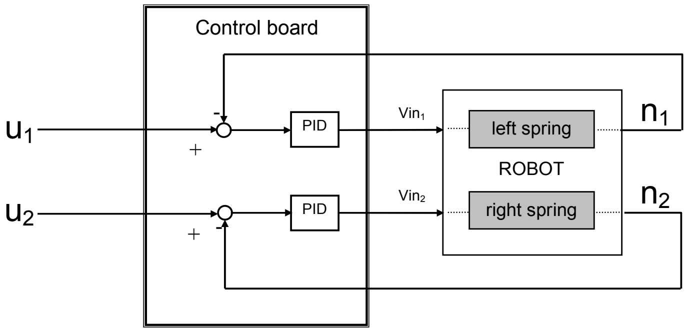


Figure 3.7. Open-loop control schema. The control board computes the PID control law to perform a desired motion. The input to the system is the number of active coils (u1, u2).

Let us draw the force-length characteristics of both springs on the same plot. The system equilibrium point corresponds to the points where the forces applied by the two springs are equal; it can be computed as the point in space where the curves intersect. If as a result of an external force the system is perturbed, it is possible to compute the resulting restoring force as the distance between the characteristics; it is worth noting that the angle made by the two lines implicitly defines the stiffness of the system (Figure 3.8).

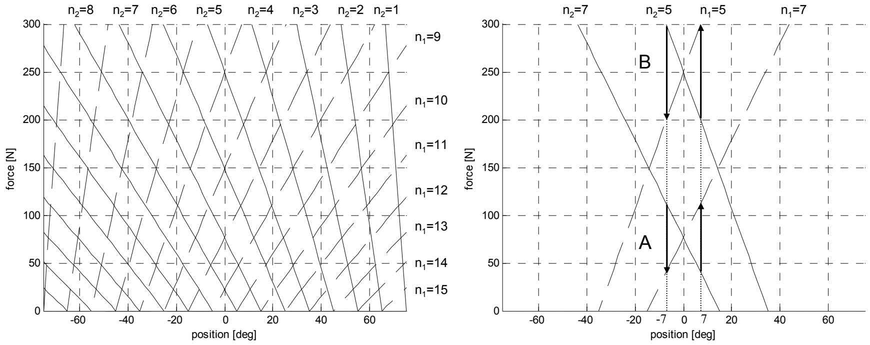


Figure 3.8. Left: force-displacement plane. Dashed lines represent force-length characteristics of the left springs. Solid lines: right spring. Number of turns range from 1 to 15 (n1 and n2). The points were the lines intersect correspond to the equilibrium points for the joint; the angle between left and right spring characteristics implicitly define the stiffness. Right: an external force moves the system from the equilibrium position of +/- 7 degrees. Two configurations are depicted, low stiffness (A) and high stiffness (B). The restoring force exerted by the two springs together can be graphically computed by measuring the distance between solid and dashed lines. It is easy to verify that in (A) the restoring force is stronger than in (B), although the displacement is the same.

When the springs are stretched or compressed they release or store energy. The amount of stored energy is:

$$E = \frac{1}{2} K \cdot X^2 = \frac{F^2}{2K} \tag{3.8}$$

To take into account the variation of stiffness equation (3.4), equation (3.8) can be rewritten as follows:

$$E = \frac{1}{2} F^2 n \cdot c \tag{3.9}$$

At equilibrium the springs exert opposite and equal forces. The total potential energy stored in the system is hence the sum of the potential energies of the two springs:

$$E_T = \frac{1}{2} F^2 \left(n_1 + n_2 \right) c \tag{3.10}$$

The area corresponding to higher F – top region in the force-displacement plane of Figure 3.8 – represents configuration with higher potential energy.

This plane was introduced here as a means of representing possible trajectories in the position-stiffness plane. Figure 3.9 represents three exemplar trajectories. The initial position of the joint in all three cases is equal to -20 degrees, whilst the final

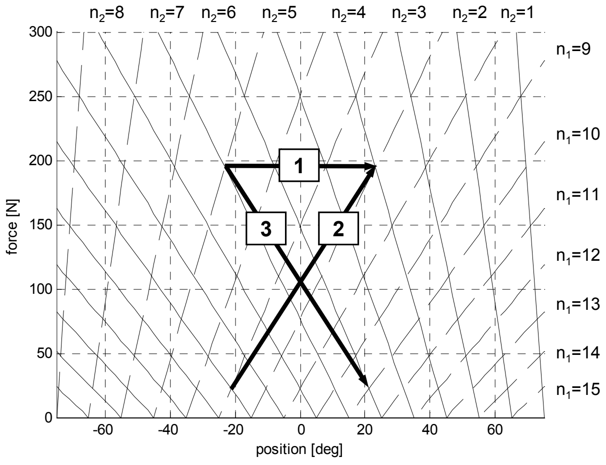


Figure 3.9. Three exemplar trajectories. In all three cases the system moves from an initial position of -20 degrees to a final position of 20 degrees. As far as the stiffness of the joint is concerned, the three trajectories are quite different. In (1) the stiffness is kept constant at a relatively high value; in (2) the stiffness is initially low and gets constantly increased whereas in (3) the opposite occurs.

position corresponds to +20 degrees. In trajectory 1 the system starts with a relatively high stiffness and moves to the final position in such a way as to maintain constant stiffness. According to equation (3.10) the potential energy does not change. Conversely in trajectory 2 and 3 the stiffness increases and decreases respectively. Besides, it is worth noting that in trajectory 2 the springs increase their potential energy whereas in trajectory 3 the potential energy stored at the beginning of the motion is released.

In this experiment we investigate the possibility to exploit the energy stored within the springs to produce motion. For each of the above trajectories we recorded the position of the joint, the number of active turns and the electric current absorbed by the motors. The latter were acquired by dedicated output port on the amplifiers. The time course of all these variables is reported in Figure 3.10, Figure 3.11 and Figure 3.12, in the case of trajectory 1, 2 and 3 respectively. Unfortunately, from the results it was not possible to draw precise conclusions. Apparently when mechanical energy is released by the springs the motors absorb less current. This is more evident where the system moves from a configuration with high stiffness to one with low stiffness and vice-versa. However, frictions vary remarkably in the different conditions; this can be noticed by observing that at the end of the motion the motors absorb a considerable amount of electric current which correspond to the torque they apply to compensate friction (third plot in Figure 3.10 and Figure 3.11). Hence, it was not possible to separate mechanical energy spent compensating frictions from the one required to produce motion. As a final note Figure 3.13 compares trajectory 2 and trajectory 3; notice that at high stiffness there is a lower error at the end of motion.

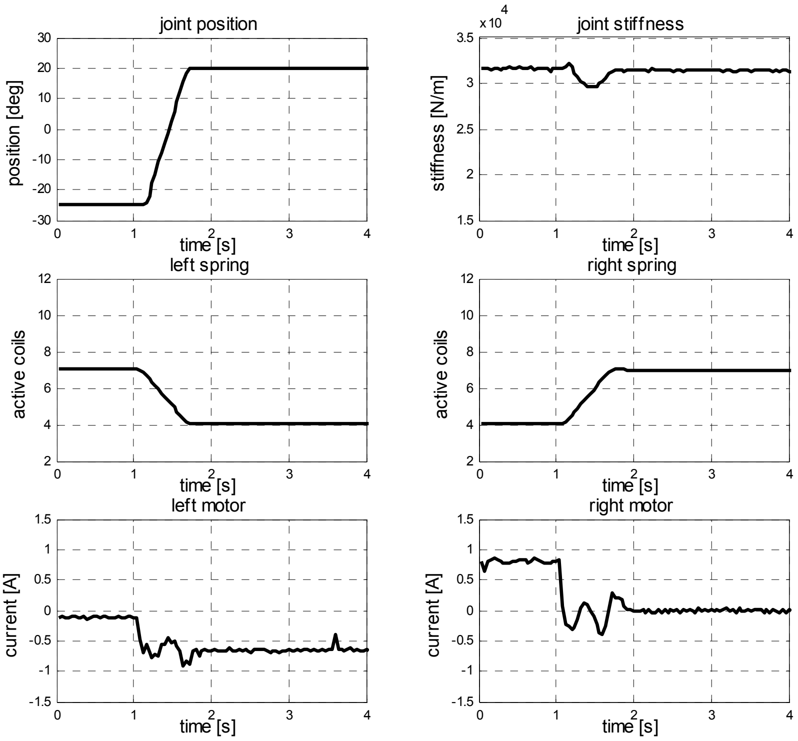


Figure 3.10. Trajectory 1. The system moves from a position of about -20° to +20°. The stiffness in this case is maintained constant at a relatively high value. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors. Note that there is a residual current whenever the controller cannot reduce the error to zero due to the friction.

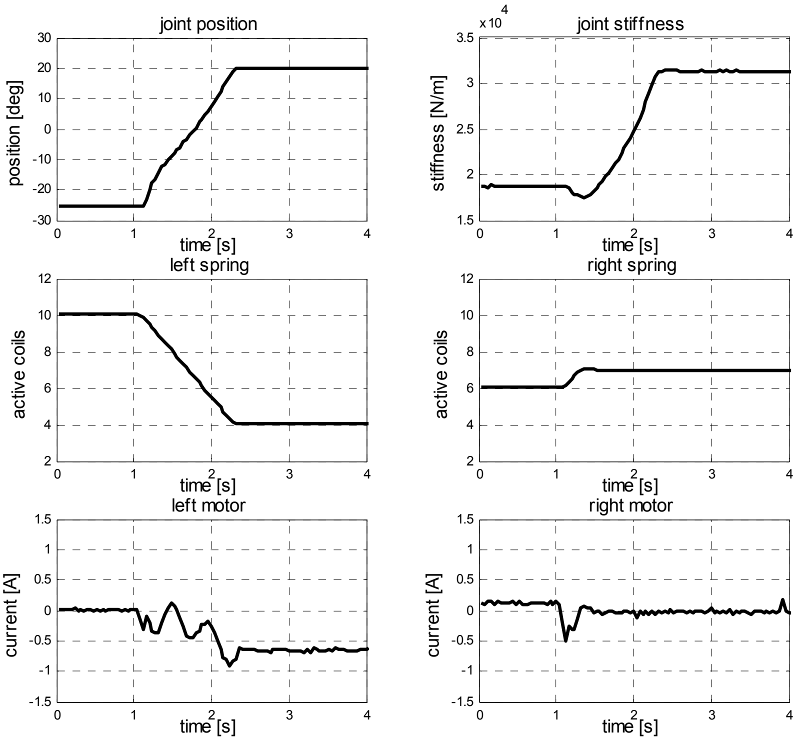


Figure 3.11. Trajectory 2. The system moves from a position of about -20° to +20° while increasing the stiffness. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors. Note that there is a residual current whenever the controller cannot reduce the error to zero due to the friction.

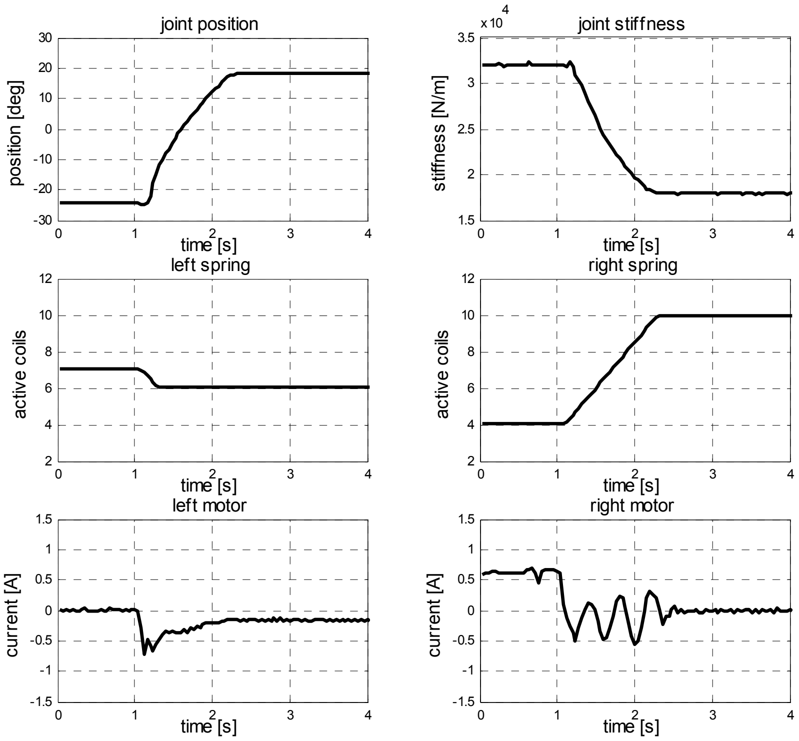


Figure 3.12. Trajectory 3. The system moves from a position of about -20 ° to +20 °while reducing the stiffness. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors.

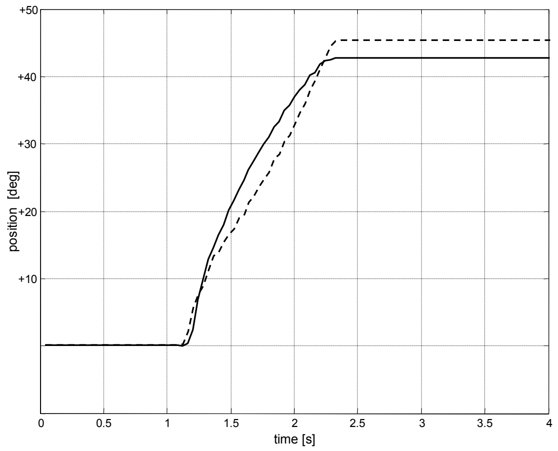


Figure 3.13. Comparison between trajectory 2 (dashed line) and trajectory 3 (solid line) (see also Figure 3.9). The plot represents the time course of the position; the curves are shifted to make zero the initial position and facilitate comparison. Note that trajectory 2 is more accurate because the higher stiffness contributes to reduce the final error due to the weight of the link (the actuator is mounted vertically, the weight opposing the motion).

3.7. Conclusions

In this chapter we have proposed a model for a mechanical actuator mimicking the elastic properties of human muscles. A physical prototype of the system was realized to test the model in static and dynamic conditions. Accordingly, two experiments were reported. In the static conditions the force-length characteristic of the actuator was derived by imposing an external displacement and measuring the net restoring force. In a dynamic experiment the input of the system was varied to produce different position and stiffness trajectories. The results show that i) by varying the position of the DC motors it is possible to control position and stiffness of the actuator and ii) the joint behaves differently if the stiffness is varied.

The ability of biological systems to regulate the mechanical compliance of their limbs represents perhaps the most remarkable difference with respect to traditional robots. It has been proposed that the central nervous system exploits the intrinsic elasticity of muscles to simplify the problem of motion control; in particular it provides automatic restoring forces to external disturbances or erratic behavior. Furthermore, intrinsic elasticity reduces shock due to unexpected collision and increases operational safety.

In robotics actuators with real springs to produce a similar mechanism have been used. Kolacinski and Quinn (Kolacinski and Quinn, 1998) realized a prototype of a variable stiffness elastic actuator. They used a scissor-like mechanism attached to a conventional spring; the end-points of the spring are free to slide along the scissor links as the result of the external force. The angle between the links varies the rate of deformation of the spring, that is the stiffness as seen from the load. However the prototype had limited range of motion and was tested in static conditions; besides the mechanical architecture of the device limits to some extent the possibility to obtain small and compact actuators.

The last experiment tried to investigate the possibility to exploit the internal energy stored within the springs to produce motion. Two conditions where tested: increasing the internal energy stored within each springs and decreasing it. Unfortunately from this experiment it was not possible to derive any clear conclusion, because the amount of friction between the springs and the screws varied significantly in the conditions we tested. Other limitations of the proposed solutions were the range of stiffness and the size of the DC motors. The former can be improved by changing the parameters of the springs (compliance of each spring single coil and their number of coils). In the latter case, it seems that most of the torque produced by the DC motors is spent to compensate friction between screw and spring, especially at high force (high stiffness). A possible solution could be to treat the springs in such a way to reduce the friction. In any case the basic idea (to change the number of active coils of the spring to modulate its stiffness) seems worth pursuing, although further research is required before this actuator can actually be used on a real robot.

Chapter 4

Eye movements

The first contact with the world takes place through vision. This chapter is concerned with the robot's visual system and describes the control strategies to control the eyes to gaze objects in space. Most of what will be described was re-implemented from previous work on the same or other robots (references to the original work are provided in each section). A short overview is reported here to give the reader the basis to understand the next chapters.

4.1. Retina-like visual system

Photoreceptors within the human retina exhibit a space-variant arrangement. Cones – which are responsible for visual perception in the light – have higher density at the centre of the visual field (the fovea) and are sparser in the periphery. The size of their receptive fields changes accordingly (see Figure 4.1 (a)). This layout allows for a central part of the visual field to be suited to carry out precise tasks, while maintaining a wide field of view. The photoreceptors in the periphery are more sensitive to changes in illumination, hence appropriate for motion detection. In this case adaptation solved the problem of minimizing the number of photoreceptor in order to have both high visual acuity and wide field of view. It is intuitive to understand the advantages of such architecture; it can be proved that the number of neural fibers required to transport the signals all the way from the retinas to the visual cortex is of orders of magnitude smaller. Needless to say this can be a

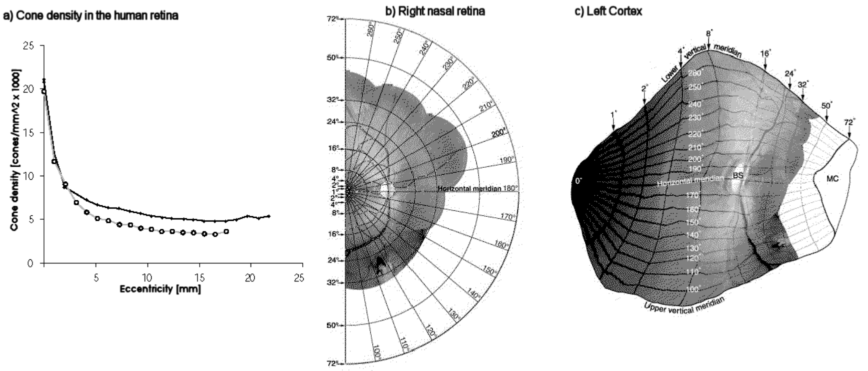


Figure 4.1. Cones density in the human retina decreases quickly as we approach the periphery (a). Dark gray is temporal retina; light gray is nasal retina (adapted from (Packer and Williams, 2003)). Retinotopic map in the striate cortex of the squirrel monkey (c) and (d). The mapping is illustrated by the system of rings and rays superimposed on the retina and plotted as they warp on the cortex; rays and circles map to horizontal and vertical straight lines. Notice also how most of the cortex is devoted to the central part of the retina; in particular half of the cortex represents rings from 0 to 8 degrees (adapted from (Adams and Horton, 2002)). Compare these pictures with Figure 4.2 and Figure 4.3.

significant advantage from a computational point of view. Indeed, studies on primates have revealed that there is a specific geometric layout in the way receptors are wired from the retina onto the cortex (Figure 4.1(b) and (c)).

Images with a space variant geometry have been used in robotics where real-time performance is important (Bernardino, 2004). Of all the possible implementations the log-polar geometry (see the next section for a mathematical formulation) better resembles the organization of the receptors in the human retina (Sandini and Tagliasco, 1980; Schwartz, 1980).

One way to obtain space-variant images is that of subsampling traditional rectangular images. Besides requiring specific hardware (a dedicated computer or a DSP) this solution looses part of the advantages, since it still requires a high bandwidth connection to transfer the images from the sensor to the device performing the sampling.

In the past years at LIRA-Lab different versions of a silicon C-MOS sensor were realized to implement the log-polar sampling (the Giotto sensor, for a review see (Sandini and Metta, 2003; Sandini et al., 2000)). The size of the electronics driving the chips is still big if compared to the one of the standard commercial cameras; for this reason the current setup mounts standard cameras and performs the log-polar mapping in software on a dedicated PC; the resulting log-polar images have exactly the same structure of the Giotto sensor. The details of the mapping are given in the next section.

4.1.1. Some maths

Photoreceptors in the Giotto sensor are arranged in 152 concentric rings of 252 pixels each. The size of the receptors varies by following a logarithmic rule. Mathematically the mapping between the polar coordinates and the log-polar plane (often referred to as the *cortical plane*) can be written as:

$$\begin{cases} \eta = q \cdot \vartheta \\ \xi = \log_a \dfrac{\rho}{\rho_0} & \text{if } \rho > \rho_0 \end{cases} \tag{4.1}$$

where $1/q$ is the minimum angular resolution, ρ_0 is the radius of the innermost circle. The preceding equation is related to the standard x-y coordinate system by the following well known equations:

$$\begin{cases} x = \rho \cos \vartheta \\ y = \rho \sin \vartheta \end{cases} \tag{4.2}$$

As we approach the central part of the retina (the fovea), the size of the receptors decreases toward zero, until it reaches a physical limit and we are forced to change the layout of the chip. In practice this physical limit occurs below ρ_0 which is the minimum radius of the circle where equation (4.1) is valid. Within this region (the fovea), the size of the receptors does not change. Their number along a given circle is reduced linearly (for more details: (Berton, 2003)).

Graphically the mapping is represented in Figure 4.2. The original image is uniformly sampled along concentric circles; the samples from each circle (at

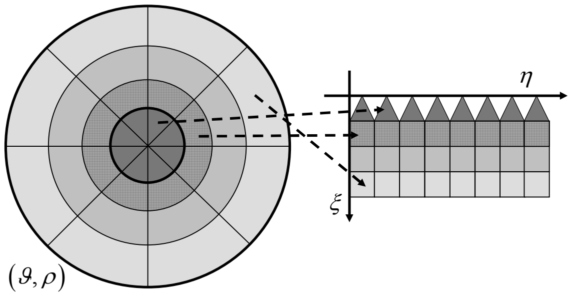


Figure 4.2. A simplified explanation of the log-polar mapping. The image is divided in concentric circles which are uniformly sampled and arranged along the rows of the logpolar image. The outermost and innermost circles are placed in the last and first rows respectively (different shades of gray are used for different radii). The darkest region at the center is the fovea, which is divided in triangles and reported in the cortical plane (Berton, 2003).

constant radius) are arranged along the rows in the cortical plane. The innermost and outermost circles correspond to the first and last row respectively. The pixels in the fovea are arranged in the log-polar domain in contiguous triangles. Log-polar images are distorted version of the original; the most evident effects are that circles in the Cartesian image become straight lines in the log-polar plane and that almost half the number of the its pixels is employed to represent the central part of the scene. This last point illustrates the computational advantage of this representation (Figure 4.3 depicts the log-polar sampling of a real image).

4.2. Gaze control

In an active system with space-variant vision eye movements are essential to build a coherent percept of the world. They can be voluntary as well as reflexive. Reflexive movements allow us to stabilize the visual world when we walk, run or drive. These compensatory eye movements take care of minimizing the motion in the visual scene we perceive as our body actively moves about (like in walking) or gets passively stirred (as when we drive our car or we are on a train).

Voluntary mechanisms for moving gaze to explore the visual world are also very important in animals and became a requirement after development of fovea took over panoramic vision in more evolved species. In the latter case the entirety of our perception results from the integration between consequent images acquired as exploratory movements are performed in the environment. Together with foveal

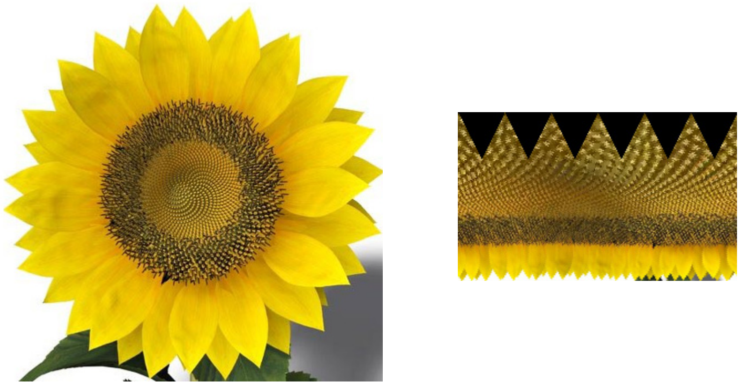


Figure 4.3. An example of log-polar mapping. Left: the original image. Right: the resulting log-polar image in the cortical plane. The particular image stresses the salient characteristics of the transformation: the circular arrangement of the petals is straightened in the log-polar domain, more than half of the pixels in the cortical plane is used to represent the central part of the flower.

vision gaze provides an automatic mechanism to direct our attention toward regions of interest in the scene – the latter may be for example an auditory, visual or tactile sensation. Perhaps more surprisingly, gaze plays a very important role in action planning as well. Experiments have shown that the direction of gaze guides locomotion and anticipates the trajectory the body is going to follow (Berthoz, 2000).

4.3. Visual stabilization

Visual stabilization is the result of at least two reflexes involving inputs from the vestibular system and visual information. They are called vestibulo-ocular reflex (VOR) and opto-kinetic reflex (OKR) respectively and cooperate to reducing the slip of the images on the retina (Berthoz, 2000; Carpenter, 1988). VOR is very fast owing to the fact that vestibular information is simple and does not need to travel a long neural pathway to be processed. A simple experiment can be made to realize the efficiency of the VOR. Raise a hand in front of your eyes and move it in an oscillatory movement from left to right. Try now to track the index finger with the eyes and to increase the speed of the hand. You will see that as soon as the speed exceeds a certain value the eyes tend to jump back and forth to the extreme positions of the hand. This is because the brain is no longer able to track the index finger efficiently and switches from smooth pursuit to saccadic control. Now, try to keep the hand still and move the head instead. From a kinematic point of view the problem is exactly the same, but it turns out to be much easier for the brain. This time, in fact, it takes advantage of information from the vestibular system. No matter how fast we move the head, our eyes can easily track the index finger.

OKR has a longer latency because requires the computation of optic-flow (i.e. the displacement of the image that projects on the retina) in the visual field. Accordingly VOR is suitable for fast movements whereas OKR is preferable for slower one. For better stabilization both mechanisms are actually integrated in the brain (see for example (Carpenter, 1988)).

Past works have addressed the implementation of such mechanisms on Babybot (Panerai et al., 2000). Although the robot in the experiments described here was placed on a table, visual stabilization – especially the VOR – was still employed during tracking to facilitate the coordination between the eyes and the neck (Metta, 2000).

4.4. Voluntary eye control

Smooth pursuit and saccades are the mechanisms used by humans for voluntary control of gaze. In the first case the eyes move relatively slowly (< 50 deg/s) to maintain the fixation on a moving target. When the brain wants to shift the attention from a point in space to another, or needs to catch up with a fast target, it performs a saccade. Saccades are ballistic, open loop, movements that, owing to their high speed (roughly, a saccade can be as fast as 1000 deg/s), increase the efficiency of the oculomotor system.

Both strategies were included in the robot. Smooth tracking has been implemented with a closed-loop control exploiting positional information. The Jacobian of the transformation between sensory and motor space – required to convert the positional feedback into appropriate velocity commands – was learnt by the robot at the very first stage of the developmental process. Conversely saccades require a transformation between sensory information and motor command. An inverse (feedforward) model is needed to compute the motor command required to fixate a target whose position is available in a given sensory space. Insofar we have used the term sensory space, because gaze can be oriented toward any sensory source (to be of auditory, tactile or visual nature). The superior colliculus is the area in the brain where the integration of the information between different sensory modalities is thought to occur (Meredith and Stein, 1986). Orienting behavior in the robot is indeed driven by visual and auditory cues (Metta, 2000; Natale et al., 2002a).

4.5. Vergence control

We have just discussed the importance of the so called orienting behavior controlling the direction of gaze, that is its *version*. In binocular systems to determine the position of an object in space it is required to have both cameras fixating it. This is what is commonly referred to as *vergence control*. In principle if both cameras are tracking exactly the same target there is no need for a separate controller. As this is rarely the case, disparity information is usually employed to improve vergence control and depth estimation. The solution adopted on Babybot uses cross-correlation to compute a measure of the similarity between the two images (the *disparity-index*). At every frame images are shifted and the disparity-index computed; the shift corresponding to the minimum disparity drives a closed-loop control to verge the eyes toward the same point (Manzotti et al., 2001). To increase accuracy, in our latest implementation the disparity-index is computed on the central part of the images (a square of 128x128 pixels centered in the fovea)

Control of version and vergence need to be integrated in a meaningful way. We implemented the following straightforward solution. The tracking algorithms (the tracker described in the next section or the hand tracker described in Section 5.2) run on the images acquired by the left camera; both eyes are controlled to track this reference signal. Stereo fusion is achieved by adding the output of the vergence algorithm to the motor command of the right eye. To put it in other words, there is a

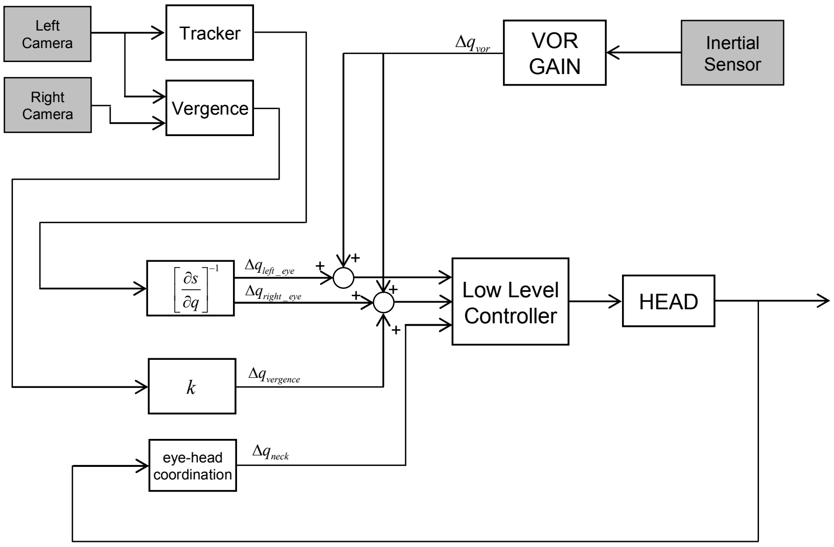

Left Camera
Right Camera
Tracker
Vergence
Δq_{vor}
VOR GAIN
Inertial Sensor
$\left[\frac{\partial s}{\partial q}\right]^{-1}$
Δq_{left_eye}
Δq_{right_eye}
+
+
+
+
+
Low Level Controller
HEAD
k
$\Delta q_{vergence}$
eye-head coordination
Δq_{neck}

Figure 4.4. Head control schema. Images from the left camera are sent to the tracker which extracts the position of the target in image coordinates; by means of the inverse Jacobian this information is converted into motor commands for the eyes. The block indicated by "vergence" computes the disparity index; the latter is then multiplied by a constant proportional gain and added to the right eye motor command. Information from the inertial sensor is used to compute the VOR component. VOR, vergence and version are summed together and issued to the low-level controller which computes the torque to drive the motor. From the encoder feedback the d.o.f. of the neck are coordinated with the eyes as described in Section 4.6. The blocks realizing saccades are not shown.

"dominant" eye where the tracking is computed and a "slave" eye which goes behind it trying to minimize the disparity. A similar strategy seems to be employed by the brain to combine version and vergence. The separation, however, is fuzzier and left and right eyes contribute, although to a different extent, to both the components (more details are reported in (Carpenter, 1988) , Section 5.3). Figure 4.4 reports a schematic diagram of the head control loop, Figure 4.5 and Figure 4.6 show trajectories of the robot performing tracking.

4.6. Eye-head coordination

Visual information is naturally represented within an eye centric reference frame. In humans, however, head-centric reference frames are also used to plan orienting behavior. On the one hand control of the neck extends the region of space the robot can attend to, however, on the other hand, it adds redundancy to the system. This means that the same 3D fixation point can be attained by using different joint configurations (in the case of the Babybot for instance the head consists of 5 d.o.f.).

A possible solution is to define a strategy to coordinate the eyes with the head. Thus, the additional pan of the neck is controlled in such a way as to maintain a "comfortable" posture where the eyes are in a symmetrical vergence configuration. The control schema that implements such strategy is a feedback loop which acts to minimize the difference between the eyes joint angles. The additional tilt of the neck moves to minimize the vertical position of the eyes with respect to the head. In other words, the neck is controlled to follow the eyes so that they are usually centered with respect to the head. This solution is simple but effective because it maintains the eyes far from the limits and thus ready to perform new movements.

4.7. Attentional system

In the previous sections we have described how the motor system can generate appropriate motor commands to orient gaze toward a given sensory source. The robot can build and continuously update a saliency map containing regions in space where interesting events occur. Of course there may be more that one region of interest. For example a motion algorithm detects someone entering the room while a color segmentation algorithm is selecting a bright red toy and the auditory system perceives a noise because an out of sight object has just fell off the table. To obtain a meaningful behavior the robot should be capable of inhibiting less interesting stimuli and direct its attention toward the more significant ones.

If on the one hand the robot needs to shift attention from a location to another, on the other hand gaze has to be maintained – at least for some time – to the same object in order for the robot to plan an action and interact with it (vergence requires some time before stereo fusion is achieved precisely). This decision could be made based on the stimuli alone (*bottom up approach*) or taking into account the internal status of the robot, its experience or the task to be performed (*top down approach*). In the first case the designer may decide to make red toys more attractive than grey ones, or to give higher priority to motion or to sound sources. In the second case the robot could get bored after some time it has been tracking the same object and decide to seek for a more interesting one (perhaps a novel one). The design of an attentional system is a tricky problem and goes beyond the scope of this research. Simple solutions were used in the experiments reported in this thesis. We used a tracking algorithm based on correlation between successive frames; the algorithm was originally written by Fitzpatrick at MIT-CSAIL and ran on Cog (Fitzpatrick, 2003) and was lately adapted to run on the Babybot. This tracker basically forces the attention of the robot to follow an object that is placed at the center of the cameras. Whenever a more controlled situation was required color segmentation was used instead; in this case the robot paid attention to objects with a predefined color. The robot can also keep memory of objects it has seen by storing their position in a map (Fitzpatrick, 2003; Johnson et al., 2003). The latter is coded in a body centric reference frame by using the kinematic model of the head so it does not change as the eyes or the neck move. In this way it is possible to redirect the gaze of the robot toward an object that is out of sight.

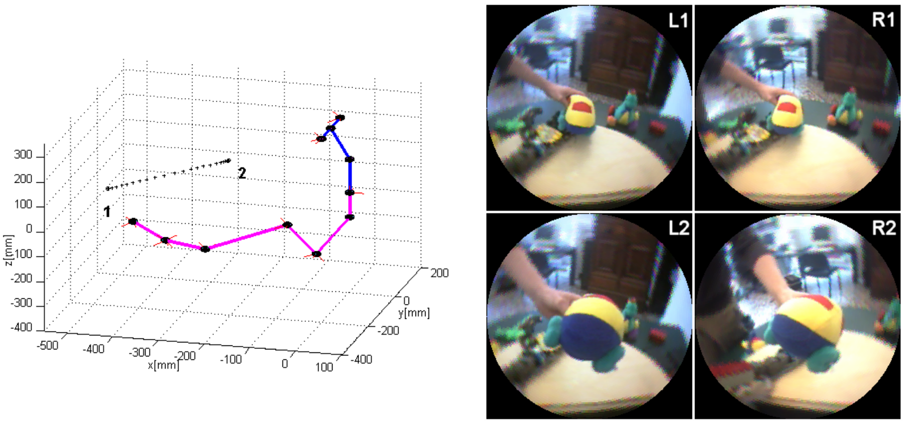


Figure 4.5. Vergence control: a toy is moved to follow a straight line toward the robot. Left: the trajectory of the fixation point in the 3D space. The fixation points is plotted with '+' every 10 frames (frame rate was 25 Hz), whereas the simple 3D model represents the robot: circles are the joints, solid lines correspond to the links (the arm did not move in this experiment). 1 and 2 mark the beginning and the end of the trajectory. Right: images from the right and left cameras at the same instants (L1-R1 and L2-R2 respectively). Notice that the car is maintained at the center of the visual fields of both eyes.

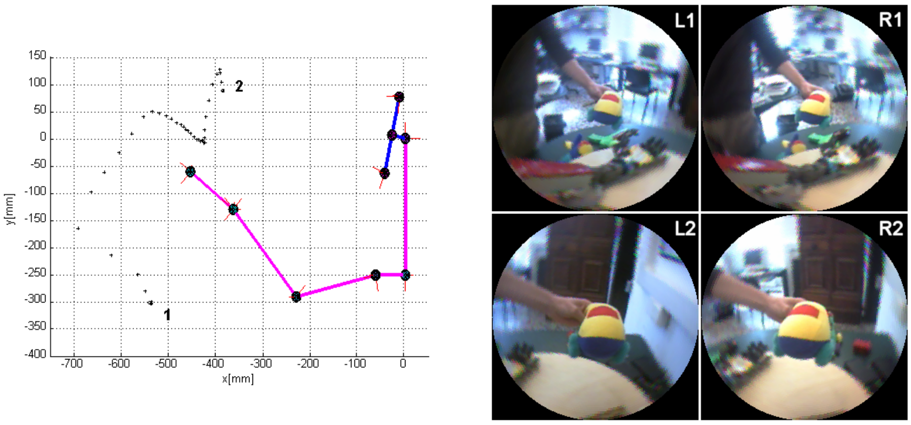


Figure 4.6. Tracking (vergence and version). A toy is moved while the robot tracks it. The left plot shows the trajectory of the fixation point during the experiment (top view); '+' marks correspond to the fixation point every 10 frames (frame rate was 25Hz). The 3D model sketches the robot: circles are the joints, solid lines correspond to the links (the arm did not move in this experiment). Initial and final points of the trajectory are marked with 1 and 2. Images at the same instants are reported on the right (L1-R1 and L2-R2 respectively). Notice that the car is maintained centered within the visual filed of the two eyes

Chapter 5

Learning a body map from experience

Before actually starting to explore the environment and act on it, the robot has to learn how to control its body. This requires a few additional abilities. Depending on the task, the control of posture and motion may be relatively easy or more difficult. The control of gaze, for example, is relatively simple because eyes and head have low inertia and are less likely to impact on unexpected objects. On the other hand the control of the arm can be more difficult due to the larger number of degrees of freedom and the higher inertia and loads involved. As outlined in the introduction (Chapter 1) these problems are addressed during the first phase of the developmental process. The robot has to be able to control gaze in order to fixate a particular location in space (or in the visual field) and to control the arm to reach out and eventually either grasp or touch the object on which it is fixating. In primates the brain maintains an internal representation of the different parts of the body, their relative position and physical properties such as size and weight. This *body schema* is thought to be used by the brain not only to plan motor actions but also to predict the outcome of potential actions before they are actually executed. Owing to this internal simulation the brain can abort possibly dangerous actions or modify an ongoing movement.

This chapter deals with the algorithms and learning mechanisms used by the robot to acquire and tune an internal representation of its body. These basic skills will be used afterwards during the interaction with the environment and the objects therein.

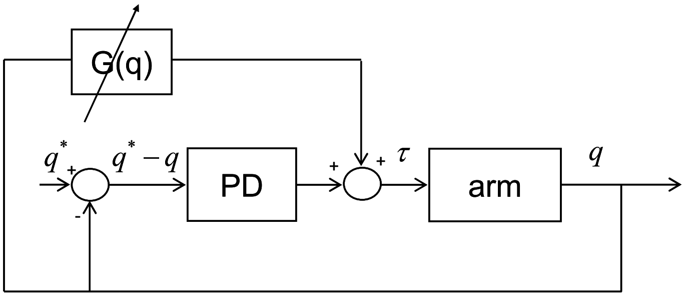


Figure 5.1. Arm control schema. It consists of two loops; a feedback inner loop employs a PD controller to achieve a desired joint angle. The block marked with G computes the gravity load term which is fed-forward to the control board.

5.1. Learning gravity compensation

Grasping an object involves the execution of a motor action with the arm to bring the hand close to the target object. To solve this problem the robot has to compute the position of the object in some reference frame (i.e. the retinal coordinate system) and transform it into a motor command suitable to drive the arm joints (i.e. motor torques). In robotics this problem has been solved in different ways. One solution is to first solve the inverse kinematics, obtain the final posture of the arm in joint space and then compute the torques necessary to drive the manipulator in that particular posture. To better formalize this last problem let us consider the Lagrange equation for a generic manipulator:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{5.1}$$

where τ is the generalized torque applied to the manipulator, M is the inertia matrix, C and G Coriolis and gravity vectors respectively. Assuming the static case $\ddot{q} \approx 0$ and $\dot{q} \approx 0$ equation (5.1) reduces to:

$$\tau \approx G(q) \tag{5.2}$$

which merely gives the torque required to maintain the arm in a given posture. If $G(q)$ is unknown a straightforward solution is to employ a PID controller:

$$\tau = K_p \Delta q + K_v \dot{q} \tag{5.3}$$

The latter is equivalent to simulating a damped spring-mass system with stiffness K_P. Substituting (5.2) in (5.3) in the static case gives:

$$\Delta q = G(q)/K_P \tag{5.4}$$

which states that Δq can be reduced at will by increasing K_P.

If the robot acts in an unstructured natural environment it is impossible – especially during learning – to avoid unwanted collisions with objects. A means to reduce the risk of damage due to shock and collisions is low impedance control; as discussed in Section 3.1 this can be achieved by reducing the proportional gain of the PID controller. If the gravity load term due to the arm's weight is known a priori it is possible to add it as a feedforward term to the PID output:

$$\tau + \widehat{G}(q) = G(q) \tag{5.5}$$

$\widehat{G}(q)$ decreases the torque the PID has to compensate; as a result it is possible to reduce the value of K_P (in theory, if the gravity was known exactly, the PID would not be required any longer). The resulting control loop is shown above (Figure 5.1).

The gravity terms vary with the configuration of the manipulator. In the case of the PUMA arm the feedforward control is a scalar function of three variables (q_1, q_2, q_3, see Section 2.1). To simplify learning, the controller uses only the following two variables[2]:

$$\begin{cases} q_1 \\ q_v = q_2 + q_3 \end{cases} \tag{5.6}$$

The functions whose parameters have to be estimated have been chosen empirically to be of the following form:

$$G(q_1, q_v) = aq_1^2 + bq_1q_v + cq_1 + dq_v + eq_v^2 \tag{5.7}$$

Learning takes place by random exploration of the arm workspace. At the beginning the feedforward controller does not contribute to motion, and only the low-gain PID controller is employed. As a result the control is not precise and the arm works under a large error condition. However, every time the manipulator stops in a particular posture (joint configuration) the controller can measure the current torque and use it as an estimation of the gravity term for that particular arm posture. This measure is fed to the controller and it is used as a training sample by the feedforward model. The form of the motion is that of equation (5.7) and is estimated by an iterative least squares procedure. After a certain number of learning steps the gravity compensation is activated and its output added to the output of the PID. If the forward model is precise enough the position error is consequently small. However, no force feedback is actually available in the PUMA arm and the output voltage of the control board was used instead (meaning that the frictional terms were not properly considered). For this reason, and due to the approximations we introduced, the PID controller is still required but with lower gain.

Figure 5.2, Figure 5.3 and Figure 5.4 show the error during the learning phase and the gravity term for different arm postures for the first three arm joints. The procedure waits for a few samples (25) to be collected before the system activates the feedforward block: this is visible in the first part of the error plots. As soon as the block is active the error is quickly reduced. In the case of the shoulder joint – which supports more weight – this is far more evident (Figure 5.2).

---

[2] q_1, q_2, q_3 correspond to the shoulder, arm and forearm respectively.

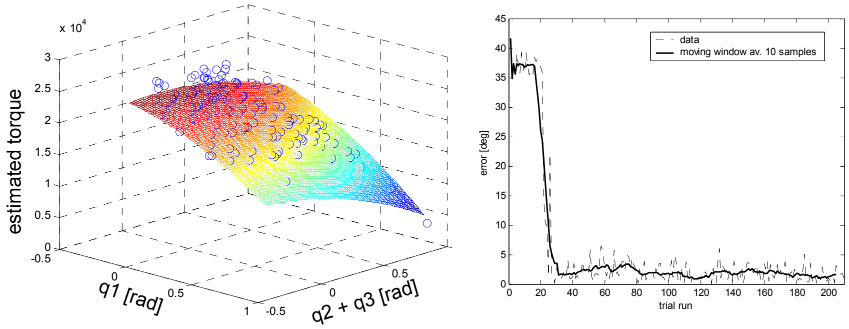


Figure 5.2. Arm gravity compensation, joint 1 (shoulder). Left: gravity load as a function of the arm joints; actual samples (circles) and estimated function (mesh) after 200 trial runs. Ordinate uses arbitrary scale (control board digital output). Right: error trend during learning, actual data (dashed line) and moving window average over 10 trials (solid line). After 25 trials the gravity compensation is activated and the error decreases quickly.

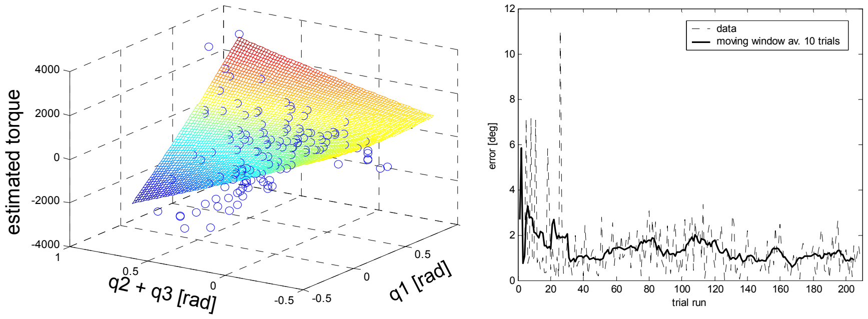


Figure 5.3. Arm gravity compensation, joint 2 (arm). Conventions as in Figure 5.1. In this case the improvement is less remarkable as the gravity load is lower compared to joint 1.

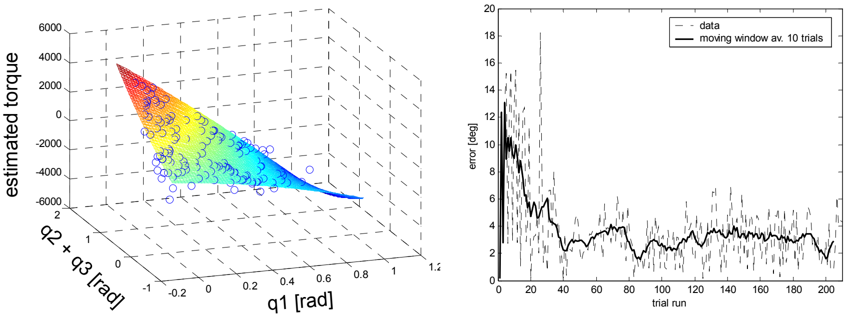


Figure 5.4. Gravity compensation, joint 3 (forearm). Conventions as in Figure 5.2. In this case the improvement is less remarkable as the gravity load is lower compared to joint 1.

5.1.1. Discussion

Gravity compensation is a well known technique in robotics. It has been presented here as a first simple example of body schema; the robot autonomously learns a physical property of its own arm (e.g. the weight of the shoulder, arm and forearm). It is important to stress a few points. Firstly, the experiment should be integrated with other behaviors; the robot could collect the learning samples while performing other tasks, for instance during the learning of the hand localization (described in the next section) or during reaching. Secondly the gravity estimation block can be conceived of as a sort of predictor whose output, for any given arm configuration, is the expected weight of the arm. Possible discrepancies between expected and real values could be interpreted as external events (e.g. a successful grasp, the presence or absence of a table). In all these cases, more accurate sensors like strain gauges, force as well as tactile sensors on the hand would be required to take frictions into account and discriminate the condition when the arm touches an object or the environment.

As a final note it is important to say that, given the particular problem and the relatively poor quality of the available feedback (the output of the control board), a second order polynomial function of two variables provided a sufficient approximation for the gravity term of the manipulator. This was a convenient solution because the output of such a simple function is generally stable and easy to monitor and visualize. Of course more accurate feedback would require a more powerful approximator (e.g. a neural network).

5.2. Learning to localize the robot's hand

One way to solve the inverse kinematics problem is to learn the functional relation between the head fixation point and the arm posture. This mapping can be used to program a ballistic movement of the arm toward the point in space upon which the head is. A closed-loop mechanism may still be required to compensate for small errors in case the ballistic motion is not accurate enough. In both cases the robot must be able to track the arm end-point (the hand in this case) and to distinguish it from the object it is going to grasp.

In psychology the ability of humans to match an action with the agent that caused it, is called the sense of *agency* (Jeannerod, 2002). The sense of agency provides humans with the sense of ownership of their actions. It requires at least the existence of a representation of the location of the arm or the other body parts (*body schema*). This knowledge can be derived from several cues, mainly visual and proprioceptive in nature. The former determine the "seen" position of the body, the latter its "felt" position. Neurons in the premotor cortex of monkeys (area 5) have been found to code posture information about the arm (Graziano, 1999). Graziano and coworkers (Graziano, 1999; Graziano et al., 2000) tested the response of these neurons in different conditions. In the first experiment the arm was visible, while in another condition it was covered with a plastic barrier to prevent visual information. In both cases the receptive field of each neuron shifted according to the position of the arm, showing that both the sight of the arm and its felt position contributed to determine its spatial location. As well as this, visual information alone modulates this response; this was confirmed by a third experiment where a realistic fake arm was used to dissociate visual feedback and proprioception. Taken together these results show that vision and proprioception converge in the premotor cortex to encode space in an arm-centric reference frame. The fact that this area projects directly to the primary motor cortex and to the spinal cord suggests that this representation contributes to the control of limb movement.

At this point one could wonder what are the mechanisms used by the brain to build and maintain such a representation during development; in fact self-knowledge emerges early on in humans and it is undergoing construction right from the beginning of infant development (infants at 5 months of age distinguish their own leg movements on a mirror from those of another infant). Combined double touch and multimodal correlation allow babies to find out that their body is a unique entity in the environment. By moving their limbs around, babies learn that when their hand touches their face they feel a synchronous tactile stimulation on both hand and face. The same does not happen when they grasp an object or touch the floor when they crawl. Also, infants coordinate information from different perceptual systems to disambiguate between parts of the visual field whose motion matches what they expect based on their proprioceptive and kinesthetic feedback; they know that these parts of the visual fields are likely to be part of their own body. In particular perception of *intermodal form* is thought to have the most

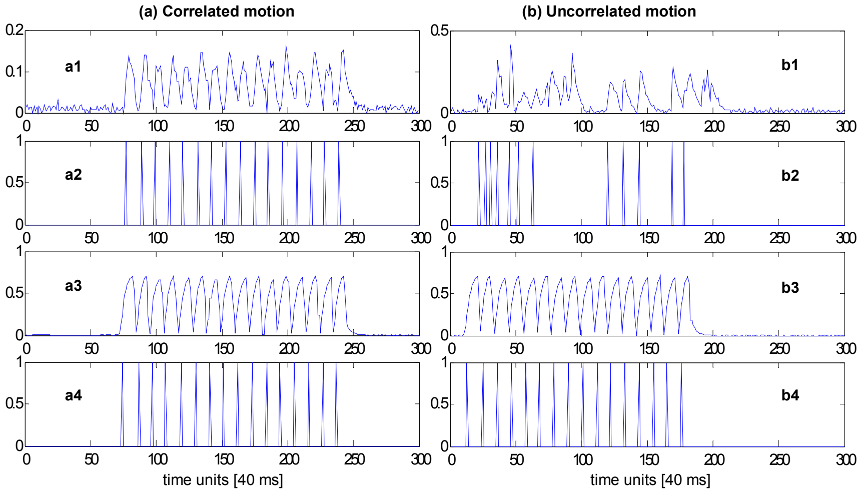


Figure 5.5. Examples of correlated (a) and uncorrelated motion (b). The picture plots motion in the images for two pixels (a1 and b1) and the result of the zero-crossing algorithm (a2 and b2). Arm motion for the wrist joint is reported below (a3 and b3) together with the result of the zero-crossing algorithm (a4 and b4). By comparing a2 to a4 and b2 to b4 it is clear that (a) corresponds to the pixel that belongs to the hand. Ordinates are arbitrary scales (normalized values).

significant role in the development of self-perception. This term represents the existence of sensory pattern that are similar across different sensory channels. During development infants become more attuned to similarities of pattern between proprioceptive and other sensory feedback. Other factors, like *spatial correlation* and *timing* (time coincidence of events) have a more flexible contribution in the self identification process. A possible explanation is that this flexibility is necessary later on during development for learning tool use (in this case tools extends the body), or for detecting causal links between one's actions and their corresponding, delayed, consequences (Rochat and Striano, 2000).

Similar approaches have been pursued in robotics as well. Yoshikawa et al (Yoshikawa et al., 2003) exploited invariances in the multi-sensory data. The idea, in this case, is that the body is invariant with respect to the proprioceptive information available to the robot. A neural network was hence trained to segment the arms of a mobile robot from the background. A somewhat different approach was followed by Fitzpatrick and Metta (Metta and Fitzpatrick, 2003) who exploited self-generated motion to segment the manipulator endpoint from the background. The rationale was that among the objects in the scene the body is defined as the one that behaves in an expected, repetitive way. Correlation between motor commands and motion in the visual field was used to segment the hand. The robot performed a periodic movement of the forearm to generate motion in the visual scene. Optic-flow was then compared to the proprioceptive feedback to select regions of the image more likely to be part of the robot's arm.

The approach followed here is similar. The wrist of the robot was moved in order to produce a small movement of the whole hand. Visually, a simple motion detection algorithm was implemented, computing time difference information between each frame and a model of the background (the description of the algorithm is reported in Section 8.1). For each pixel in the "motion image", a zero-crossing algorithm was used and a periodogram computed. Similar periodic information was computed on the motion signals coming from the motors. Pixels that moved periodically, and whose period of oscillation matched that of the motor commands, could be selected as being part of the hand. Parts of the image whose motion was uncorrelated (different period or aperiodically) could be segmented out (i.e. someone walking in the background). Figure 5.5 shows an example of the detection procedure for two different pixels whose motion is correlated (a) and uncorrelated (b) with that of the robot's hand.

5.2.1. Segmentation and prediction

The output of the algorithm is a pixel map; in order to get a dense segmented region a series of low-pass filters at different scale are run on the pixel map image. Processing is carried out in the log-polar domain by using the *integral image* representation; originally proposed by Viola and Jones (Viola and Jones, 2001) the integral image allows for fast computation of multi-scale filters. A simple threshold was applied after filtering to get the region of interest as explained in Figure 5.6. As it is, this algorithm cannot be used to track the hand of the robot or to localize it

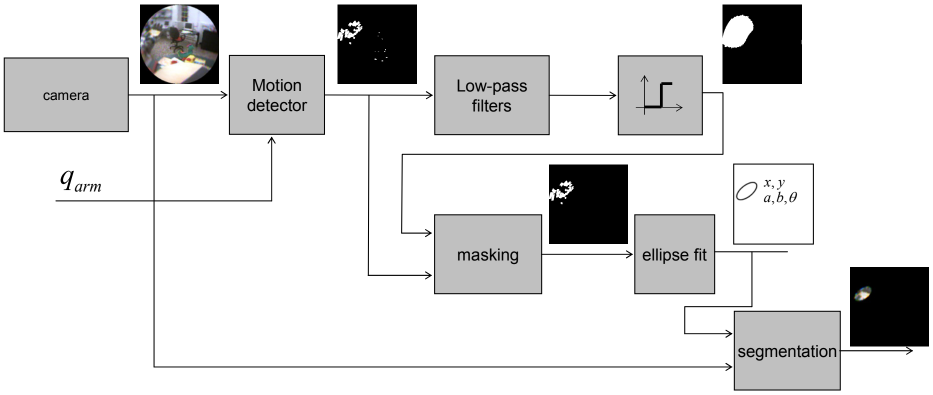


Figure 5.6. Hand segmentation schema.

during a grasping action. Nevertheless it is a good starting point for building more elaborate models of the hand, including visual features like color and shape. In particular a simple color histogram was evaluated. The color histogram was computed in the Hue-Saturation space and it was constructed from many results of the motion-based segmentation. Here we exploited the fact that the body is invariant with respect to the environment and that eventually the background contribution cancels out. Figure 5.7 shows the result of this process.

In general color is an appealing feature for object recognition and localization, owing to its invariance to scale, rotation and to occlusions. The histogram gives a statistical description of the colors of the hand; by comparing this distribution with that of the pixel in the image, it is possible to identify regions that are more likely to be part of the hand (histogram intersection). If such regions are more or less uniformly colored (as in the case of the palm of the robot's hand) histogram backprojection could be used instead. Although much faster, backprojection is less

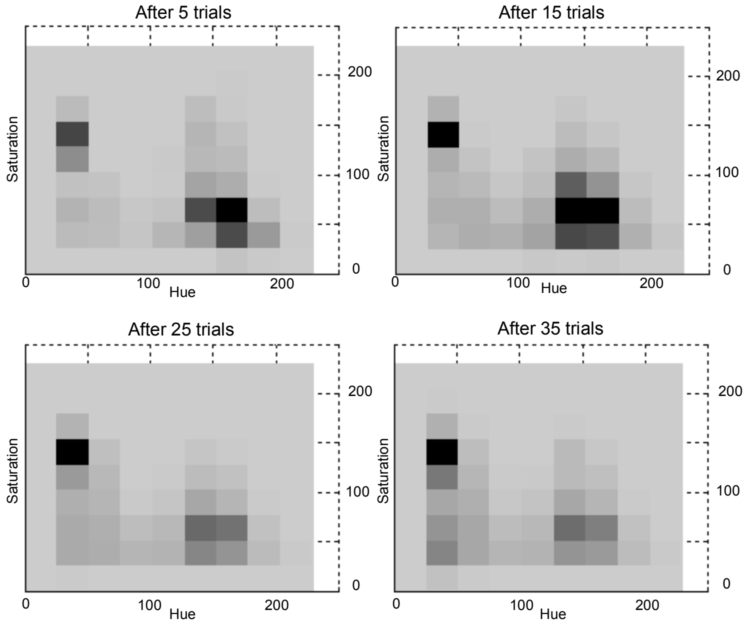


Figure 5.7. Hand color histograms during learning (top-view). The Hue-Saturation space is divided in 10x10 bins to sample the interval from 0 to 255; different shades of gray are used to represent the probability: from light gray (0.0), to dark gray (1.0). Histograms are normalized with respect to the maximum. As the learning progresses the contribution of the background cancels out and the histogram gets skewed toward the color of the hand (about (30, 150)).

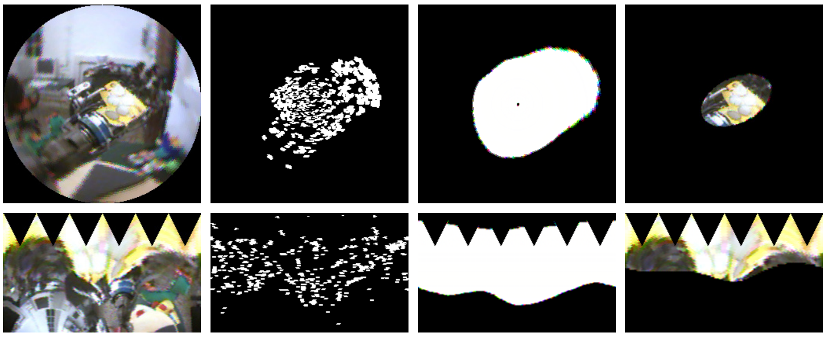


Figure 5.8. Example of hand detection and segmentation (1). Top sequence, from left to right: original image, result of the detection algorithm, low-pass filtering, ellipse fitting and segmentation. Bottom: the same sequence in the logpolar domain.

accurate and easily fooled by objects with similar color. Furthermore the histogram cannot be tuned too sharply, as color is sensitive to spectral changes in illumination and reflectance. To overcome these intrinsic limitations, shape information can be used to further disambiguate between similar regions. A parametric model could be fitted to the contours of the segmented image and integrated into the recognition process (e.g. size and orientation of the hand). To this aim several algorithms could be used (Kass et al., 1988; Leymarie, 1990); we used a moment based ellipse representation instead (a complete description of the process is detailed in Figure 5.6 and Figure 5.8; see also Section 8.2). As geometric properties do vary with distance and rotation a predictive model consisting of two neural networks was employed.

Training data was collected by repeatedly moving the arm at different spatial positions and by running the segmentation. The inputs to the first neural network are the arm joint angles and the training samples are the position of each segmented blob. In principle the latter should take into account also the position of the cameras (e.g. the head posture). Unfortunately in this case the input space would be too large and the number of samples required too high for learning to be effective.

The following solution was applied instead. In practice the network learns to predict the position of the hand in an *egocentric* reference frame. The original retinocentric (x,y) information is converted into a bodycentric reference frame and sent to the neural network as a sample for learning; afterwards the output of the network is converted back to the retinal plane (see Figure 5.9).

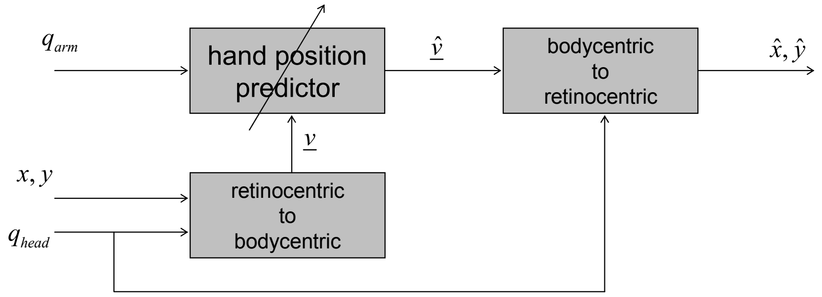

q_{arm}
hand position predictor
$\underline{\hat{v}}$
bodycentric to retinocentric
$\hat{x},\hat{y}$
$\underline{v}$
x,y
q_{head}
retinocentric to bodycentric

Figure 5.9. Hand position predictor schema.

This simplified the learning process because the input space was narrower. Essentially both transformations, back and forth from egocentric to retinocentric representation, involve knowledge of direct and inverse kinematics of the head. In the current implementation this was hardwired in the system; in the next section we discuss how this assumption might be loosened.

The inputs to the second neural network are the arm joint angles and training samples are the shape parameters as obtained by the fitting algorithm. The head posture in this case is not considered, as the shape parameters do not vary if translational effects are negligible (Figure 5.10). As soon as samples were gathered the robot could start exploiting the localization system to track the hand. The detection was further improved because the hand was more likely to appear in the centre of the visual field; shape parameters are in fact unreliable when the hand is in the periphery or partially out of sight (Figure 5.12 shows some examples of segmentations during learning).

The robot carried out the learning in a self-supervised way. The neural networks were trained online in the following way: the learning module keeps a list of the training samples. When enough patterns are collected the network is randomly initialized and the learning algorithm starts. Training samples occurring during this process are stored in the list for later use. The list of training patterns thus gets longer as new samples are gathered, although it is possible to discard (forget) older samples with a first-in-first-out policy.

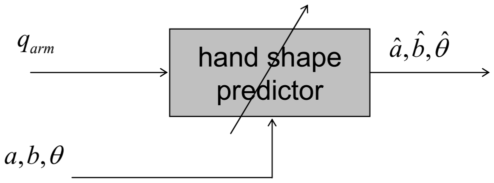

q_{arm}
hand shape predictor
$\hat{a},\hat{b},\hat{\theta}$
a,b,θ

Figure 5.10. Hand shape predictor schema.

In the case of the hand localization the outcome of learning was validated by testing the ability of the robot to predict new hand positions. Thus, as soon as a new sample is available it is compared to the current output of the neural network. As the learning progresses new samples tends to be closer to the output of the network, meaning that the prediction has improved (Figure 5.11).

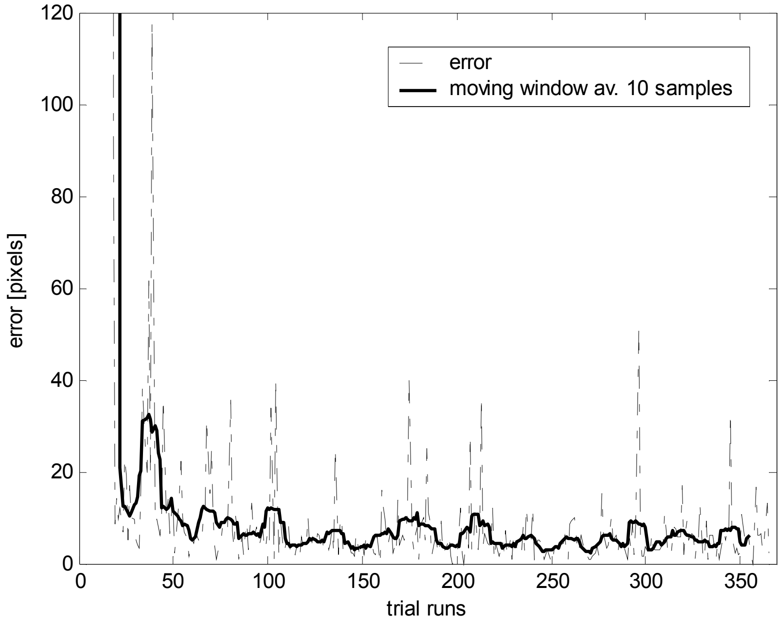


Figure 5.11. Testing the learning performance. As soon as a new sample is available it is compared to the current output of the neural network. As the learning progresses new samples are closer to the output of the network, meaning that the prediction has improved. Ordinate reports the root square error in the image plane in pixels (dashed line is original data, solid line is moving window average over 10 samples).

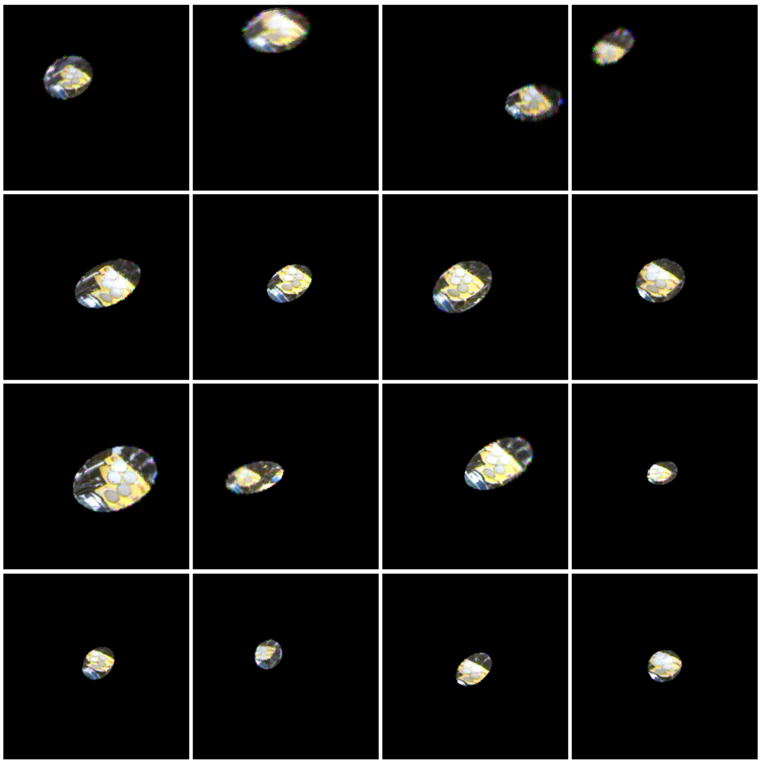


Figure 5.12. Examples of segmentations. As learning progresses (from top-left to bottom-right) the robot starts tracking the hand; as a result the hand is more likely to appear in the center of the visual field.

5.2.2. Exploiting the hand prediction

As described in the previous section, after training the robot is able to compute the position of the hand given the current posture of the arm. The hand localization block consists of a mapping between the arm joint angles and the Cartesian coordinates of the arm end-point in the visual field. It is now straightforward to implement a tracking behavior by connecting the hand localization block to the controller of the eyes described in Section 4.4. This "blind" tracking proved to be surprisingly accurate; for this reason visual information was not used to improve localization (Figure 5.14 shows a sequence of the robot tracking the hand). The color histogram was employed as a simple means to confirm the output of the forward module; pixels of the histogram backprojection were integrated and thresholded over the ellipse area to detect if the hand was occluded or not (Figure 5.16).

The very same mapping could be used to predict the final outcome of a reaching movement. Before executing an arm movement, the robot can query the map with the desired arm joint angles, and compute (predict) the expected location and shape of the hand at the end of the movement (Figure 5.15). The prediction might be used in several ways, for example to improve the hand tracking behavior.

5.2.3. Discussion

In this section we have shown the mechanisms by which the robot can autonomously develop an internal model of its body. The robot exploits self-generated actions and seeks correlation between proprioceptive and visual information to segment its own hand from the background. Compared to other methods of segmentation this solution is more effective because it can cancel out parts of the background which move in different (uncorrelated) ways. This segmentation is used as a bootstrapping mechanism to extract invariant features that will be used to perform the localization without the need to go through the detection process all the time. A forward model is trained to compute position and shape of the hand based on the arm posture. The localization was used by the robot to track the hand.

The hand localization system provides a response similar to the one observed in the premotor cortex of monkeys (Graziano, 1999; Graziano et al., 2000), where neural response is modulated both by visual and proprioceptive information. However some neurons were found to be less influenced by the sight of the arm, whereas others exhibited a response that was modulated more by visual feedback. In the system proposed here only proprioception is used; in this respect we modeled only part of the response of the premotor cortex. In humans the visual response is probably required to improve the spatial sensitivity of the localization and compensate for the relative inaccuracy of muscular proprioception. In robots, motor encoders are usually much more precise (fractions of degrees); in this case vision may still be helpful to correct drifts and avoid calibration.

The idea of a body schema is not completely new in robotics. Yoshikawa and colleagues (Yoshikawa et al., 2003) exploited the idea that the body is an invariant entity in the environment to train a fully connected network and build a cross modal map of the robot's body. By moving around the robot "labeled" those parts of the environment that proved to be invariant as being part of its own body. Although visual and proprioceptive information were included in the model, the experiment was limited to a single body posture (e.g. the robot moved in the environment but its arms did not). Assuming the arm is the only moving entity in the world, Marjanovic (Marjanovic et al., 1996) used optic-flow to segment the manipulator end-point. Arsenio and colleagues (Arsenio et al., 2003) used periodic movements for segmenting an object waved by a human teacher. In this case the hypothesis being that the object was the only entity in the visual scene moving

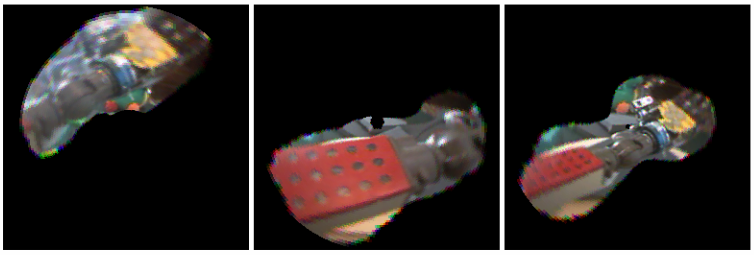


Figure 5.13. Forearm segmentation. The algorithm used for the hand localization could be replicated for other body parts. In this case three examples are reported for the segmentation of the forearm.

periodically. These restrictions were loosened by Metta and Fitzpatrick (Metta and Fitzpatrick, 2003) who followed a similar approach to the one presented here. They used optic-flow and computed cross-correlation between motion in the visual field and the arm motor command. Cross-correlation and optic-flow are probably more biologically plausible solutions and, in principle, they could allow more general movements to be exploited (instead of periodic ones). On the other hand, both cross-correlation and optic-flow are quite expensive from a computational point of view. For this reason they used coarse resolution images (128x128) and computed optic-flow over blocks of 16x16 pixels. Motor and visual channels were manually synchronized to compensate for different delays in the channels. In the case presented here the algorithm is much simpler and can be run in real time in the log polar domain at the resolution of 152x252. As a result the segmentation is a dense image which allows additional features like color and shape to be extracted. Since the period of oscillation is exploited instead of the cross-correlation, the detection is independent of the delays and there is no need to synchronize the signals.

The mechanism used to learn the hand localization could be extended to other body parts. For example the same algorithm was applied to the arm; the robot segmented the region of the image that moved as the result of a periodic movement applied to the forearm. The results of a few trials are reported in Figure 5.13; in this case it is not possible to distinguish between the hand and the forearm as both moved in the same way. By first learning the localization of the hand, however, it would be possible to remove its contribution to the motion of the forearm. This procedure, applied from the distal to the proximal limbs, allows building a complete model of the robot body.

A few simplifications are worth discussing. During the detection of the periodic movement the head was kept stationary; this was required to facilitate motion estimation. At least at low speed the head motion could be compensated for by improving visual stabilization and by estimating the motion component due to the head (egomotion). To simplify the learning of the hand localization module, we assumed knowledge of the head kinematics. This reduced the state space for learning that otherwise would have been too large to be explored. The position of the hand in the visual field was converted into a body centered reference frame; for this purpose only the rotational component of the head kinematics was actually required. This could be estimated by tracking parts of the visual scene during random movements of the head and learning a correspondence between motor commands and corresponding retinal slips.

Finally, other cues could be integrated in the segmentation. For instance the algorithm discussed here used only information from a single camera, but the segmentation could be improved by including disparity information (Bernardino and Santos-Victor, 2002).

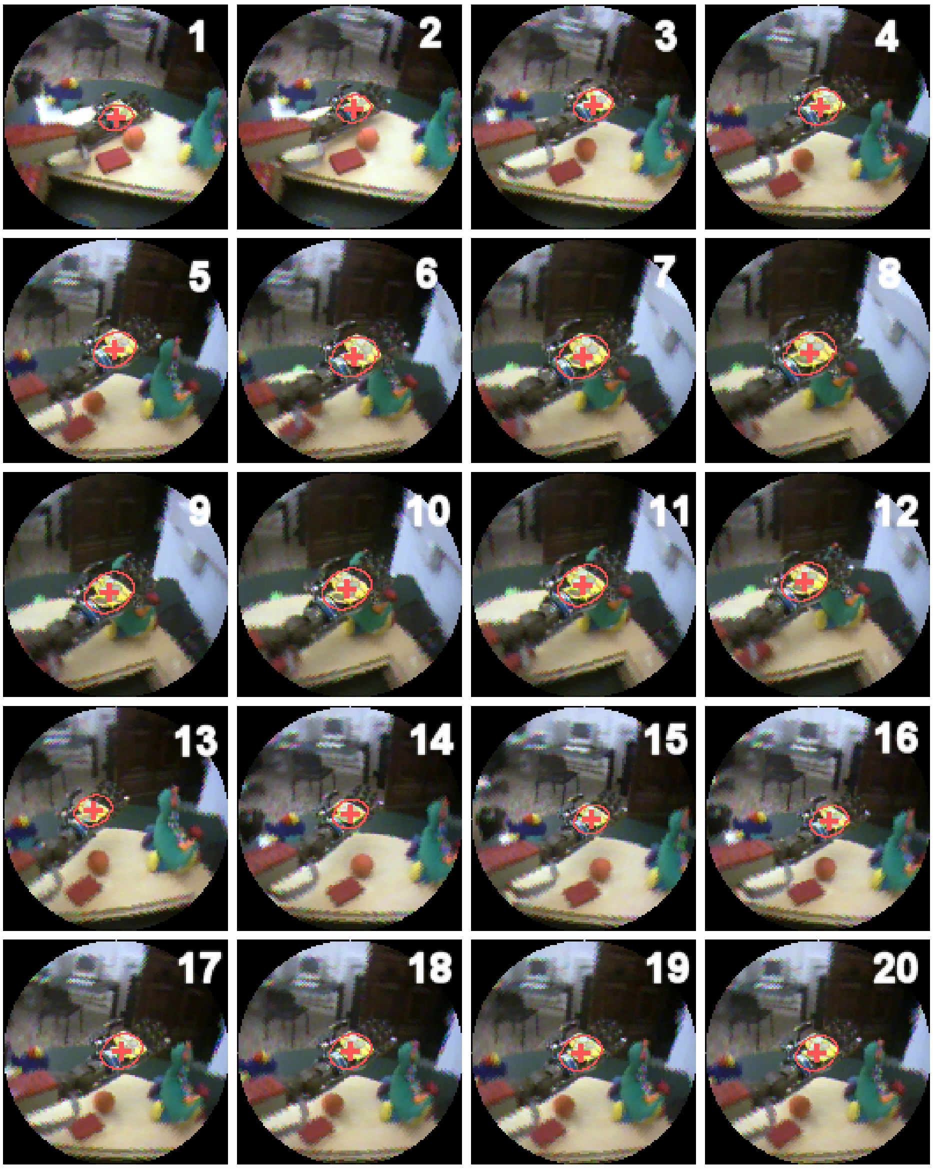


Figure 5.14. Hand localization (1). Frames from a 20 second sequence of the robot tracking the hand (each frame is taken at 1 second interval). The cross represents the position of the hand estimated from the arm posture, the ellipse plots its approximate shape; the gaze of the robot is controlled to maintain fixation on the cross.

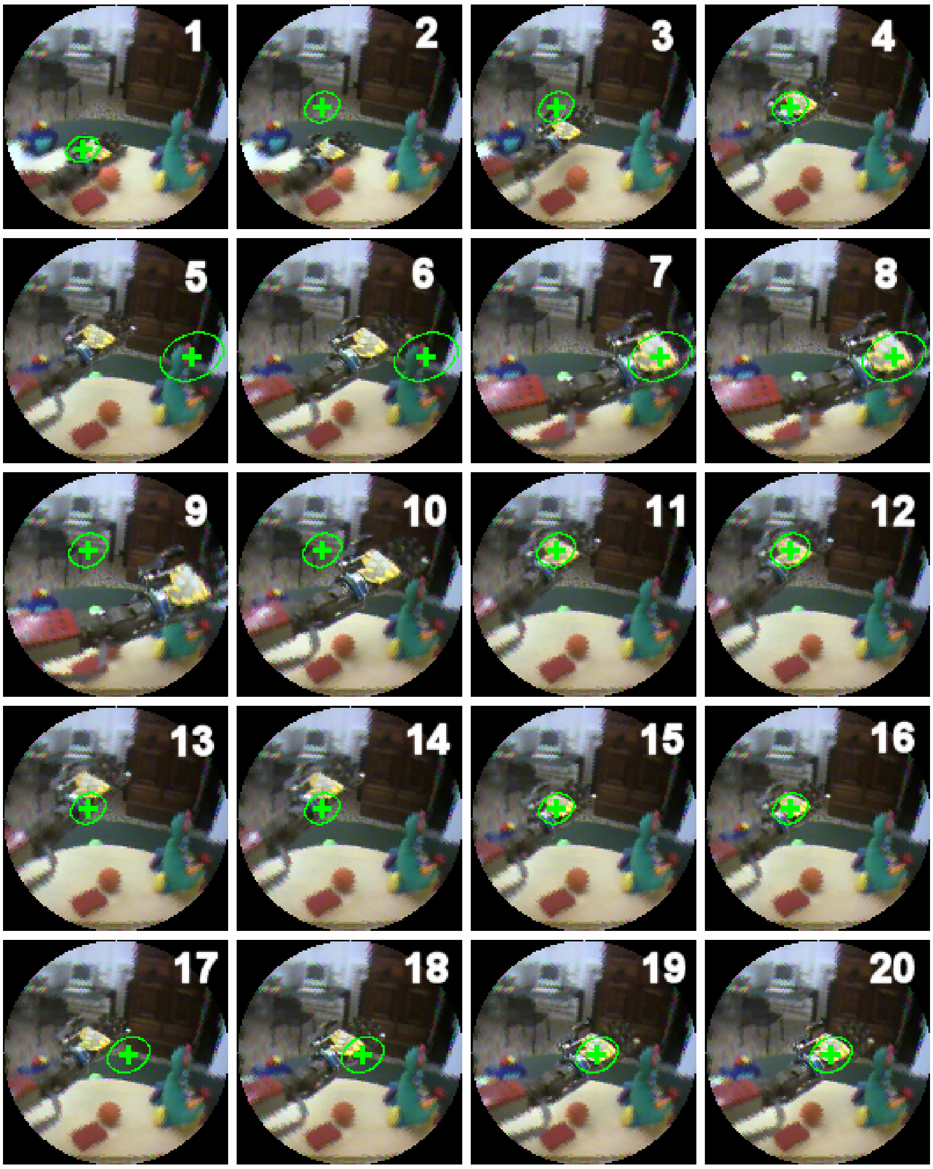


Figure 5.15. Hand prediction. In this 20 second sequence the head of the robot does not move (notice the position of the toys in each frame). At frames 2, 5, 9, 13 and 17 a new motor command is issued; for each of these commands the map predicted the region of the image the hand will be at the end of the movement (frames 4, 8, 12 and 20 respectively).

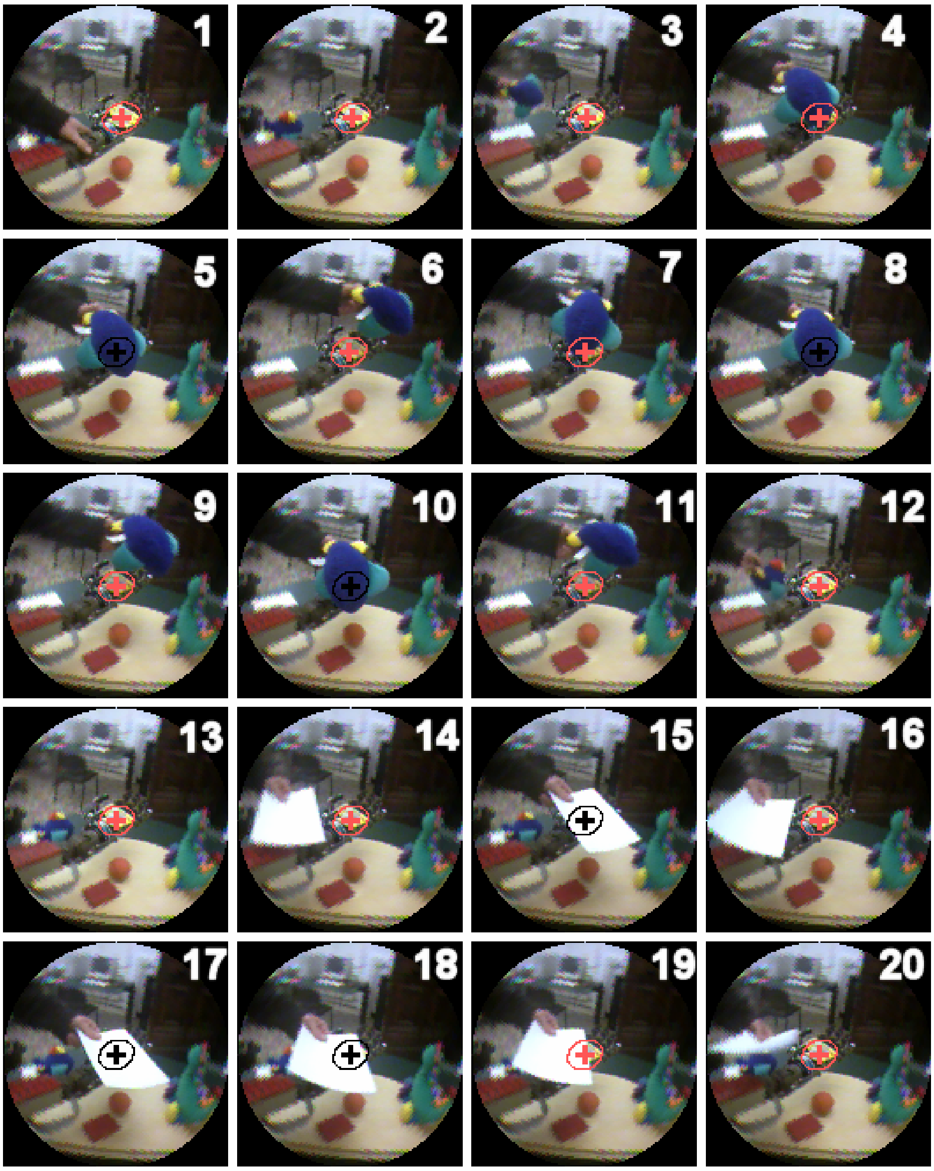


Figure 5.16. Hand localization (2). The color histogram is used to check if the hand is actually visible or not. Both arm and head are stationary in this sequence; different objects are introduced to cover the hand. Cross and ellipse are depicted in black when the hand gets completely occluded (frames 5, 8, 10, 15, 18).

Chapter 6

Learning to act on objects

In this section we move forward from the exploration of the robot's body to the exploration of the environment: the robot adventures to explore the outer world. Three experiments are reported. In the first one the robot learns to reach for objects by building a mapping between the target location – expressed by means of the hand position when the object is being fixated – and the appropriate arm motor command. The second experiment (published in (Fitzpatrick et al., 2003; Natale et al., 2002b)) deals with the problem of learning how objects behave in the world when touched. Very simple actions – like pushing and poking – are used by the robot to explore the environment and link the effect of its own actions to the object's behavior. This link is afterwards used to plan a motor action to achieve a desired goal. In the third experiment the robot uses the hand and a stereotyped initial motor synergy to explore physical properties of objects like shape, softness and weight. A representation of objects in a set is built by the robot with very little prior knowledge.

6.1. Reaching

In order to reach for an object in space the robots need to solve two problems: the first concerns the kinematic transformation between the position of the object in space and the corresponding posture of the arm, the second is the computation of the actual trajectory to achieve this posture with the appropriate dynamics. At least in part we have dealt with the latter in Section 5.1, where we have shown the control schema used by the robot to generate motor torques required to attain a desired joint configuration. In this section we address the first problem: given an object in space, how to convert the visual information describing its spatial location into the appropriate sequence of motor commands. In particular we are interested in studying how the robot could learn autonomously this transformation.

The very first aspect of the problem is the computation of the spatial location of the object in space from its projection to image plane. In theory stereoscopic information must be used to solve this problem; the distance of the object can be computed from binocular disparity between the stereo images. In practice, however, stereo reconstruction requires calibration of the cameras and hence precise knowledge about the camera parameters and their relative position. Besides, it assumes we are able to determine to location of matching points in the left and right images (the *correspondence* problem). The problem is further complicated because some points of the object in one image may be occluded in the other. For this reasons, techniques for depth estimation from stereo images are usually not suited for real-time applications.

A possible solution is to rely on the tracking behavior and assume that the object to be reached coincides with the fixation point of the head. Fixation is achieved by fusing together the *version* and *vergence* components of the movement, as described in Section 4.5. Vergence, of course, makes use of disparity cues but the measure is computed globally on the fovea of the two eyes and it is thus faster and more accurate. Once the target is fixated, the posture of the head implicitly defines the goal for reaching. This approach, originally proposed by Metta (Metta et al., 1999), uses a “motor-motor” map (i.e. a mapping between joint angles) to link the position of the head with the motor command required to move the arm end-effector to the fixation point. This motor command is specified as the final position in joint space.

This map can be easily acquired by tracking the end-effector: during an exploratory phase the robot moves the arm while tracking the hand. For each arm position the corresponding head posture defines a sample for training the map. A neural network is trained to compute the arm position based on the head posture, that is:

$$\underline{q}_{arm} = f\left(\underline{q}_{head}\right) \tag{6.1}$$

Reaching starts by first fixating the object; once fixation is completed the head posture $\underline{q}_{head}$ is used to address the motor-motor map and to recover the arm command $\underline{q}_{arm}$. Metta proposed to mix goal directed movements and learning. His approach was inspired by observations about development of reaching in infants. In fact at birth infants are already able to perform arm movements; these movements are not completely reflexive but can be directed towards objects or the mouth (von Hofsten, 1982). Although in some cases arm motion is visually controlled to

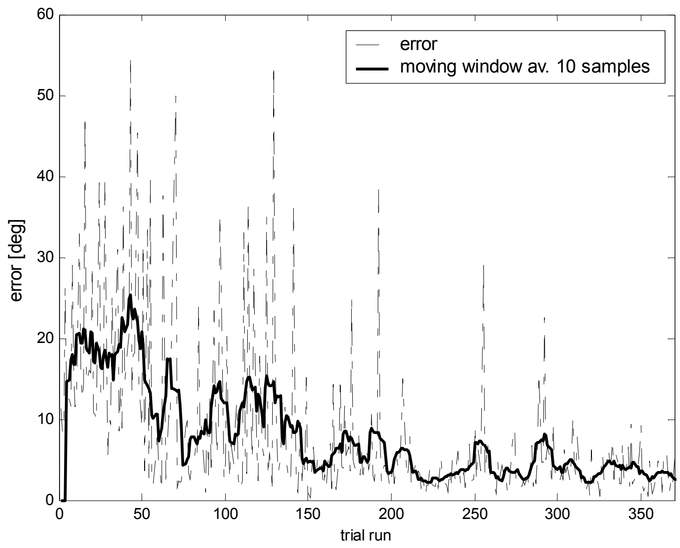


Figure 6.1. Testing the learning performance. Whenever a new sample is acquired it is also used to address the network The output of the network is compared to the same input to estimate the ability of the network to predict new samples (see text). The error trend shows that the training is consistent.

maintain view of the hand (Van der Meer et al., 1995), arm trajectories are mostly ballistic and vision is not used to control and correct the ongoing action (see (McCarty et al., 2001) for a review).

In the robot the map of equation (6.1) was initialized with three values to mimic a (crude) reflexive head-arm coordination. These three reflexes were designed to maintain the arm within the field of view; in practice the arm would stretch toward the direction of gaze, the latter being discretely sampled in three areas, right, center and left. The exploration was then achieved by adding noise to the arm command. Accordingly the robot would start reaching for objects by using the initial reflexes; at the end of each run the head would move to fixate the hand and collect a new sample to fill the map with. The initial noisy configuration allows the arm to explore the joint space and fill the "space" of the map in between the initial reflexes. This exploration strategy is advantageous because it limits implicitly the movement of the arm within a safe portion of the workspace; furthermore the behavior of the robot bias the exploration towards points that are more commonly used (e.g. the space in front of the robot).

In the experiment reported in this section a slightly different strategy was used. The first part of the reaching proceeds as described before. The difference is that after reaching the robot performs a few small random movements while tracking the hand. The noise follows a Gaussian distribution with mean value of 0 degrees and standard deviation of 5 degrees. This strategy was implemented to speed up learning and gather several training samples out of each reaching action.

Learning was tested online during the acquisition of the samples. As the location of the target is not directly accessible, to provide a measure of performance the following method was used instead. After each sample $\left(\underline{q}_h^n, \underline{q}_a^n\right)$ is acquired – and before it is used to perform a learning step – the network is addressed. The output vector $\underline{q}_a$ is then compared to the current sample to compute the error:

$$error = \left\|\underline{q}_{error}\right\| = \left\|\underline{q}_a - \underline{q}_a^n\right\| \tag{6.2}$$

In this way we tested the capacity of the network to predict new samples; the trend of the error during about 400 trials is reported in Figure 6.1. Notice that the average error decreases, meaning that the learning is effective and consistent. After the learning the robot can reach for an object it is fixating at (two examples are shown in Figure 6.3 and Figure 6.4). Since it is not possible to command an instantaneous joint transition to the arm, a mechanism to generate smooth trajectories is required. A simple linear trajectory generator is used to interpolate joint angles between the actual and the final position; each command is hence sent to the low-level control board which computes the torque to drive the motors. The control loop runs at 1 KHz frame rate whereas the trajectory generator runs at 25Hz.

A final note concerns the size of the input to the reaching map. The head posture is specified by 5 joint variables, but not all of them are required to uniquely specify the position of the target. In the first implementation by Metta the input vector was reduced by heuristically coding the direction of gaze into 3 variables only (version, vergence and tilt, respectively). In this experiment the head kinematics was employed to reduce the head posture to 3 variables representing the position of the fixation point in the Cartesian space (x,y,z). This solution is more precise but less biologically plausible; another coding might use the orientation components of the head kinematics and substitute the vergence angle for distance. A possible strategy to estimate the head kinematics is discussed in Section 5.2.3. The complete arm control loop is reported below (Figure 6.2).

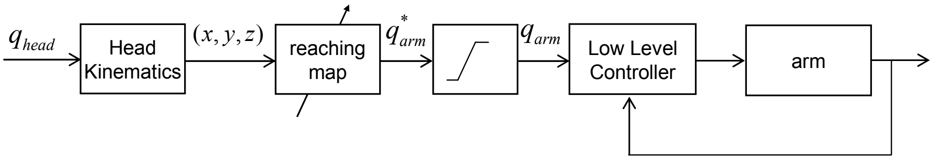


Figure 6.2. Arm control schema (reaching). The fixation point is computed from the current head posture (joint angles). The motor-to-motor map converts it into the desired command for the arm; the trajectory generator produces a set of "smooth" commands which are sent to the low-level controller (the latter is described in Section 5.1).

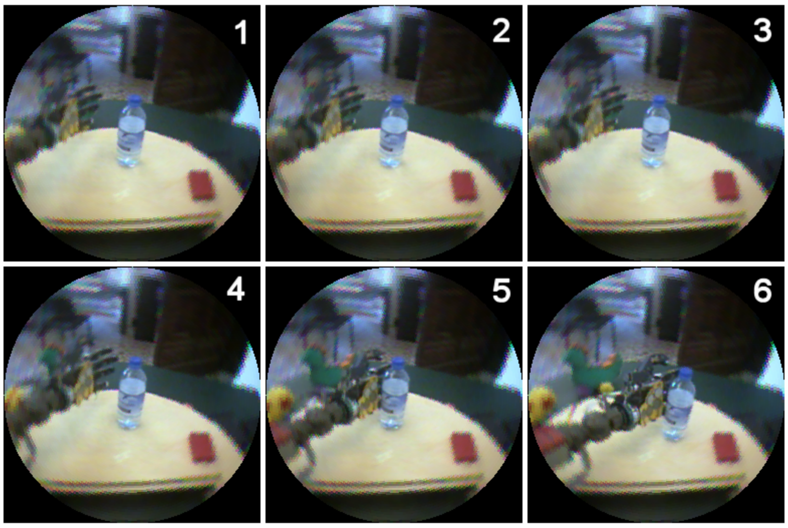


Figure 6.3. Reaching sequence (1). The robot tracks the bottle and then reaches for it. Frames are taken at 1 second each, the sequence lasts 6 seconds.

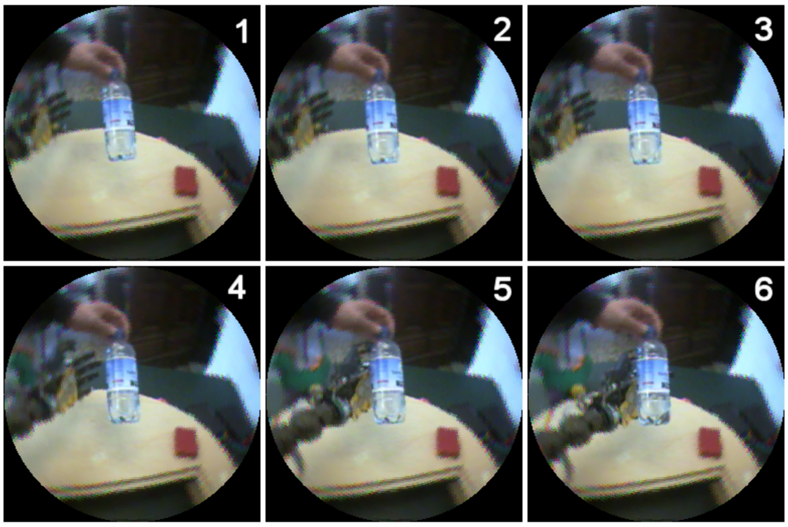


Figure 6.4. Reaching sequence (2). The robot tracks the bottle and then reaches for it. Frames are taken at 1 second each, the sequence lasts 6 seconds.

6.2. Learning to act on objects

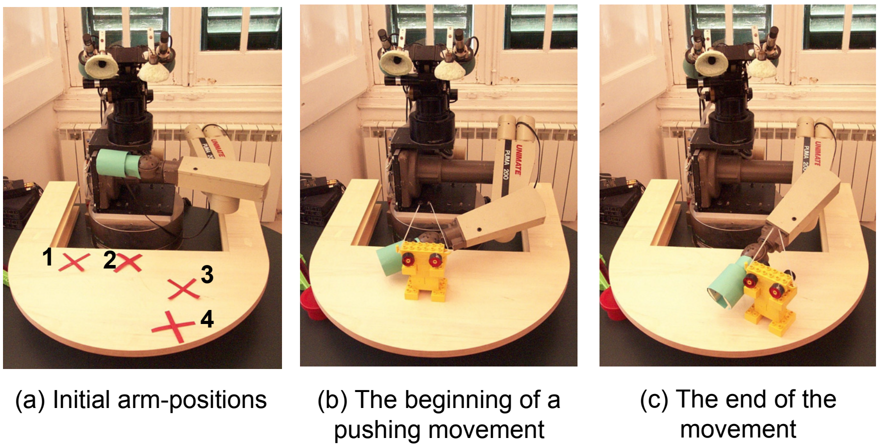
In this experiment the robot learns the effects of its actions on the object and thereafter uses this knowledge to drive motor-planning. It is important to note that by "effect" we mean not only the effect of the object, but also the effect on the robot – the force felt by the robot or the amount it had to move its head to continue tracking, for example. In this experiment we consider only one effect: the direction that the object moves in, as a result of the action. There are naturally many other effects that one could also pay attention to: how far the object moves, how long the object continued moving after the initial contact and so on. However, in the experiment described here the robot attends only to the instantaneous direction of motion of the target just after it has been pushed/pulled by the robot.

The goal of the experiment is to learn the instantaneous direction of motion of the target object for each of several different approach motions of the hand from different directions. This learned knowledge is later used by the robot to select the appropriate motor action to move an object in a desired direction.

6.2.1. Description of the experiment

This experiment was carried out before the robotic hand was mounted on the arm; for this reason the manipulator endpoint is a simple metal stub. The tracking and hand localization algorithms were also not fully developed. Color segmentation was used to localize and track the toy placed on the robot play-table and the arm endpoint. Color segmentation was carried out in the HSV space on the log-polar images from both eyes. Retinal target position was then extracted as the centroid of the segmented region. Gaze was controlled as described in Chapter 4. At the beginning of each run the robot started from a randomly chosen initial position in a set of four (Figure 6.5). While reaching for the target the robot continuously fixated it; thus, ideally, the object was always centered on the fovea, whereas the moving hand was being tracked – not fixated – in the peripheral vision. Figure 6.5 shows a possible sequence of such a trial. The pushing action started from position 1 (b) and ended with the target having been shoved to one side (c). The moment when the hand first touched the object – *moment of impact* – is important because it signals the instant when the relevant measures have to be taken. This instant corresponds to a large error in tracking and was localized by using the sharp increase in the magnitude of retinal target position. At this moment the direction of displacement (*in retinal coordinates*) of the target was extracted. No transformation to body-centered reference coordinates was required in this experiment as the robot built a link between this retinal error and the corresponding motor action.

After the initial impact the system continued to track for the object if it was still in the field of view, or lost track of it. In both cases a human brought back the object to the center of the table so that another run could start.



(a) Initial arm-positions (b) The beginning of a pushing movement (c) The end of the movement

Figure 6.5. Details of the experiment. (a) the four initial positions for the arm. (b) and (c) report a pushing trial run.

During each trial, the robot continuously monitored several state variables:

- Vision: position of the hand in retinal coordinates – extracted from color segmentation.
- Vision: position of target object in retinal coordinates – extracted from color segmentation of the object.
- Proprioception: 3 joint coordinates of the arm (the wrist was fixed in this experiment).
- Proprioception: 5 joint coordinates of the head.
- Proprioception: 3 force components and 3 torque components at the wrist $\left[F_x, F_y, F_z\right]$ and $\left[T_x, T_y, T_z\right]$ respectively.

For the purpose of this experiment, however, we extracted only the initial position of the arm and the instantaneous direction of the target displacement vector at the moment of impact, along with force and torque sensed by the wrist at the same instant (see Figure 6.6).

The goal of this experiment was to learn the effect of a set of pushing/pulling actions from different directions on a toy object. After learning this knowledge could be used to plan an action to bring about a desired effect (e.g. to move an object toward another one). The system learnt a mapping from the initial position of the hand to the direction of target motion. The trajectory was not explicitly planned here as it was determined uniquely by the initial hand-positions; for this reason this information was sufficient to plan the motor action.

Associated with each initial hand position was a direction map (polar histogram) that summarizes the directions that the target moved in when approached from that

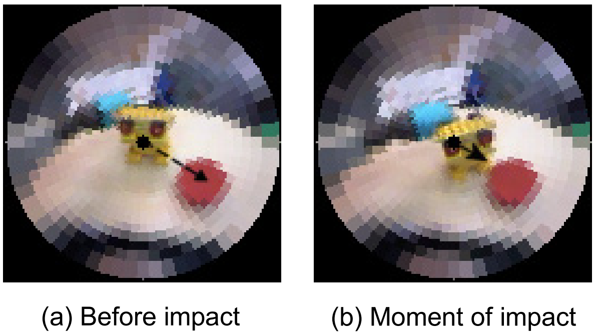

(a) Before impact (b) Moment of impact

Figure 6.6. Relevant visual features. Images are from the robot's point of view and were here remapped to the Cartesian space to facilitate understanding.

position. After each trial the appropriate map was updated with the extracted target displacement vector. The map is a nearest neighbor look-up table where input and output values are accumulated and extracted when required. The map was updated by averaging new samples with the value already present in the nearest position (if any). If the table was empty a new entry was created to store the current sample.

It is important to note that the arm position was preferred as an alternative for other measures that could correlate as well with the target direction. A possible candidate for instance was the direction of approach of the hand to the target. The reason is that in this way it was easier to invert the learning without explicitly solve

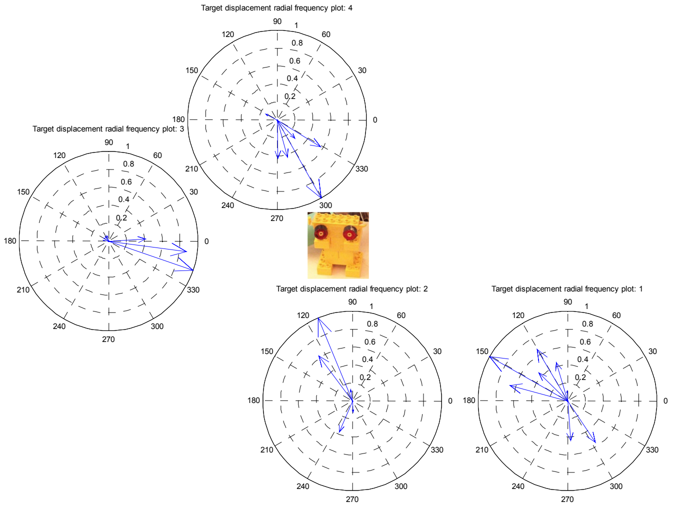


Figure 6.7. The learned target-motion direction maps for each initial hand position.

the inverse kinematics problem (as it would have been the case if the direction of approach of the hand had been used instead). Given a desired direction of motion of the target the robot had just to lookup the positions were the arm should have to be initially placed. The testing of the learning was done by presenting a stationary toy as a new desired target position. The robot's goal was to use the learned maps to correctly pre-position the hand and push the target and move it toward the toy.

6.2.2. Results

Approximately 70 trials distributed evenly across the four initial starting positions were conducted. Figure 6.7 shows the four direction maps learned, one for each initial arm position considered. The maps plot the frequency with which the target moved in a particular direction at the moment of impact. Accordingly, longer radial lines in the plot point toward the most common direction of movement. As an example Figure 6.8 reports the four force maps during a single trial. In this case the polar histograms plot the force measured by the force sensor at the wrist (only two components were reported). As we can see, all maps are sharply tuned toward a dominant direction. In the case of the force, the moment of impact corresponds to a stronger vector opposing the direction of motion of the target (as predicted by Newton's third law).

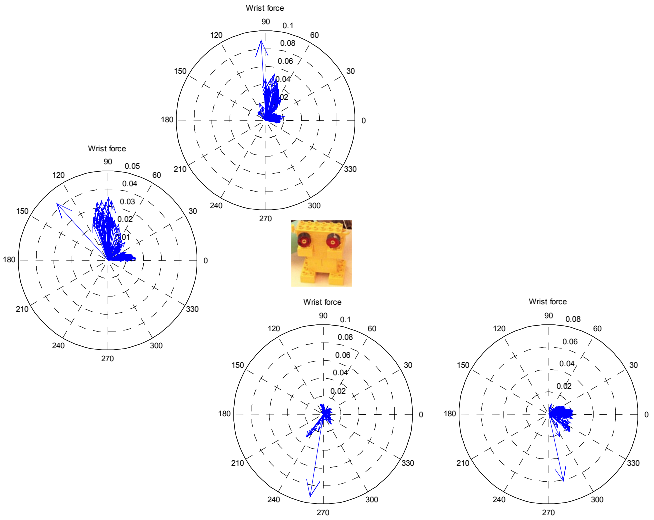


Figure 6.8. The wrist force maps, for each initial hand position (single trial).

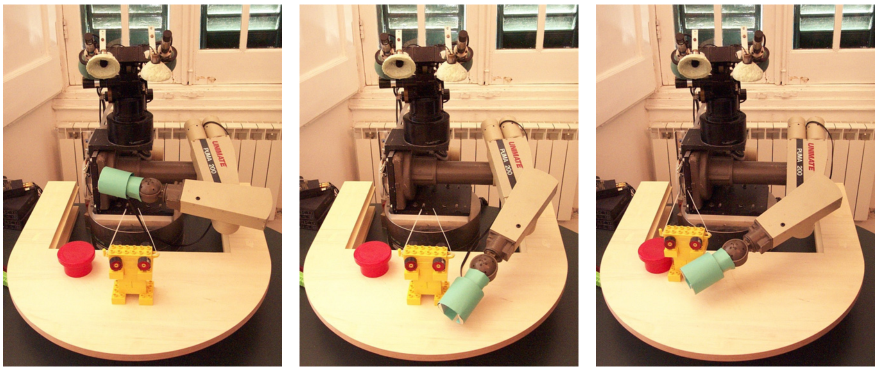
6.2.3. Testing the learned maps

As shown in Figure 6.9 the learned maps were used to drive motor planning in a straightforward way:

- The system was presented with the usual target as before. Another toy was placed nearby (Figure 6.9 (a)), the goal being to push the target toward it. The system foveated the target while locating the new toy in its peripheral vision. The retinal displacement of the toy was used as the desired position r_d.
- The angle θ of this displacement vector was taken to be the direction of desired motion and was used to find the direction map M_θ with the closest matching dominant direction.
- The robot first moved its hand to the hand-position associated with the map M_θ and then began its motion toward the target (Figure 6.9 (b) and (c)). The result was the target being pushed toward the desired direction.

An exemplar sequence of this behavior is reported in Figure 6.9. The round toy defined the new desired position toward which the target must be pushed (a); the robot picked the correct action to fulfill the task based on what it had previously learned (b) and (c).

The performances were tested by taking a quantitative measure of error before and after the learning. The error was the angle between the desired direction of motion and the actual direction the target moved in after being pushed as drawn in Figure 6.6. The control case (baseline) consisted in 54 trials where the goal direction (round toy) and the initial position of the hand were varied randomly among the


(a) The round toy is the new desired target position

(b) The robot positions the arm in preparation for the pushing movement

(c) The end of the movement

Figure 6.9. Using the direction maps to drive goal-directed actions.

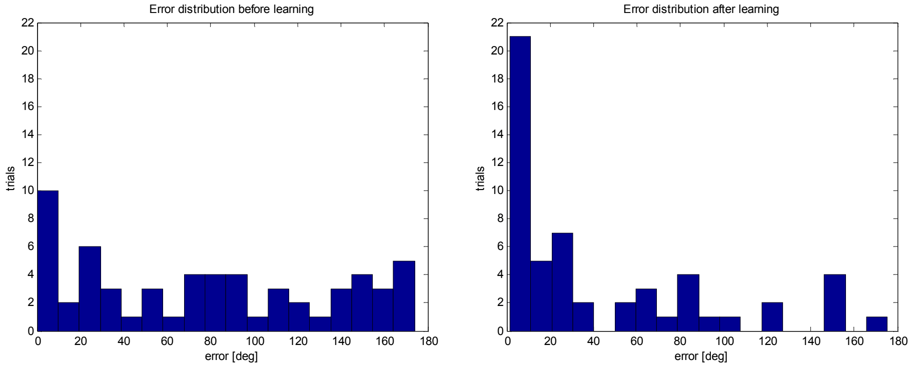


Figure 6.10. Learning performance. Distribution of the angle between desired and actual direction, before (left) and after (right) the learning. Zero degrees indicates no error, whereas 180 degrees indicates maximum error.

four possibilities. The left plot in Figure 6.10 reports the resulting error plot; as expected the distribution is almost uniform across all possible values. The same experiment was carried out this time using the learned maps to position the hand and yielded the error plot of Figure 6.10 (right). The distribution this time is significantly skewed toward an error of 0 degrees meaning that the robot was able to correctly pick the initial hand position from the maps. A few errors close to 180 degrees are still present not as a consequence of an erratic behavior but of an error in measurement. Although the target correctly moved in the desired direction, its motion was perceived as actually happening in the opposite direction. This happened because in these cases the head was already moving while the retinal target displacement was being measured resulting in an apparent backward motion of the target. The same effect is visible in the learned maps as those vectors which are pointing in the direction opposite to the most frequent one (Figure 6.7); this did not affect the behavior of the robot because cancelled out as the result of averaging across several trials. In any case the solution is to integrate the head movement signals to compensate egomotion and extract the "true" motion of the target.

6.2.4. Discussion

The experiment discussed is a first but important step toward "learning to act". Usually the effect of a robot's action on an object is implicitly assumed in the planning, we choose to learn it through play/exploration. The experiment makes some simplifications to test the basic idea. The main directions for improvement are the following. *Moving to a continuous space of hand-positions*. Only four initial hand positions were considered, but a more natural approach would be to pick hand-positions at random during trials and fill the table. To achieve better generalization a neural network could be used instead of the nearest neighbor look-up table. *Use of 3D information*. Although 3D information was implicitly used to reach the fixated target, we made the tacit assumption that the objects were placed on a table; two dimensional visual information was hence sufficient to estimate the direction of target movement. Disparity information could be employed to move toward a more general case. *Interleaving learning with planning*. For simplicity we first had the learning/discovery phase and then the motor planning phase; in principle there is no need for this separation and the learning could easily be carried out during action execution. *Increasing the number of event variables monitored*. Throughout the experiment the same hand speed and target were used. This resulted in the target being displaced roughly by the same amount for each trial. This of course does not happen if the speed of the hand varies or if another object is used. Consider for instance a ball or a bottle, both will tend to roll thus moving for a longer time and with different velocity. This will require a larger set of event features to be included: the size of the target, its shape, the distance moved, the force profile of the hand to mention a few (see for instance (Fitzpatrick et al., 2003; Metta and Fitzpatrick, 2003)).

In conclusion, the work described here is a novel contribution to the area of "event-interpretation" because constraints imposed by the combined modalities of vision, motor, and proprioception may make it easier to interpret certain self-generated events than with vision alone. Furthermore, interpreting self-generated events may be a necessary first step to interpret more complex object-object events (we will come back to this in Chapter 7).

6.3. Learning about objects' shapes

In the previous experiments we showed how the robot could exploit self-generated actions to explore object properties (see also (Fitzpatrick et al., 2003; Natale et al., 2002b)). However, in those cases the robot did not have a dexterous hand and very simple actions were used instead (such as poking and prodding). This section describes a more sophisticated experiment; the goal is to explore the possibility of gathering physical properties of objects from very little prior knowledge and to understand what kind of parameters can be extracted from proprioceptive/tactile feedback. We show that given an extremely simple explorative strategy the robot is able to build a representation of objects that happen to touch its hand. The motor action is defined in advance and elicited by tactile stimulation. The explorative strategy and the hand's passive compliance suffice in starting to acquire structured information about the physical properties of objects drawn from a small set. In particular, we show that the system categorizes objects by exploiting differences on their shape and weight.

6.3.1. Hand calibration

The hand used in this experiment has been illustrated thoroughly in Section 2.1. We give here a short description of the most salient aspects and some details about the calibration of the encoders. The hand consists of five fingers and a total of 16 d.o.f. Six motors control the thumb, the index and a virtual finger made of medium, ring, and small fingers. The latter are linked together with springs, so that if motion of one of the real finger is inhibited the others are free to move. The other joints are also driven by elastic elements to achieve autonomous shape adaptation and to measure the force exerted by each joint.

The hand posture can be computed from Hall-effect (magnetic) encoders from the joints and motor optic encoders. The magnetic encoders are mounted to measure the displacement of each joint due to the passive compliance of the springs. This information, together with the position of the motors obtained from the optic encoders, allows computing the direct kinematics (posture) of the hand. Since magnetic encoders have a non-linear response, they were manually calibrated by moving every joint individually and recording for each position the corresponding output. A cubic function was then fitted to this data. As an example Figure 6.11 shows the input-output characteristic of the encoder mounted in the index finger second phalanx.

Given the length of the links, and once the output of the encoders has been converted into an angular value, it is straightforward to compute the hand

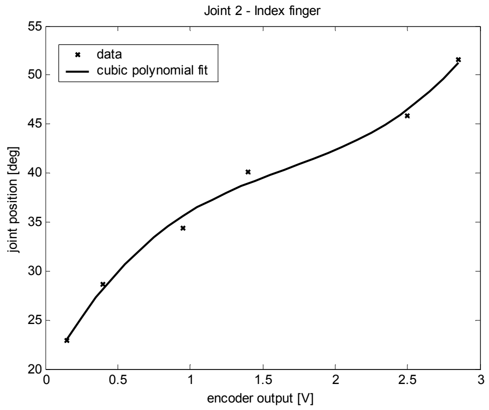


Figure 6.11. Calibration of the magnetic encoders. The output of the encoders was sampled by manually moving each joint ('x' marks). The voltage-position characteristics were linearized by fitting a cubic polynomial function (solid line). The plot reports the characteristic of the index finger second phalanx.

kinematics. Figure 6.12 shows a few exemplar postures of the hand and the corresponding reconstruction by means of a simplified 3D Matlab model.

It is important to notice that the calibration does not need to be exact, it was required in order to convert the encoder feedback (number of ticks and electrical voltage) into a common scale (joint angles).

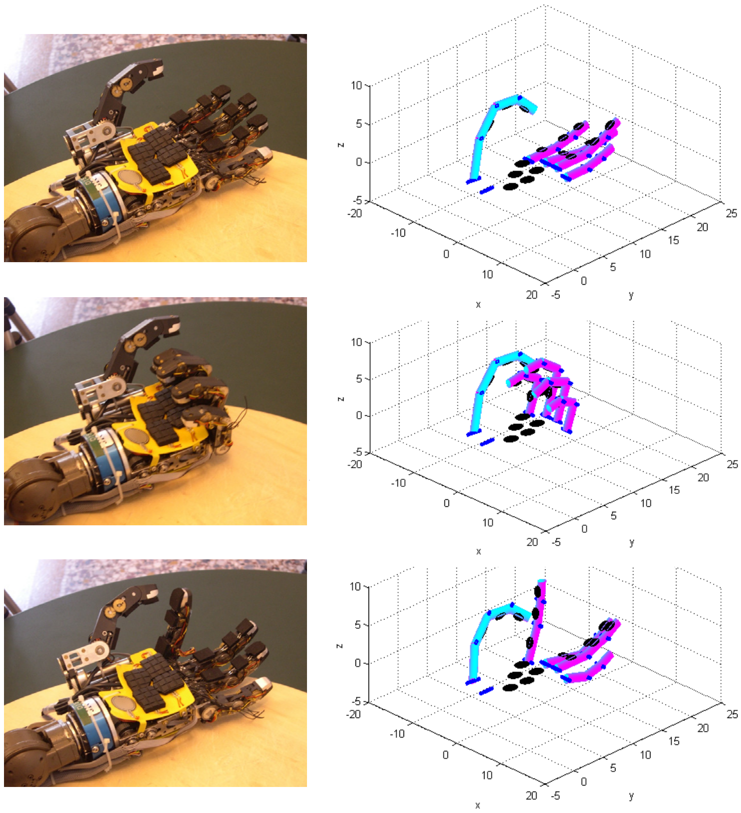


Figure 6.12. Hand postures. Hand pictures (left) and the corresponding Matlab 3D model (right).

6.3.2. Touch-elicited grasp

Newborns reveal a larger variety of finger movements involving both the whole hand and differentiated finger movements. They may be used in exploring objects but not when grasping them. In fact neonates reach for objects but they do not grasp them during this action. The reason is that reaching and grasping are coupled into extension and flexion synergies and, therefore, it is difficult for the child to flex the hand while the arm is extended (von Hofsten, 2003). If an object is put into the hand, however, the neonate might grasp it. Grasping is performed with the whole hand; only much later children start employing relatively differentiated finger movements (8-9 months) (Ronnqvist and von Hofsten, 1994).

In humans the ability to grasp objects at a very early age constitutes an important means of interaction with the environment. For this reason it seemed reasonable to implement the same mechanism in the robot. Similarly to what happens in newborns, tactile stimulation of the palm, initiates a clutching action. As described in Section 2.1 force sensing resistors (FSRs) are mounted on the hand to give the robot tactile feedback. These commercially available sensors exhibit a change in conductance in response to a change of pressure. Although not suitable for precise measurements, their qualitative response can be used to detect touch and measure to some extent the force exerted to the object surface.

Figure 6.13 shows data recorded during a grasp elicited by tactile stimulation; in this case the action is performed with index, medium, ring, and small fingers opposing the palm. At time T1 a soft ball touches the palm (upper trace) eliciting a motor response. The lower trace here reports one of the encoder of the index finger. The finger touches the ball at time T2 and continues pressing it until time T3; the object is held between fingers and palm from T3 to T4. At time T5 it falls off the hand. Although still qualitative these plots show that proprioceptive information can be gathered through this simple grasping action.

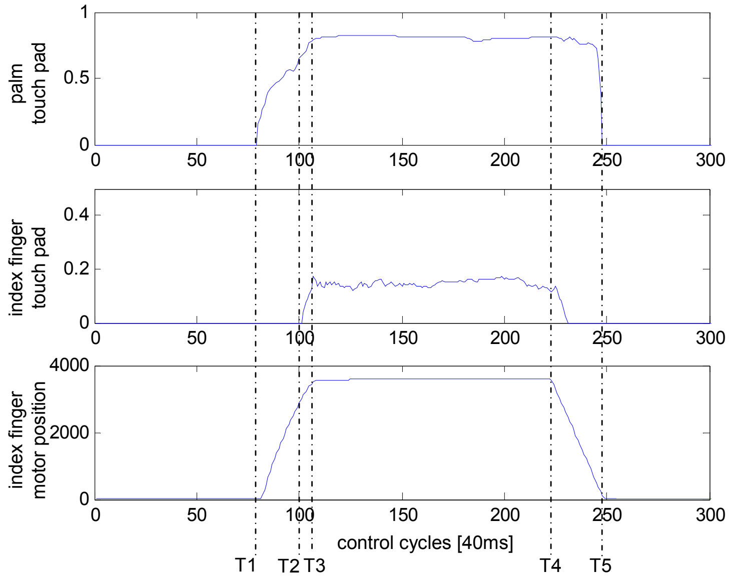


Figure 6.13. Proprioceptive feedback during an exemplar grasping action elicited by tactile stimulation. Upper and middle: palm and index touch sensor signals, normalized arbitrary scales. Below: index finger motor encoder, first phalanx. The scale in this case is encoder ticks; the conversion factor being 0.015 deg/tick (4000 corresponds to 60 degrees)

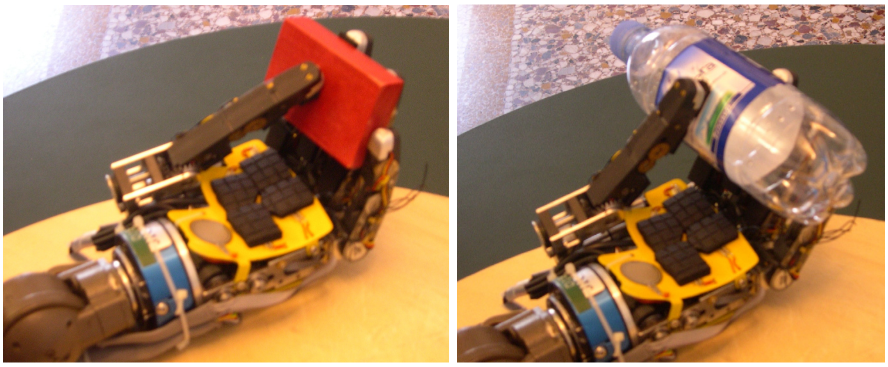


Figure 6.14. Two pictures of the hand grasping a small brick (left) and a bottle (right). The same motor command is used in both cases; the shape of the hand adapts to the object that is being grasped.

6.3.3. Description of the experiments

The goal of these experiments was to explore the possibility to gather physical properties of objects by exploiting stereotyped motor actions like the one described above. Another but important aspect was to understand what kind of physical parameters it could be extracted from proprioceptive/tactile feedback.

In this case the robot did not yet explore the world by actively reaching for objects but grasped toys that either were placed in the palm or touched the fingers. Whenever pressure was applied to the fingers the hand closed by using a predefined motor command (synergy).

The fingers stopped when the maximum torque value – e.g. the motor error in the controller – exceeded a certain threshold for a certain amount of time. Objects in a set were randomly chosen and given to the robot; the robot closed the hand and after a certain amount of time the grasp was released. The motor action did not change from trial to trial; owing to the intrinsic elasticity of the joints, the action of the object on the fingers was exploited to adapt the hand to the target of the grasp (Figure 6.14). For each grasp the posture of the hand reflected the physical size of the object; the vector of joint angles was sent to a self-organizing map (SOM, see Section 8.3 for a brief description of the architecture of this network).

6.3.4. Experiment 1

We employed a set of 6 objects with different shapes (see Figure 6.15 left). The condition where no object was actually placed in the hand was included in the experiment. For each object about 30 grasps were performed, the result of the clustering is reported in Figure 6.15 (right). The network had 225 neurons arranged

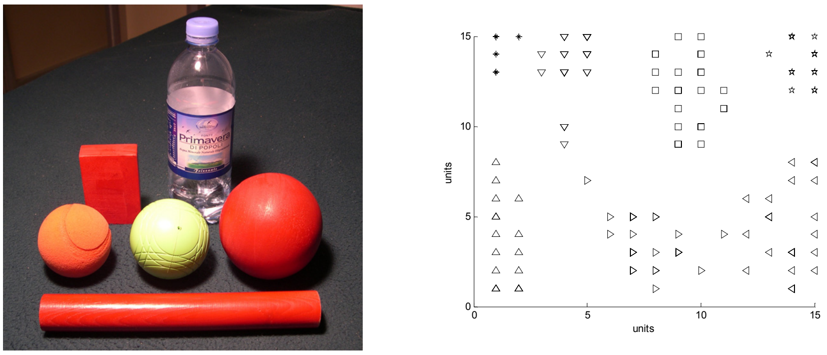


Figure 6.15. Experiment 1. Left: 6 objects were used, a bottle, a brick, a rod, a wooden ball, a small tennis ball made of foam rubber and a small plastic bowl. Right: result of the clustering. 6 classes were formed, one for each object plus one for the no-object condition. The map shows the grid of units (15x15), markers correspond to the neuron which resulted activated the most when a particular input pattern was applied; different markers correspond to different objects. In this case touch sensors were not used.

on a 2-dimensional grid of 15x15 units. For each input pattern we reported the unit which was activated the most on the 15x15 grid; different markers were used for different objects.

The SOM formed 7 clusters, each for a different object plus the no-object condition. Although some objects were quite different in terms of shape, the two small spheres, the plastic bowl and the tennis ball were almost of the same size. These two objects, however, are correctly separated by the network; this is also because the tennis ball is soft especially if compared to the rigid plastic covering of the bowl. As the fingers bent around the soft object they squeezed it a bit, thus facilitating the separation of the clusters. This result is remarkable since recognition, in this case, would have been harder for a putative recognition system based on vision only.

6.3.5. Experiment 2

A second experiment was carried out with two object having identical shape and size, but of different weight. At this purpose we used two plastic small bowls, one of which filled with water to increase its weight (Figure 6.16). The hand was oriented upwards, the palm facing the ceiling, so gravity affected the force exerted by the fingers during grasp. The robot grasped each object about 60 times and the collected information was sent to the SOM. In this case, since only two objects were

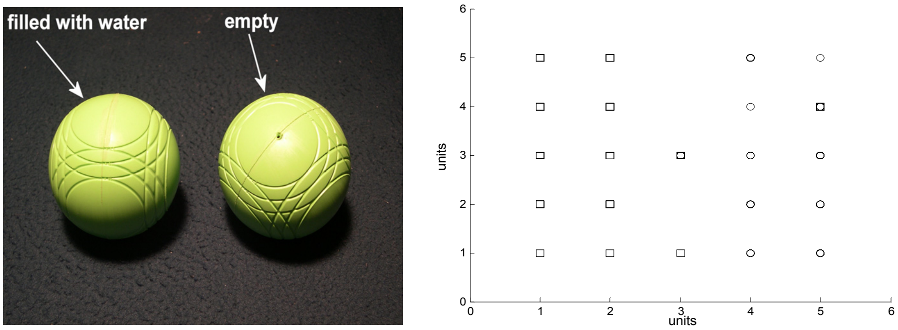


Figure 6.16. Experiment 2. Left: two identical sphere of different weight were used. Right: result of clustering. Markers represent the unit which was activated the most for each input pattern. Different markers correspond to different objects. In this case touch sensors were not used.

used, the network consisted only in 25 units (arranged on a 5x5 grid). The result of the clustering, reported in Figure 6.16, shows that the network was able to separate the two sets as being originated from different objects. As the two spheres had exactly the same size, the capacity of the network to categorize the input patterns was due to the fact that the fingers applied different forces; the hand posture thus implicitly coded objects' weight.

6.3.6. Discussion

We described two experiments where the robot used its hand to explore physical properties of objects drawn from a set. Objects were placed in the palm or between the opposing fingers; the grasping action was elicited by pressure either on the palm or on the fingers. We showed that given the specific design of the hand, and very little prior knowledge, the robot was able to collect certain physical features of the objects it manipulated. A self organizing map was employed to categorize the postural information obtained from grasping. The clustering is not surprising in itself, being just a natural result of the mechanical design of the hand (the elasticity components connecting the joints) and the motor synergy exploited by the robot. Nevertheless the network implicitly coded not only physical features like shape (that in principle could be visually extracted) but also intrinsic properties like weight. Other physical features, like the object's compliance, facilitated clustering. For these reasons, we believe that the results are important; they prove that an active, embodied system can easily solve problems that otherwise would be hard (in the case of the balls of similar size), or even impossible (like in the case of the two identical small bowls having different weight).

The experiment did not employ visual information, but it is not hard to conceive possible ways to include it. Visual parameters like color and shape (central moments) could be extracted from the objects and included in the input vector to the SOM.

Chapter 7

Conclusions

In this thesis we have described a possible approach to the implementation of perceptual abilities in a humanoid robot. This research extends and continues past work on the Babybot (Metta, 2000) by presenting a developmental path that proceeds in three stages, mainly: learning a body map, learning to interact and learning to understand by looking (see Section 1.6 in the Introduction). The previous chapters reported the details of the implementation of aspects related to the first and second stages. Within each chapter specific sections discussed the experimental results. In this section we want to integrate the discussion to show how these results can be placed within a broader picture. We claim that these first two stages are necessary for the third stage to unfold and lead to the realization of a truly cognitive system.

7.1. Motor theories of perception

Traditionally we consider the existence of five senses: touch, sight, hearing, taste and smell. In addition humans are endowed with several receptors which provide information about body motion. The *sense of movement* or, according to Berthoz (Berthoz, 2000), “the sixth sense” results from a combination of muscular proprioception and output from the vestibular system. Motor theories of perception suggest that action and perception are deeply intertwined in the brain; perception is not a mere and passive interpretation of sensory information, but the result of an active internal simulation of actions. Perceiving an object recalls memory of grasping the object, the tactile sensation of the object in our hand and the consequent proprioceptive feedback. Seeing an action activates the very same neural circuitry that would be activated if we were performing the same action ourselves. The interaction between action and perception occurs at different levels. As reported by Viviani and Stucchi (Viviani and Stucchi, 1992) at least three levels can be distinguished: *ecological level*, *active exploration* and *generation of expectations*.

Perception and actions are linked at the *ecological level*, meaning that animals obtain a stable and coherent representation of the world as a collection of sensory changes arising from the movement of the body.

Active exploration establishes a link between action and perception, especially vision and touch. Explorative movements allow an agent to gather sensory information about the environment and to connect together different sensory modalities like, for instance, the sight of an object with the tactile sensation acquired by grasping it. There is a strict connection between this point of view and the *active vision* paradigm (Ballard and Brown, 1992). Ballard and Brown proposed that an artificial vision system can solve visual problems (for example to solve a 3D reconstruction problem) by actively changing its point of view. Indeed actions can simplify visual problems in several ways. By producing a causal connection between action and perception it is possible to focus the attention on events happening on a well defined time window, or to correlates together changes in the visual and proprioceptive percepts. In this thesis a similar solution was adopted to segment the robot's hand from the background and to produce impulsive target motion in the pushing/prodding task. Besides, the agent can autonomously decide to repeat an action on an object if further information is required.

Finally, perception and action are coupled in the *generation of expectation*. A very intriguing question is about how the brain can cope with the relatively high delays in neural circuitry. For instance, visual information can take up to several hundreds of milliseconds before it is actually processed; in case a fast reaction to an external event is required, visual information takes just too long for a proper response to be planned. When a motor command is issued to the body, a copy is sent also to an internal simulator; the latter predicts the consequence of the motor command and anticipates a proper reaction in advance. The brain uses forward models to improve motor control by simulating the effect of motor actions before they are actually executed (see (Miall and Wolpert, 1995) and (Wolpert and Flanagan, 2001) for a review). For instance an internal model of the arm's dynamics could compute an estimation of the future state based on the current state and motor command. This estimation could be used within the sensory motor loop to compensate the delay of the actual feedback. Internal models are thought to be employed by the brain during motion to update a sensory input to plan a delayed response. For example, during a saccade a flashed spot in the retina elicits a consequent saccade toward the correct spatial location, although the retinal slip at the moment the flash occurred would produce a wrong motor response (the double saccade paradigm, (Gilmore and Johnson, 1998)). Another location in the brain where prediction occurs is probably the MST where, during smooth pursuit, egomotion is properly compensated for (Krauzlis and Stone, 1999). In Section 5.2 the robots learnt a forward model of the hand to predict the position of the arm end-point in the visual field. The same model was used to predict the future position of the hand based on the current motor command. Similarly in Section 6.2 the robot built a table to predict the direction an object would move in if pushed/pulled along a given direction (this model is similar to a "motor schema" (Berthoz, 2000)).

7.2. Objects' affordances and action

During our life we discover the use of hundreds of objects. However, not all of them are completely different in the way they are actually handled. For instance glasses are used roughly in the same way irrespectively of their size and shape, the same is true for cups, bottles or books, just to mention a few. However, you can read a book but you may very well decide to use it to kill an annoying mosquito or place it near the door to keep it from closing. To every object we can associate a set of actions; the psychologist J.J.Gibson called these actions *affordances* to represent the actions the object afford. Interestingly, different animals find different affordances depending on their specific sensorimotor repertoire and body. A nice example is reported by Berthoz to explain this concept: "Sometimes opposite properties are interesting to different animals. Thus the degree of firmness (which can be measured as the relationship between pressure and displacement) of the ground allows humans to walk, whereas its friability permits the earthworm to move about" (Berthoz, 2000).

Visual appearance of an object together with contextual information (the last place we have seen it) trigger the selection of the correct affordance to accomplish a particular task. More interesting it is possible to generalize affordances among objects of similar structure. For instance you may guess how to handle a pair of pliers (assuming you have never encountered a pair before) by generalizing the use of a pair of scissors. Indeed both objects close when pressure is applied to their handles.

Affordances play a direct role in action selection, by linking the sight of an object to a specific sequence of motor commands. This process does not necessarily involve a conscious recognition of the object. This last point was extensively proven by psychological as well as physiological experimental results. For instance it was reported the clinical case of a patient who was unable to name objects correctly or to judge which objects might be used together (like a knife and a fork). However the same patient was able to gesture to mimic how the object could be used. Opposite examples were also described about people who were able to recognize and name objects visually, but failed to use them correctly, when asked to. These people did not have any motor problem, but were somehow unable to access the "affordace representation" of objects (for a review see (Humphreys, 2001)).

The representation required to recognize objects is what we have already called the semantic representation, whereas affordances form what we defined a pragmatic representation coding the information required for action ((Jeannerod, 1994) see also Section 1.5). Of course these representations are not completely separated in the brain. The pragmatic representation may help to solve an object identification task. Humprheys (Humphreys, 2001) reported the case of a patient who could not detect a target among other objects if the target was identified by means of its name. However he could solve the task if he was cued with the description of an action, like for instance "find the object to drink from". In somewhat symmetric cases, patients who could not access the pragmatic representation were reported to be able to select the correct action to grasp and use well known objects as opposed to generic ones (e.g. a lipstick versus a small cylinder). In the former case the semantic representation provided information to the pragmatic system (Jeannerod et al., 1995).

These findings are in accordance with the hypothesis that there exist two different visual pathways in the primate's brain (Milner and Goodale, 1995). The dorsal pathway processes information required to solve actions-directed tasks, whereas the ventral pathway is concerned with more abstract concepts, like object identification and recognition. In the first case information related to where an object is and how to handle it would be computed, in the second case the result of the computation would produce the notion of what the object is.

Recent physiological results have identified in the monkey motor cortex an area (F5) where neurons seem to code a pragmatic representation of objects. Area F5 is a premotor area involved mostly in the control of hand movements. Neurons in this area were reported to fire during specific goal-directed actions such as grasping, tearing, holding and manipulating. Murata and coworkers (Gallese et al., 1996; Murata et al., 1997) found that these neurons had also visual properties and that they fired not only when an action was executed (motor-response) but also during object fixation. Thus the neural response had neither an intentional nor an attentional meaning. Besides, the visual response of these neurons was remarkably object-specific and congruent with the type of grasp the neuron coded. They interpreted these results proposing that F5 may contain a sort of motor vocabulary from which appropriate actions are automatically selected from the visual properties of an object (that is a motor representation of object affordances).

7.3. Linking action to perception

Viviani and Stucchi (Viviani and Stucchi, 1992) propose a fourth level at which the interaction between the motor and the perceptual system may occur. They call it a "more abstract level" because it does not involve either the actual execution of movement or a planning stage. They continue by saying that "motor information relevant to the perceiver does not concern some actual specific gesture, but rather procedural knowledge about his own entire repertoire of potential gestures".

This point is particularly evident in speech perception but was extended to the domain of vision by experiments about visual perception of biological motion. Viviani and Stucchi describe an experiment by Beardworth and Bukner who tested the ability of a group of students to recognize each other from recording of their walking. The students were shown a schematic dynamic reconstruction of the walking movement of their schoolmates and were asked to guess the name of the walker. The subject's own walking was included in the test set. Although subjects had much more visual experience about the gait of other people, they made less mistakes in distinguishing their own motion. This suggests that structural information about the motor system is indeed linked at some level to the perceptual mechanism which analyzes visual motion. Other experiments proved that biological constraints on the bodily mechanical structure may play a role in the perception of ambiguous motion. Similarly, geometric and kinematic visual illusions were observed, suggesting that the brain biases motion perception by assuming a biologically-plausible model of the constraints between trajectory and velocity (Viviani and Stucchi, 1992).

Further support to these ideas was recently provided by Rizzolati who discovered neurons in area F5 which fire not only when the monkey performs a grasping action but also when it sees another monkey or the experimenter performing the same action (Gallese et al., 1996). Owing to this peculiar property, these neurons were called *mirror*. Interestingly mirror neurons exhibit kinematic preference for actions respecting biological constraints. In other words, they fire when the action the monkey attends to is performed with hands; they do not fire if very similar actions are made using tools (e.g. if a piece of food is grasped with pliers). The response of mirror neurons results from meaningful interaction of an agent with an object; agent, object and action are all required for the neurons to fire. Response is absent either when the action is not directed toward an object or when there is an object but no action is actually executed.

A possible interpretation of the mirror neurons' function has been given in terms of both motor learning and action interpretation/understanding. In the first case mirror neurons are thought to extract the information essential to describe an action and to link it directly to the motor representation coded by the "canonical" (that is non-mirror) F5 neurons. In the second case the mirror system extends the predictive abilities of the brain to the action performed by others. The observation of an action is mapped into the internal motoric representation identical to the one that is activated when the same action is performed by the subject. In this way the brain can interpret the meaning of the actions it attends to.

According to these observations, the first two developmental stages addressed in this thesis are important precursors to the third one. In other words, learning to act is important not only to guide motor behavior but it may also be a necessary step for event-interpretation when the motor system is not directly involved. Computer vision approaches to the problem of event interpretation have tried to solve this problem in the domain of vision alone. We claim the hypothesis that action generation is an important precursor to event-interpretation. In other words, in order to learn to visually interpret more complicated events in the environment it may be first necessary to learn to act on it. We have shown two aspects of the problem: learning to act and learning about object properties. Together they lead to the development of object affordances and to building a representation similar to the one observed in the monkey mirror system. Of course, further research is required to fill this gap.

Chapter 8

Appendix

8.1. Motion algorithm

The simplest way to compute motion is by taking the difference between the current frame (at time *t*) and the previous one (at time *t-1*). This method, called *frame difference*, is very fast but also sensitive to noise. A more sophisticated solution is to compute a model of the background and compare it to each frame by subtraction. In this case the problem is how to update the model of the background so that, if an object enters into the scene and stops, sooner or later it gets incorporated into the background model and it is no longer considered as being moving (*background modeling*). An easy solution is to update the background by means of a weighted average with the current frame. Mathematically:

$$\begin{cases} B(0)=I(0) \\ B(t)=\alpha I(t)+(1-\alpha)B(t-1) \end{cases} \tag{8.1}$$

where $I(t)$ and $B(t)$ are the current frame and the estimation of the background at time *t* respectively. The parameter α determines the speed at which the background is updated. If $\alpha = 0$ the update is suppressed and the initial guess never changes; on the other hand if $\alpha = 1$ we are back to computing frame difference (the model of the background is completely updated at each frame). All the possibilities in the between are valid, in the experiments reported in this thesis the pretty conservative value of $\alpha = 0.2$ was used. Among the possible choices (e.g. optic-flow) this algorithm was chosen because it can be computed efficiently at high resolution in the log-polar domain. An exemplar sequence is reported below (Figure 8.1).

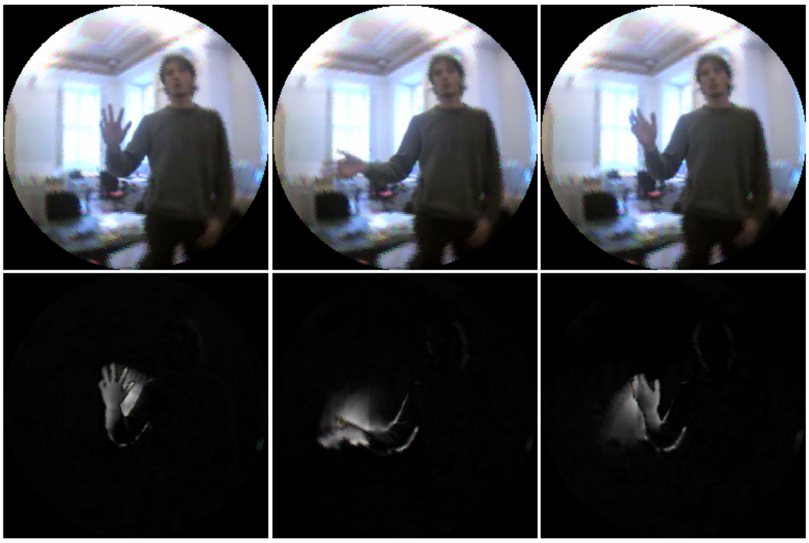


Figure 8.1. Examples of motion. The computation was carried out in the log-polar domain; images here are remapped for simpler understanding.

8.2. Ellipse Fitting

Let us consider an MxN greyscale image $I(x,y)$ which represents the segmentation of an object in the scene. With analogy to mechanics we define the Cartesian moments of the image as:

$$m_{pq} = \sum_{x=1}^{M}\sum_{y=1}^{N} x^p y^q I(x,y) \tag{8.2}$$

where p and q define the order of the moment.

The first two moments can be used to compute the coordinates $(\hat{x},\hat{y})$ of the *center of mass* of the image:

$$\begin{cases} \hat{x} = \dfrac{m_{10}}{m_{oo}} \\ \hat{y} = \dfrac{m_{01}}{m_{oo}} \end{cases} \tag{8.3}$$

Central moments can be computed then as:

$$\mu_{pq} = \sum_{x=1}^{M}\sum_{y=1}^{N} (x-\hat{x})^p (y-\hat{y})^q I(x,y) \tag{8.4}$$

Essentially, central moments are equivalent to Cartesian moments computed after translating the origin of the image to its center of mass; for this reason central moments represent shape features that are translation invariant.

μ_{20} and μ_{02} correspond to the moments of inertia of the image and μ_{11} to the cross moment of inertia. The orientation of the object is the angle of the axis of least inertia. We write the integral of the square of the distance between the points of the object and all possible lines through its center of mass:

$$E = \sum_{x=1}^{M} \sum_{y=1}^{N} r^2 I(x,y) \tag{8.5}$$

where r is the distance from the point (x,y) to the generic line through the center of mass of the object. The line for which this integral is a minimum defines the orientation of the object. Solving the minimization and a performing a few substitution (see (Horn, 1986) , Chapter 3) lead to:

$$\phi = \frac{1}{2}\arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \tag{8.6}$$

The object ellipse is defined as the ellipse whose least and greatest moments of inertia equal those of the object. The equation of a generic ellipse with center at $(0,0)$ is:

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \tag{8.7}$$

whose semimajor and semiminor axes (a,b) are related to the central moments by

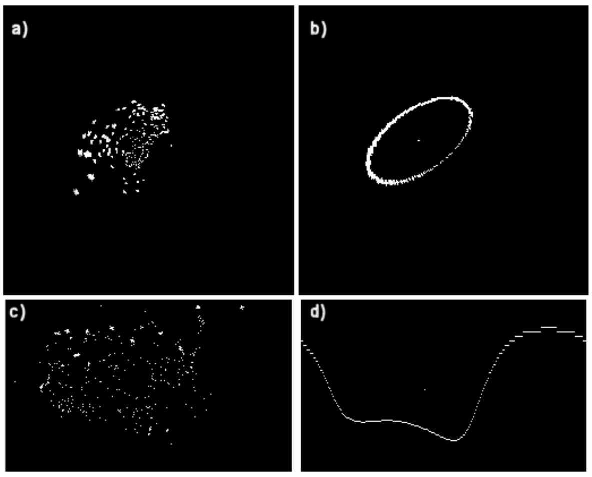


Figure 8.2. An example of ellipse fit. The original image (b) and the ellipse object estimated by means of the algorithm described in the text (d). (a) and (b) depict the same images remapped to Cartesian space. Note that the processing is carried out on (b).

the following equation:

$$\left(a,b\right)=\sqrt{\frac{\mu_{20}+\mu_{02}\pm\sqrt{\left(\mu_{20}+\mu_{02}\right)^2+4\mu_{11}}}{\mu_{00}}} \tag{8.8}$$

Another representation of a generic ellipse can be expressed by means of a quadratic equation:

$$\underline{x}A\underline{x}^T=\left[x \quad y\right]\begin{bmatrix}\alpha_{11} & \alpha_{12}\\ \alpha_{12} & \alpha_{22}\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=1 \tag{8.9}$$

whose coefficients are related to the axis and orientation by:

$$\begin{aligned}\alpha_{11}&=\left(\frac{\cos\phi}{a}\right)^2+\left(\frac{\sin\phi}{b}\right)^2\\ \alpha_{12}&=\left(\frac{1}{a^2}-\frac{1}{b^2}\right)^2\sin\phi\cos\phi\\ \alpha_{22}&=\left(\frac{\sin\phi}{a}\right)^2+\left(\frac{\cos\phi}{b}\right)^2\end{aligned} \tag{8.10}$$

The matrix A is the ellipse shape representation of $I(x,y)$.

To compute Cartesian and central moments in a log-polar image $I\left(\eta,\xi\right)$, equations (8.2) and (8.4) have to be expressed in terms of $\left(\eta,\xi\right)$. By changing coordinate system we have:

$$m_{pq}=\sum_{\eta=1}^{M}\sum_{\xi=1}^{N}x\left(\eta,\xi\right)^p\,y\left(\eta,\xi\right)^q\,I\left(\eta,\xi\right)\left|J\left(\eta,\xi\right)\right| \tag{8.11}$$

and

$$\mu_{pq}=\sum_{\eta=1}^{M}\sum_{\xi=1}^{N}\left(x-\hat{x}\right)^p\left(y-\hat{y}\right)^q\,I\left(\eta,\xi\right)\left|J\left(\eta,\xi\right)\right| \tag{8.12}$$

where $x\left(\eta,\xi\right)$ and $y\left(\eta,\xi\right)$ can be computed from the log-polar equations (see Section 4.1) and $\left|J\left(\eta,\xi\right)\right|$ is the determinant of the jacobian of the transformation, or:

$$J\left(\eta,\xi\right)=\begin{bmatrix}\dfrac{\partial x}{\partial\eta} & \dfrac{\partial x}{\partial\xi}\\ \dfrac{\partial y}{\partial\eta} & \dfrac{\partial y}{\partial\xi}\end{bmatrix} \tag{8.13}$$

8.3. Self organizing feature map (SOM)

Let us assume we have a set of samples in a given space $\mathbb{R}^n$; we want to project these points onto a discrete space $\mathbb{R}^m \quad m \le n$ in such a way so that the output preserves the topology of the input: that is, we want those points that are neighbors in the source space to map to the same or close points in the output space.

A Kohonen Self Organizing Map (SOM) solves this task. It is a fully connected single layer linear network where each i^{th} unit computes its activation as:

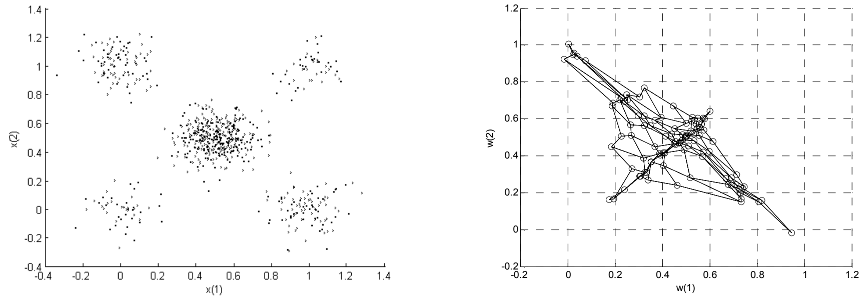
$$net_i = \sum_{k=1}^{m} x_k \cdot w_k \tag{8.14}$$

where $\underline{x} \in \mathbb{R}^m$ is the current input to the network and $\underline{w} \in \mathbb{R}^m$ is the unit's vector of the weights.

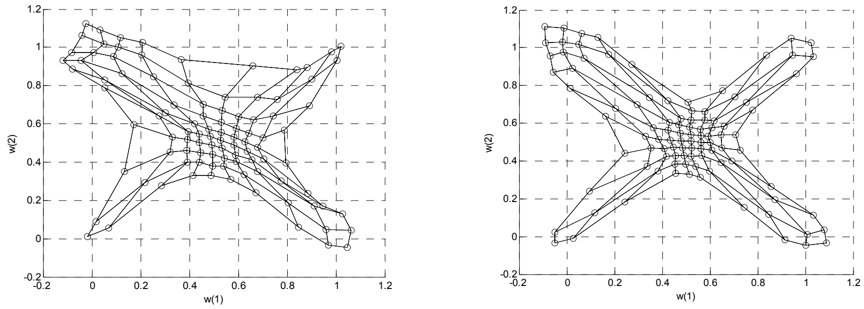
Units may be arranged in one or higher dimensional space (although rarely more than two). In this space each unit has a set of neighbors to which it is connected (for example in a one-dimensional SOM each unit has two neighbors, the preceding and the following one). The learning rule to update the weights of the i^{th} unit is the following:

$$\underline{w}_i(t+1) = \underline{w}_i(t) + \eta(t) \cdot \left[\underline{x}(t) - \underline{w}_i(t)\right] \tag{8.15}$$

where t is the iteration number, $\underline{x}(t)$ is the current input and $\eta(t)$ is the learning rate. The former depends on t as it is usually reduced during learning. Not all the units get their weights updated, a soft competition rule is followed: if i^* is the "winning" unit (that is the unit whose activation is the highest) only the weights of neighbor units are changed. A possible strategy is to vary the amount by which each unit is updated according to a decreasing exponential rule. In other words the weights of the winner and its neighbors are attracted towards the input pattern. As a result neighbors units code similar (closer) input patterns. In the case of a 2-dimentional SOM the output of the network can be represented as a 2D grid whose points correspond to the neurons. The output of the network is, for a particular input, the unit that gets activated the most. A further nice feature of the SOM is that it implicitly codes the probability of the source space ($p(\underline{x})$); in fact after the learning more units are attracted towards regions of higher probability. Figure 8.3 shows an example where a SOM is used to map a set of points from a 2D source space.



a) source patterns b) weights (1 epoch)



c) weights (10 epochs) d) weights (500 epochs)

Figure 8.3. A SOM is trained on a set of input points. The source patterns consisted in 5 sets whose points were generated from Gaussian distribution having different mean and same variance (a). The sets have also different probabilities as it can be observed in the graph. The weights of the network are reported during learning after 1, 10 and 500 epochs; on the same plots the connections between the units are also drawn (b), (c) and (d). The plots show how the SOM learns the topology of the input. Notice that more units are attracted towards the regions with higher probability (compare (a) with (d)).

List of figures

Table 1-A. Goals of Artificial Intelligence (adapted from (Russell and Norvig, 1995)). 15

Table 1-B. A possible developmental path for the robot. Cells represent successive motor and perceptual competencies acquired during development. The table details for each phase the goal of the learning, the goal of the system and the link that is established between action and perception. The rightmost column reports the delay occurring between action and perception; the latter is related to the time course of development and the stages as reported in Figure 1.1. Shaded cells are topics that in part were addressed in this thesis....29

Figure 1.1. The development of the robot takes place by following this simplified schema. The first stage involves learning about the robot's own body (limb size and dynamics). The second one concerns learning to interact with the environment whereas the third (hypothetical) stage is devoted to learning event interpretation. Stages are not completely separated as they evolve together; consequent stages rely on the competencies acquired in the previous ones (compare to Table 1-B). 27

Figure 2.1. The robotic setup, the Babybot. 32

Figure 2.2. Mechanical coupling between phalanges. The second phalanx of the index finger is directly actuated by a motor. Two gears transmit the motion to the third phalange. The movement is respectively of 90 and 45 degrees. 32

Figure 2.3. Elastic coupling. (a) and (b) show two different postures of the hand. Note however that in both cases the position of the motor shafts is the same. In (b) the intrinsic compliance of the medium finger allows the hand to adapt to the shape of the object....33

Figure 2.4. The inertial sensor of the Babybot developed at LIRA-Lab. It consists of three mono-axial sensors arranged along three orthogonal axes. 34

Figure 2.5. Tactile sensors. 17 Sensors have been placed: five in the palm, three on each finger apart the little finger. In this picture the sensors in the thumb are hidden. The short blue cylinder that links the PUMA wrist to the hand is the JR3 force sensor. 34

Figure 2.6. Hardware architecture. It consists of two separated switched networks. One network is dedicated to visual processing; the other to control signals and other data. The nodes are computers (from 2.4Ghz PIV to 750 Ghz PIII) connected either to one or both networks. The machine equipped with frame grabbers acquires the images and broadcast them across network 1. Nodes with motor cards drive the robot and receive position feedback (head, arm and head); in some cases supplementary cards may be used (e.g. in the case of the hand the acquisition of the magnetic encoders and the tactile sensors requires additional A/D converters). Other machines can be connected at will to perform other processing (e.g. learning). The server provides access to a shared file system and runs the name service. From the server console it is possible to launch control scripts, which remotely execute new processes and manage the running ones (this includes termination, connection and disconnection)....................................36

Figure 2.7. The YARP communication architecture, simplified schema. Five ports are represented in a hypothetical configuration; each port consists in a command receiver (dark gray) and one or more portlets (light gray). Portlets are active objects instantiated to handle connections: the TCP port requires a portlet for each connection whereas the Multicast port instantiates only one. Notice also that input ports may receive from different protocols..39

Figure 2.8. YARP libraries: dependence chart. Dark gray represents third part software and libraries; light gray are libraries and software modules that are part of YARP. All blocks (excepted virtual device drivers) use ACE to make the software platform independent. Details about specific blocks are reported in the text. 41

Figure 2.9. The structure of a control class for a generic device. The virtual device driver provides a generic interface to the hardware. Idiosyncrasies of the particular setup (wiring of the robot, initialization procedure) are implemented in a separate class (“Local definition class”). 42

Figure 2.10: A module for learning sensorimotor coordination. 44

Figure 2.11. The combination of learning modules in a hypothetical subsumption arrangement...45

Figure 3.1. Muscle tension-length characteristic. Left plot: muscle stiffness varies according to neural activity (adapted from (Ghez, 1991)). Right plot: Length-tension curves measured in the cat’s soleus muscle at different activation rates. The initial part of the curves is linear, stiffness increases with the activation (adapted from (Mussa-Ivaldi and Bizzi, 1993))...49

Figure 3.2. CAD model of the realized prototype. See text for a detailed description. 52

Figure 3.3. Detail of a single elastic actuator. A lead screw controlled by a DC motor varies the number of coils of the springs. The spring stiffness changes as the motor rotates. A strain gauge provides force feedback (F) whereas digital encoders (not shown) measure the position of the motor shaft as well as the position of the rotary joint. 53

Figure 3.4. Left: two actuators linked together. Tendons connect two springs to the joint in push-pull configuration. Right: the effect of the two springs is equivalent to a single spring whose stiffness and resting length depends on the number of active coils. 54

Figure 3.5. Force-length curves of a single actuator. The number of active coils varies from 10 (top-left) to 5 (bottom-right). In each plot a line was fitted on the data for F>0 and its angular coefficient taken as an estimation of the stiffness. These measures are reported in the last plot which reports the variation of the stiffness with respect to the number of active coils. 55

Figure 3.6. Force-length characteristics for the linked actuators. Abscissa represents the position of the link (roughly in the range +/- 50 degrees), ordinate reports the net restoring force acting on the link (+/- 500 N). Plots are reported for different values of active coils from 9 (top left) to 5.5 (bottom right). The stiffness follows an inverse proportional law with respect to the number of coils of the springs. 56

Figure 3.7. Open-loop control schema. The control board computes the PID control law to perform a desired motion. The input to the system is the number of active coils (u1, u2). 57

Figure 3.8. Left: force-displacement plane. Dashed lines represent force-length characteristics of the left springs. Solid lines: right spring. Number of turns range from 1 to 15 (n1 and n2). The points were the lines intersect correspond to the equilibrium points for the joint; the angle between left and right spring characteristics implicitly define the stiffness. Right: an external force moves the system from the equilibrium position of +/- 7 degrees. Two configurations are depicted, low stiffness (A) and high stiffness (B). The restoring force exerted by the two springs together can be graphically computed by measuring the distance between solid and dashed lines. It is easy to verify that in (A) the restoring force is stronger than in (B), although the displacement is the same. 57

Figure 3.9. Three exemplar trajectories. In all three cases the system moves from an initial position of -20 degrees to a final position of 20 degrees. As far as the stiffness of the joint is concerned, the three trajectories are quite different. In (1) the stiffness is kept constant at a relatively high value; in (2) the stiffness is initially low and gets constantly increased whereas in (3) the opposite occurs…58

Figure 3.10. Trajectory 1. The system moves from a position of about -20° to +20°. The stiffness in this case is maintained constant at a relatively high value. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors. Note that there is a residual current whenever the controller cannot reduce the error to zero due to the friction.........60

Figure 3.11. Trajectory 2. The system moves from a position of about -20° to +20° while increasing the stiffness. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors. Note that there is a residual current whenever the controller cannot reduce the error to zero due to the friction.........61

Figure 3.12. Trajectory 3. The system moves from a position of about -20° to +20° while reducing the stiffness. Top: position and stiffness of the joint (measured by the encoder and computed from n1 and n2). Middle: time course of n1 and n2 measured from the motor encoders. Bottom: electric current absorbed by the motors.........62

Figure 3.13. Comparison between trajectory 2 (dashed line) and trajectory 3 (solid line) (see also Figure 3.9). The plot represents the time course of the position; the curves are shifted to make zero the initial position and facilitate comparison. Note that trajectory 2 is more accurate because the higher stiffness contributes to reduce the final error due to the weight of the link (the actuator is mounted vertically, the weight opposing the motion). 63

Figure 4.1. Cones density in the human retina decreases quickly as we approach the periphery (a). Dark gray is temporal retina; light gray is nasal retina (adapted from (Packer and Williams, 2003)). Retinotopic map in the striate cortex of the squirrel monkey (c) and (d). The mapping is illustrated by the system of rings and rays superimposed on the retina and plotted as they warp on the cortex; rays and circles map to horizontal and vertical straight lines. Notice also how most of the cortex is devoted to the central part of the retina; in particular half of the cortex represents rings from 0 to 8 degrees (adapted from (Adams and Horton, 2002)). Compare these pictures with Figure 4.2 and Figure 4.3. 66

Figure 4.2. A simplified explanation of the log-polar mapping. The image is divided in concentric circles which are uniformly sampled and arranged along the rows of the logpolar image. The outermost and innermost circles are placed in the last and first rows respectively (different shades of gray are used for different radii). The darkest region at the center is the fovea, which is divided in triangles and reported in the cortical plane (Berton, 2003). 67

Figure 4.3. An example of log-polar mapping. Left: the original image. Right: the resulting log-polar image in the cortical plane. The particular image stresses the salient characteristics of the transformation: the circular arrangement of the petals is straightened in the log-polar domain, more than half of the pixels in the cortical plane is used to represent the central part of the flower. 68

Figure 4.4. Head control schema. Images from the left camera are sent to the tracker which extracts the position of the target in image coordinates; by means of the inverse Jacobian this information is converted into motor commands for the eyes. The block indicated by "vergence" computes the disparity index; the latter is then multiplied by a constant proportional gain and added to the right eye motor command. Information from the inertial sensor is used to compute the VOR component. VOR, vergence and version are summed together and issued to the low-level controller which computes the torque to drive the motor. From the encoder feedback the d.o.f. of the neck are coordinated with the eyes as described in Section 4.6. The blocks realizing saccades are not shown. 71

Figure 4.5. Vergence control: a toy is moved to follow a straight line toward the robot. Left: the trajectory of the fixation point in the 3D space. The fixation points is plotted with '+' every 10 frames (frame rate was 25 Hz), whereas the simple 3D model represents the robot: circles are the joints, solid lines correspond to the links (the arm did not move in this experiment). 1 and 2 mark the beginning and the end of the trajectory. Right: images from the right and left cameras at the same instants (L1-R1 and L2-R2 respectively). Notice that the car is maintained at the center of the visual fields of both eyes. 73

Figure 4.6. Tracking (vergence and version). A toy is moved while the robot tracks it. The left plot shows the trajectory of the fixation point during the experiment (top view); '+' marks correspond to the fixation point every 10 frames (frame rate was 25Hz). The 3D model sketches the robot: circles are the joints, solid lines correspond to the links (the arm did not move in this experiment). Initial and final points of the trajectory are marked with 1 and 2. Images at the same instants are reported on the right (L1-R1 and L2-R2 respectively). Notice that the car is maintained centered within the visual filed of the two eyes 74

Figure 5.1. Arm control schema. It consists of two loops; a feedback inner loop employs a PD controller to achieve a desired joint angle. The block marked with G computes the gravity load term which is fed-forward to the control board. ...76

Figure 5.2. Arm gravity compensation, joint 1 (shoulder). Left: gravity load as a function of the arm joints; actual samples (circles) and estimated function (mesh) after 200 trial run. Ordinate uses arbitrary scale (control board digital output). Right: error trend during learning, actual data (dashed line) and moving window average over 10 trials (solid line). After 25 trials the gravity compensation is activated and the error decreases quickly. 78

Figure 5.3. Arm gravity compensation, joint 2 (arm). Conventions as in Figure 5.1. In this case the improvement is less remarkable as the gravity load is lower compared to joint 1. 78

Figure 5.4. Gravity compensation, joint 3 (forearm). Conventions as in Figure 5.2. In this case the improvement is less remarkable as the gravity load is lower compared to joint 1. 79

Figure 5.5. Examples of correlated (a) and uncorrelated motion (b). The picture plots motion in the images for two pixels (a1 and b1) and the result of the zero-crossing algorithm (a2 and b2). Arm motion for the wrist joint is reported below (a3 and b3) together with the result of the zero-crossing algorithm (a4 and b4). By comparing a2 to a4 and b2 to b4 it is clear that (a) corresponds to the pixel that belongs to the hand. Abscissas are arbitrary scales (normalized values)......81

Figure 5.6. Hand segmentation schema. 82

Figure 5.7. Hand color histograms during learning (top-view). The Hue-Saturation space is divided in 10x10 bins to sample the interval from 0 to 255; different shades of gray are used to represent the probability: from light gray (0.0), to dark gray (1.0). Histograms are normalized with respect to the maximum. As the learning progresses the contribution of the background cancels out and the histogram gets skewed toward the color of the hand (about (30, 150)). 83

Figure 5.8. Example of hand detection and segmentation (1). Top sequence, from left to right: original image, result of the detection algorithm, low-pass filtering, ellipse fitting and segmentation. Bottom: the same sequence in the logpolar domain. 84

Figure 5.9. Hand position predictor schema. 85

Figure 5.10. Hand shape predictor schema. 85

Figure 5.11. Testing the learning performance. As soon as a new sample is available it is compared to the current output of the neural network. As the learning progresses new samples are closer to the output of the network, meaning that the prediction has improved. Ordinate reports the root square error in the image plane in pixels (dashed line is original data, solid line is moving window average over 10 samples). 86

Figure 5.12. Examples of segmentations. As the learning progresses (from top-left to bottom-right) the robot starts tracking the hand; as a result the hand is more likely to appear in the center of the visual field....87

Figure 5.13. Forearm segmentation. The algorithm used for the hand localization could be replicated for other body parts. In this case three examples are reported for the segmentation of the forearm. 89

Figure 5.14. Hand localization (1). Frames from a 20 second sequence of the robot tracking the hand (each frame is taken at 1 second interval). The cross represents the position of the hand estimated from the arm posture, the ellipse plots its approximate shape; the gaze of the robot is controlled to maintain fixation on the cross. 91

Figure 5.15. Hand prediction. In this 20 second sequence the head of the robot does not move (notice the position of the toys in each frame). At frames 2, 5, 9, 13 and 17 a new motor command is issued; for each of these commands the map predicted the region of the image the hand will be at the end of the movement (frames 4, 8, 12 and 20 respectively). 92

Figure 5.16. Hand localization (2). The color histogram is used to check if the hand is actually visible or not. Both arm and head are stationary in this sequence; different objects are introduced to cover the hand. Cross and ellipse are depicted in black when the hand gets completely occluded (frames 5, 8, 10, 15, 18). 93

Figure 6.1. Testing the learning performance. Whenever a new sample is acquired it is also used to address the network The output of the network is compared to the same input to estimate the ability of the network to predict new samples (see text). The error trend shows that the training is consistent....97

Figure 6.2. Arm control schema (reaching). The fixation point is computed from the current head posture (joint angles). The motor-to-motor map converts it into the desired command for the arm; the trajectory generator produces a set of “smooth” commands which are sent to the low-level controller (the latter is described in Section 5.1)....98

Figure 6.3. Reaching sequence (1). The robot tracks the bottle and then reaches for it. Frames are taken at 1 second each, the sequence lasts 6 seconds....99

Figure 6.4. Reaching sequence (2). The robot tracks the bottle and then reaches for it. Frames are taken at 1 second each, the sequence lasts 6 seconds...........................99

Figure 6.5. Details of the experiment. (a) the four initial positions for the arm. (b) and (c) report a pushing trial run. 101

Figure 6.6. Relevant visual features. Images are from the robot's point of view and were here remapped to the Cartesian space to facilitate understanding. 102

Figure 6.7. The learned target-motion direction maps for each initial hand position. 102

Figure 6.8. The wrist force maps, for each initial hand position (single trial). 103

Figure 6.9. Using the direction maps to drive goal-directed actions. 104

Figure 6.10. Learning performance. Distribution of the angle between desired and actual direction, before (left) and after (right) the learning. Zero degrees indicates no error, whereas 180 degrees indicates maximum error....................105

Figure 6.11. Calibration of the magnetic encoders. The output of the encoders was sampled by manually moving each joint ('x' marks). The voltage-position characteristics were linearized by fitting a cubic polynomial function (solid line). The plot reports the characteristic of the index finger second phalanx...............107

Figure 6.12. Hand postures. Hand pictures (left) and the corresponding Matlab 3D model (right)...108

Figure 6.13. Proprioceptive feedback during an exemplar grasping action elicited by tactile stimulation. Upper and middle: palm and index touch sensor signals, normalized arbitrary scales. Below: index finger motor encoder, first phalanx. The scale in this case is encoder ticks; the conversion factor being 0.015 deg/tick (4000 corresponds to 60 degrees)...109

Figure 6.14. Two pictures of the hand grasping a small brick (left) and a bottle (right). The same motor command is used in both cases; the shape of the hand adapts to the object that is being grasped...110

Figure 6.15. Experiment 1. Left: 6 objects were used, a bottle, a brick, a rod, a wooden ball, a small tennis ball made of foam rubber and a small plastic bowl. Right: result of the clustering. 6 classes were formed, one for each object plus one for the no-object condition. The map shows the grid of units (15x15), markers correspond to the neuron which resulted activated the most when a particular input pattern was applied; different markers correspond to different objects. In this case touch sensors were not used...111

Figure 6.16. Experiment 2. Left: two identical sphere of different weight were used. Right: result of clustering. Markers represent the unit which was activated the most for each input pattern. Different markers correspond to different objects. In this case touch sensors were not used..112

Figure 8.1. Examples of motion. The computation was carried out in the log-polar domain; images here are remapped for simpler understanding..........................122

Figure 8.2. An example of ellipse fit. The original image (b) and the ellipse object estimated by means of the algorithm described in the text (d). (a) and (b) depict the same images remapped to Cartesian. Note that the processing is carried out on (b)...123

Figure 8.3. A SOM is trained on a set of input points. The source patterns consisted in 5 sets whose points were generated from Gaussian distribution having different mean and same variance (a). The sets have also different probabilities as it can be observed in the graph. The weights of the network are reported during learning after 1, 10 and 500 epochs; on the same plots the connections between the units are also drawn (b), (c) and (d). The plots show how the SOM learns the topology of the input. Notice that more units are attracted towards the regions with higher probability (compare (a) with (d))..126

References

Adams, D.L. and Horton, J.C., 2002. Shadows Cast by Retinal Blood Vessels Mapped in Primary Visual Cortex. Science, 298.

Anguita, D., Parodi, G. and Zunino, R., 1994. An Efficient Implementation of BP on RISC-based Workstations. Neurocomputing, 6: 57-65.

Arsenio, A., Fitzpatrick, P., Charles, K.C. and Metta, G., 2003. The Whole World in Your Hand: Active and Interactive Segmentation, Third International Workshop on Eigenetic Robotics. Lund University Cognitive Studies, Boston, USA, pp. 49-56.

Asada, M., MacDorman, F., Karl, Ishiguro, H. and Kuniyoshi, Y., 2001. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. Robotics and Autonomous Systems, 37: 185-193.

Ballard, D.H. and Brown, C.M., 1992. Principles of Animate Vision. Computer Vision Graphics and Image Processing, 56(1): 3-21.

Barth, F., G., Humphrey, J.A.C. and Secomb, T.W. (Editors), 2003. Sensors and Sensing in Biology and Engineering. Springer-Verlag Wien New York.

Beer, R., D., 2000. Dynamical approaches to cognitive science. Trends in Cognitive Sciences, 4(3): 91-99.

Beer, R., D., Chiel, H., J., Quinn, R., D. and Ritzmann, R.E., 1998. Bioribotic Approaches to the Study of Motor Systems. Current Opinion in Neurobiology, 8: 777-782.

Bernardino, A., 2004. Binocular Head Control with Foveal Vision: Methods and Applications. Ph.D Thesis, Lisbon, 167 pp.

Bernardino, A. and Santos-Victor, J., 2002. A Binocular Stereo Algorithm for Log-Polar Foveated Systems, Second International Workshop, BMCV 2002. Lecture Notes in Computer Science. Springer, Tubingen, Germany, pp. 127-136.

Berthouze, L. and Kuniyoshi, Y., 1998. Emergence and Categorization of Coordinated Visual Behavior Through Embodied Interaction. Machine Learning, 31: 187-200.

Berthoz, A., 2000. The Brain Sense of Movement. Perspectives in Cognitive Neuroscience. Harvard University Press, Cambridge, MA, 337 pp.

Berton, F., 2003. Everything You Always Wanted to Know About Log Polar (but were afraid to ask), LIRA-Lab, DIST, Università di Genova, Genova.

Bizzi, E. and Mussa-Ivaldi, F.A., 1993. Geometrical and Mechanical Issues in Movement Planning and Control. In: M. Posner, I. (Editor), Foundations of Cognitive Science. MIT Press, Cambridge, MA, 769-792.

Bizzi, E., Mussa-Ivaldi, F.A. and Giszter, S.F., 1991. Computations Underlying the Execution of Movement: A Biological Perspective. Science, 253: 287-291.

Blauert, J., 1983. Spatial Hearing: the Psychophysics of Human Sound Localization. MIT Press, Cambridge, 494 pp.

Brooks, R., 1990. Elephants Don't Play Chess. Robotics and Autonomous Systems, 6: 3-15.

Brooks, R., 1991. How to build complete creatures rather than isolated cognitive simulators. Architectures for Intelligence. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 225-240 pp.

Brooks, R. et al., 1998. Alternate Essences of Intelligence, Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin.

Brooks, R.A., Brezeal, C.L., Marjanovic, M. and Scassellati, B., 1999. The COG project: Building a Humanoid Robot, Lecture Notes in Computer Science. Elsevier, 52-87.

Capurro, C., Panerai, F. and Sandini, G., 1995. Space Variant Vision for an Active Camera Mount, SPIE AereoSense95, Orlando, Florida.

Capurro, C., Panerai, F. and Sandini, G., 1997. Dynamic Vergence using Log-Polar Images. International Journal of Compuer Vision, 24(1): 79-94.

Carpenter, R.H.S., 1988. Movements of the Eyes. Pion Limited, London.

Cheng, G. and Kuniyoshi, Y., 2000. Complex Continuous Meaningful Humanoid Interaction: A Multi Sensory-Cue Base Approach, IEEE International Conference on Robotics and Automation, ICRA 2000, San Francisco, CA, pp. 2235-2242.

Chiel, H., J. and Beer, R., D., 1997. The brain has a body: adaptive behavior emerges from interactions of nervous sysyem, body and environment. Trends in Neuroscience, 20(12): 553-557.

Dario, P., Sandini, G. and Aebischer, P. (Editors), 1993. Robots and Biological Systems: Towards a New Bionics ? Computer and Systems Science, 102. Springer-Verlag, Berlin, 786 pp.

Desmurget, M., Pélisson, D., Rossetti, Y. and Prablanc, C., 1998. From Eye to Hand: Planning Goal-directed Movements. Neuroscience and Behavioral Reviews, 22(6): 761-788.

Fadiga, L., Fogassi, L., Gallese, V. and Rizzolatti, G., 2000. Visuomotor neurons: ambiguity of the discharge or 'motor' perception? Internation Journal of Psychophysiology, 35(2-3): 165-177.

Fitzpatrick, P., 2003. From First Contact to Close Encounters: A Developmentally Deep Perceptual System for a Humanoid Robot. Ph.D. Thesis, Massachusetts Institute of Technology, Boston.

Fitzpatrick, P., Metta, G., Natale, L., Rao, S. and Sandini, G., 2003. Learning About Objects Through Action: Initial Steps Towards Artificial Cognition, IEEE International Conference on Robotics and Automation (ICRA 2003), Taipei, Taiwan.

Fu, K.S., Gonzalez, R.C. and Lee, C.S.G., 1987. Robotics : control, sensing, vision, and intelligence. CAD/CAM, robotics, and computer vision. McGraw-Hill, New York, 580 pp.

Gallese, V., Fadiga, L., Fogassi, L. and Rizzolatti, G., 1996. Action recognition in the premotor cortex. Brain, 119: 593-609.

Ghez, C., 1991. Muscles: Effectors of the Motor Systems. In: E.R. Kandel, J.H. Schwartz and T.M. Jessel (Editors), Principles of Neural Science. Appleton&Lange, Norwalk, CT, 548-563.

Gibson, J.J., 1977. The theory of affordances. In: R. Shaw and J. Bransford (Editors), Perceiving, acting and knowing: toward an ecological psychology. Lawrence Erlbaum, Hillsdale, 67-82.

Gilmore, R.O. and Johnson, M.H., 1998. Learning What is Where: Oculomotor Contributions to the Development of Spatial Cognition. In: F. Simion and G. Butterworth (Editors), The Development of Sensory. Motor and Cognitive Capacities in Early Infancy: from perception to cognition. Psychology Press, 25-47.

Gomi, H. and Kawato, M., 1997. Human arm stiffness and equilibrium-point trajectory during multi-joints movement. Biological Cybernetics, 76: 163-171.

Graziano, M.S.A., 1999. Where is my arm? The relative role of vision and proprioception in the neuronal representation of limb position. Proceedings of the National Academy of Science, 96: 10418-10421.

Graziano, M.S.A., Cooke, D.F. and Taylor, C.S.R., 2000. Coding the location of the arm by sight. Sience, 290: 1782-1786.

Grosso, E., Manzotti, R., Tiso, R. and Sandini, G., 1995. A Space-Variant Approach to Oculomotor Control, IEEE International Symposium on Computer Vision, Coral Gables, California.

Hirai, K., Hirose, M., Haikawa, Y. and Takenaka, T., 1998. The development of Honda humanoid robot, IEEE International Conference on Robotics and Automation, Leuven, Belgium, pp. 1321-1326.

Hogan, N., 1985. The Mechanics of Multi-Joint Posture and Movement Control. Biological Cybernetics, 52: 316-331.

Horn, B., 1986. Robot vision. The MIT electrical engineering and computer science series. MIT Press, Cambridge, Mass., 509 pp.

Humphreys, G., 2001. Objects, affordances... action! The Psycholigist, 14(8): 408-412.

Irie, R.E., 1995. Robust Sound Localization: An Application of an Auditory Perception System for a Humanoid Robot, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge.

Jeannerod, M., 1994. Object Oriented Action. In: K.M.B. Bennet and C. U. (Editors), Insights into the Reach to Grasp Movement. Elsevier Science, 3-15.

Jeannerod, M., 2002. The mechanism of self-recognition in humans. Behavioural Brain Research, 142: 1-15.

Jeannerod, M., Arbib, M.A., Rizzolatti, G. and Sakata, H., 1995. Grasping objects: the cortical mechanisms of visuomotor transformation. Trends in Neurosciences, 18(7): 314-320.

Johnson, C.A., Adams, J.A. and Kawamura, K., 2003. Evaluation of an Enhanced Human-Robot Interface, 2003 IEEE International Conference on Sysytema, Man, and Cybernetics, Washington, D.C., USA.

Johnson, M.H., 1997. Developmental Cognitive Neuroscience. Fundamentals of Cognitive Neuroscience, 1. Balckwell Publisher Inc., Malden, MA and Oxford UK, 234 pp.

Jordan, M., I., 1996. Computational Motor Control. In: M.S. Gazzanica (Editor), The Cognitive Neuroscience. MIT Press, Cambridge, MA, 597-609.

Kass, M., Witkin, A. and Terzopoulos, D., 1988. Shakes: Active Contour Models. International Journal of Compuer Vision: 321-331.

Katayama, M. and Kawato, M., 1993. Virtual trajectory and stiffness ellipse during multijoint arm movement predicted by neural inverse models. Biological Cybernetics, 69: 353-362.

Knudsen, E.I., 1981. The Hearing of the Barn Owl. Scientific American, 245: 82-91.

Knudsen, E.I. and Knudsen, P.K., 1985. Vision Guides the Adjustment of Auditory Localization in Young Barn Owls. Science, 230: 545-548.

Kolacinski, R., M. and Quinn, R., D., 1998. A novel biomimetic actuator system. Robotics and Autonomous Systems, 25: 1-18.

Krauzlis, R.J. and Stone, L.S., 1999. Tracking with the mind's eye. Trends in Neuroscience, 22(12): 544-550.

Leymarie, F.F., 1990. Tracking and Describing Deformable Objects using Active Contour Models. Master Thesis, McGill University, Montreal, Quebec, Canada.

Lungarella, M., Metta, G., Pfeifer, R. and Sandini, G., 2003. Developmental Robotics: A Survey. Connection Science, Forthcoming.

Manzotti, R., Gasteratos, A., Metta, G. and Sandini, G., 2001. Disparity estimation in log polar images and vergence control. Computer Vision and Image Understanding, 83(2): 97-117.

Marjanovic, M., Scassellati, B. and Williamson, M., 1996. Self-Taught Visually-Guided Pointing for a Humanoid Robot. From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB-96): 35-44.

Maturana, H., R. and Varela, F., .J., 1998. The tree of knowledge, the biological roots of human understanding. Shambhala Publications, Inc., Boston & London, 269 pp.

McCarty, M.E., Clifton, R.K., Ashmead, D.H., Lee, P. and Goubet, N., 2001. How Infants Use Vision for Grasping Objects. Child Development, 72(4): 973-987.

Meredith, M.A. and Stein, B.E., 1986. Visual, auditory and somatosensory convergence on cells in superior colliculus results in multisensory integration. Journal of Neurophysiology, 56(3): 640-662.

Metta, G., 2000. Babybot: a Study on Sensori-motor Development. Ph.D. Thesis Thesis, University of Genova, Genova, 176 pp.

Metta, G. and Fitzpatrick, P., 2003. Early Integration of Vision and Manipulation. Adaptive Behavior, 11(2): 109-128.

Metta, G., Sandini, G. and Konczak, J., 1999. A Developmental Approach to Visually-Guided Reaching in Artificial Systems. Neural Networks, 12(10): 1413-1427.

Miall, R.C. and Wolpert, D.M., 1995. Forward Models for Phsycological Motor Control. Neural Networks, 9(8): 1265-1279.

Middlebrooks, J.C., Makous, J.C. and Green, D.M., 1989. Directional sensitivity of sound-pressure levels in the human ear canal. The Journal of the Acoustical Society of America, 86: 89-108.

Milner, A.D. and Goodale, M.A., 1995. The Visual Brain in Action. Oxford Psychology, 27. Oxford University Press, Oxford.

Murata, A. et al., 1997. Object representation in the ventral premotor cortex (area F5) of the monkey. Journal of Neurophysiology(78): 2226-2230.

Mussa-Ivaldi, F., A. and Bizzi, E., 1993. Structural Constrains And Computationsl Problems in Motor Control. In: P. Dario, G. Sandini and P. Aebischer (Editors), Robots and Biological Systems: Towards a New Bionics ? Springer-Verlag, Berlin, 339-359.

Mussa-Ivaldi, F.A. and Giszter, S.F., 1992. Vector field approximation: a computational paradigm for motor control and learning. Biological Cybernetics, 67: 491-500.

Mussa-Ivaldi, F.A., Giszter, S.F. and Bizzi, E., 1993. Convergent Force Fields Organized in the Frog's Spinal Cord. The Journal of Neuroscience, 13(2): 467-491.

Mussa-Ivaldi, F.A. and Hogan, N., 1991. Integrable Solutions fo Kinematic Redundancy via Impedance Control. Internation Journal of Robotics Research, 10(5): 481-491.

Natale, L., Metta, G. and Sandini, G., 2002a. Development of Auditory-evoked Reflexes: Visuo-acoustic Cues Integration in a Binocular Head. Robotics and Autonomous Systems, 39(2): 87-106.

Natale, L., Rao S. and Sandini, G., 2002b. Learning to act on objects, Second International Workshop, BMCV 2002. Lecture Notes in Computer Science. Springer, Tubingen, Germany, pp. 567-575.

Packer, O. and Williams, D.R., 2003. Light, the Retinal Image, and Photoreceptors. In: S.K. Shevell (Editor), The Science of Color. Elsevier, 41-102.

Panerai, F., Metta, G. and Sandini, G., 2000. Visuo-inertial Stabilization in Space-variant Binocular Systems. Robotics and Autonomous Systems, 30(1-2): 195-214.

Panerai, F., Metta, G. and Sandini, G., 2002. Learning Stabilization Reflexes in Robots with Moving Eyes. Neurocomputing, 48(1-4): 323-337.

Pfeifer, R., 1996. Building "Fungus Eaters": Design Principles of Autonomous Agents, Fourth International Conference on Simulation of Adaptive Behavior. From Animals To Animats. MIT Press/Bradford Books, pp. 3-12.

Pfeifer, R., 2000. On the role of morphology and materials in adaptive behavior, Sixth International Conference on the Simulation of Adaptive Behavior. From Animals To Animats. MIT Press, Paris, pp. 23-32.

Pratt, G.A. and Williamson, M.M., 1995. Series Elastic Actuators, IROS, Pittsburgh.

Robinson, D.A., Pratt, J.E., Paluska, D.J. and Pratt, G.A., 1999. Series Elastic Actuator Development for a Biomimetic Robot, IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, Atlanta, Georgia, USA, pp. 561-568.

Robinson, D.W., 2000. Design and Analysis of Series Elastic Actuators in Closed-loop Actuator Force Control. Ph.D Thesis, MIT, Boston.

Rochat, P. and Striano, T., 2000. Perceived self in infancy. Infant Behavior & Development, 23: 513-530.

Ronnqvist, L. and von Hofsten, C., 1994. Neonatal finger and arm movements as determined by a social and an object context. Early Development and Parenting, 3(2): 81-94.

Rosander, K. and von Hofsten, C., 2003. Infants' emerging ability to represent object motion. Cognition, In press.

Rucci, M., Edelman, G.M. and Wray, J., 1999. Adaption of Orienting Behavior: From the Barn Owl to a Robotic System. IEEE Transactions on Robotics and Automation, 15(1): 96-110.

Russell, S. and Norvig, P., 1995. Artificial Intelligence: A Modern Approach. Artificial Intelligence. Prentice Hall.

Sandini, G. and Metta, G., 2003. Retina-Like Sensors: Motivations, Technology and Applications. In: G.B. Friedrich, J.A.C. Humphrey and T.W. Secomb (Editors), Sensors and sensing in biology and engineering. Springer-Verlag Wien New York, Wien, 251-262.

Sandini, G., Metta, G. and Konczak, J., 1997. Human Sensori-motor Development and Artificial Systems, AIR&IHAS '97, Japan.

Sandini, G., Questa, P., Scheffer, D., Dierickx, B. and Mannucci, A., 2000. A Retina-Like CMOS Sensor and Its Applications, 1st IEEE SAM Workshop, Cambridge, USA.

Sandini, G. and Tagliasco, V., 1980. An Anthropomorphic Retina-like Structure for Scene Analysis. Computer Vision, Graphics and Image Processing, 14(3): 365-372.

Schmidt, D.C., 2003. The ADAPTIVE Communication Environment. http://www.cs.wustl.edu/~schmidt/ACE.html.

Schmidt, D.C. and Huston, D.H., 2002. C++ Network Programming: Mastering Complexity Using ACE and Patterns. Addison-Wesley Longman.

Schwartz, E.L., 1980. A Quantitative Model of the Functional Architecture of Human Striate Cortex with Application to Visual Illusion and Cortical Texture Analysis. Biological Cybernetics, 37: 63-76.

Stein, B.E. and Meredith, M.A., 1993. The merging of the senses. Bradford Book. MIT Press, Cambridge.

Streri, A., 1993. Seeing, reaching, touching : the relations between vision and touch in infancy. Developing body and mind. MIT Press, Cambridge, Mass., xvi, 224 pp.

Van der Meer, A.L.H., Van der Weel, F.R. and Lee, D.N., 1995. The Functional Significance of Arm Movements in Neonates. Science, 267: 693-695.

Vernon, D., 2003. Personal Communication.

Viola, P.A. and Jones, M.J., 2001. Robust Real-Time Object Detection. CRL 2001/01, Compaq Cambridge Research Laboratory, Cambridge.

Viviani, P. and Stucchi, N., 1992. Motor-perceptual interactions. In: G. Stelmach, E. and J. Requin (Editors), Tutorials in Motor Behavior. Elsevier Science, 229-248.

von Hofsten, C., 1982. Eye-hand coordination in newborns. Developmental Psychology, 18: 450-461.

von Hofsten, C., 1983. Catching skills in infancy. Experimental Psychology: Human Perception and Performance, 9: 75-85.

von Hofsten, C., 2003. Personal Communication.

Williamson, M., 1996. Postural primitives: interactive behavior for a humanoid robot arm, From Animals to Animats: 4th International Conference on Simulation of Adaptive Behavior. Complex Adaptive Systems. MIT Press, Cape Cod, MA.

Wolpert, D.M. and Flanagan, R.J., 2001. Motor Prediction. Current Biology, 11(18): R729-R732.

Yoshikawa, Y., Hosoda, K. and Asada, M., 2003. Does the invariance in multi-modalities represent the body scheme ? - a case study with vision and proprioception -, 2nd Intelligent Symposium on Adaptive Motion of Animals and Machines, Kyoto, Japan.

Index

ACE, 39, 41
action, 24
  and perception, 25, 28, 115
  interpretation, 119
active agent, 23
active control, 51
active exploration, 116
active object, 40
active system, 68, 112
active touch, 24
active vision, 116
actor, 44, 45
actuator
  hydraulic, 51
  pneumatic, 51
  series elastic, 51
adaptation
  physical, 47
  shape, 107
adaptive behavior, 21
Adaptive Communication Environment. *See* ACE
adaptive control, 50
affordances, 25, 117, 118, 120
agency, 80
Anguita, D., 45
area 5, 80
arm control schema, 98
Arsenio, A., 88
Artificial Intelligence, 15, 16
  Behavioral Based, 16, 18
  Classical, 16
  Embodied, 17
  goals of, 15
  Knowledge-Based, 16
  Symbolic, 16, 17
Asada, M., 21
attentional system, 72
audition, 33
Babybot, 20, 31, 32, 34, 35, 69, 70, 71, 72, 115
backdrivability, 51
background modeling, 121
backpropagation, 45
Ballard, D.H., 116
Barn Owl, 19
Barth, F., 19
batch learning, 45
Beer, R., 17, 18, 19
behavior
  adaptive, 21
  bootstrapping, 43
  compliant, 51
  exploratory, 43
  hand tracking, 88
  imitative, 22
  innate, 22
  motor, 120
  orienting, 70
  random, 44
  reflexive, 22, 26
  sensorimotor, 43
  tracking, 87, 96
Behavioral Based AI, 16, 18

Bernardino, A., 90
Berthouze, L., 19
Berthoz, A., 69, 115, 117
Berton, F., 67
biological motion, 119
biological systems, 16, 19, 21, 47, 63
biomorphic actuator, 20, 28
Bizzi, E., 19, 20, 50
Blauert, J., 19
body schema, 75, 79, 80, 88
body-map, 27, 115
Brooks, R., 17, 19, 20, 21, 51
Brown, C.M., 116
BT848, 35
C++, 39, 40, 42
canonical neurons, 119
Capurro, C., 19
Carpenter, R., 69, 71
Cartesian moments, 122
Cartesian space, 98
categorization, 24
CCD, 35
central moments, 124
central nervous system, 48, 49, 50, 63
Cheng, G., 20
chess play, 16
Chiel, H.J., 18
Chinese room argument, 16
Classical AI, 16
closed-loop, 70, 80
CNS. *See* central nervous system
COG, 21, 39, 72
cognition, 16, 17, 18, 20
  theories of, 17, 18
cognitive abilities, 28
cognitive system, 18, 115
Cognitivism, 17, 18
cognitivistic approach, 17
color histogram, 83
color segmentation, 72, 100, 101
communication, 37, 38, 39, 40, 42
compliance, 33, 47, 51, 52, 53, 63, 64, 106, 107
  object's, 112
  passive, 51
computer vision, 16, 24, 120
cones, 65, 66
Connectionism, 17
connectionist system, 17
control
  closed-loop, 70
  computed torque, 50
  feedforward, 77
  force, 33, 51
  impedance, 51, 76
  low-impedance, 20
  position, 50
  saccadic, 69
  stiffness, 51
  vergence, 70
coordinate system, 48, 67, 76, 124
coordination
  eye-hand, 26
  eye-head, 26, 27, 71
  head-arm, 97
  sensorimotor, 17, 20, 26, 43, 44
Coriolis, 76
correspondence problem, 96
critic, 44, 45
cross-correlation, 70, 89
curse of dimensionality, 23
daemon, 42
Dario, P., 19
DC motors, 35
Desmurget, M., 20
development, 17, 21, 26, 27, 68, 80, 81, 96, 120
  cognitive, 21
  sensorimotor, 26
Dick, P.K., 13

direct kinematics, 107
disparity, 70, 71, 90, 96, 106
distal learning, 44
dorsal pathway, 118
dorsal stream, 24
double saccade, 116
DSP, 35, 66
dynamical approach, 17
dynamical approaches, 17
dynamical interaction, 22
dynamical systems, 18
dynamics
  body, 22
  inverse, 20, 48, 50
ecological level, 116
egocentric reference-frame, 84
egomotion, 89, 105, 117
elastic actuators, 20
elasticity, 19, 33, 50, 63, 110, 112
ellipse, 84, 87, 91, 93, 123
  fitting, 122
  representation, 84
Embodied AI, 17
embodiment, 17, 18, 20
enaction, 18
encoders, 33, 52, 53, 60, 61, 62, 88, 107
  Hall-effect, 33
  magnetic, 36, 107
end-effector, 50, 51, 96
epigenetic robotics, 21
equilibrium point, 49
equilibrium point hypothesis, 20, 50
Ethernet, 35
event-interpretation, 120
evolution, 17
exploitation, 43
eye-hand coordination, 26
eye-movements, 26, 68
F5, area, 25, 118, 119
Fadiga, L., 25
feedback
  encoder, 108
  error learning, 43, 44
  force, 51, 52, 53, 77
  kinesthetic, 80
  position, 52
  proprioceptive, 80, 82, 106, 109, 110, 115, *See* proprioception
  tactile, 106, 109, 110, *See* touch
  visual, 80
feedforward, 77
Fitzpatrick, P., 39, 72, 81, 89, 95, 106
Flanagan, R.J., 116
force feedback, 77
force sensing resistor, 33, 109
force sensor, 34
force sensors, 79
formal logic, 16
forward model, 88, 116, 117
fovea, 19, 33, 65, 67, 68, 70, 96, 100
frame difference, 121
frame grabbers, 35
FSR. *See* force sensing resistor
Fu, K.S., 50
function approximator, 43
Gallese, V., 25, 118, 119
generation of expectation, 116
Ghez, C., 48, 49
Gibson, J.J., 25, 117
Gilmore, R.O., 116
Giotto sensor, 66, 67
Giszter, S.F., 20
Gomi, H., 20
Goodale, M.A., 24, 118
grasping action, 83, 109, 112, 119
gravity compensation, 76, 77, 79
Graziano, M.S.A, 80
Grosso, E., 19
haed kinematics, 98
Hall-effect encoders, 33, 107

hand calibration, 107
hand localization, 86, 89, 100
hand prediction, 87
head posture, 98
Hirai, K., 20
histogram
  backprojection, 83, 87
  color, 83
  intersection, 83
  polar, 101
Hogan, N., 20, 50
homunculus, 20
Hooke’s law, 49, 51
Hooke's law, 52, 53, 54
Horn, B., 123
how representation, 25
HSV. *See* Hue-Saturation space
Hue-Saturation space, 83, 100
humanoid robot, 21, 41, 115
humanoid robots, 20, 21, 31
Humphreys, G., 118
Huston, D.H., 39
imitation, 28
impedance control, 51, 76
inertial sensors, 19
information
  3D, 106
  contextual, 117
  disparity, 70, 90, 106
  kinesthetic, 23
  motor, 119
  multimodal, 24
  periodic, 82
  positional, 70
  postural, 112
  posture, 80
  proprioceptive, 81, 88, 109
  retinocentric, 84
  sensory, 70, 115
  shape, 84
  stereoscopic, 96
  structural, 119
  vestibular, 26, 69
  visual, 69, 71, 80, 87, 88, 96, 106, 113, 116
inhibition, 43, 45
integral image, 82
Intel, 40
Intel Processing Library (IPL), 40
intelligence, 15, 16, 17, 18, 20, 47
intermodal form, 80
internal model, 22, 26, 88, 116
interpretation, 24
inverse kinematics, 103
inverse model, 70
IOCtl, 42
Irie, R., 19
ISA, 35
Jeannerod, M., 24, 25, 80, 118
Johnson, M.H., 22, 117
Jones, M.J., 82
Jordan, M.I., 20
JR3, 34
Katayama, M., 50
Kawato, M., 20, 50
kinematic transformation, 95
kinematic visual illusions, 119
kinematics
  body, 22
  direct, 85, 107
  head, 90, 98
  inverse, 20, 48, 50, 76, 79, 85, 103
kinesthetic experience, 24
kinesthetic feedback, 80
Knowledge-based AI, 16
Knudsen, E.I., 19
Kolacinski, R., 63
Krauzlis, R.J., 117
Kuniyoshi, Y., 19, 20
Lagrange equation, 76

learing to interact, 115
learning
  about objets' shapes, 106
  architecture, 43
  batch, 45
  body-map, 115
  distal, 44
  feedback error, 43, 44
  gravity compensation, 76
  motor, 119
  reinforcement, 43
  self-supervised, 23, 85
learning to act, 100, 105
learning to interact, 27
learning to interpret events, 27
learning to understand, 27, 115
Linux, 37, 39
Linux RT, 37
location transparency, 38
log-polar, 33, 35, 66, 67, 68, 82, 100, 121, 122, 124
look-up table, 102, 106
low-impedance control, 20
low-stiffness, 35
Lungarella, M., 21
manipulation, 23, 24, 29, 48
map
  body-map, 27
  cross modal, 88
  direction, 101, 104
  force, 103
  handlocalization, 44
  motor-motor, 26, 96, 98
  ocular, 26
  reaching, 98
Marjanovic, M., 88
Matlab, 41, 108, 134
Maturana, H.H., 18
Meredith, M.A., 20, 70
Metta, G., 20, 26, 27, 66, 69, 70, 81, 89, 96, 98, 106, 115
Miall, R.C., 116
microphones, 33, 35
Microsoft, 37, 40
Middlebrooks, J., 19
Milner, A.D., 24, 118
minimum jerk, 20
minimum torque, 20
mirror neurons, 25, 119
model
  feedforward, 77
  forward, 88, 116, 117
  internal, 116
morphology, 19
motion, 121
  self-generated, 81
motor behavior, 120
motor behaviors, 28
motor control, 19, 20, 37, 43, 44, 47, 48, 116
motor learning, 119
motor neurons, 19, 48
motor representation, 118, 119
motor schema, 117
motor schemas, 22
motor synergies, 21, 44
motor synergy, 95, 110, 112
motor system, 72, 119
motoric representation, 119
MST, 117
Multicast, 38, 39, 40, 42
multimodal representation, 24
Murata, A., 118
muscle
  equilibrium point, 49
  force-length characteristic, 49
  spring-like properties, 50
  stiffness, 49
  tension, 49

  tension-length characteristic, 49
muscles, 47, 48, 51, 52, 63
Mussa-Ivaldi, F.A., 19, 20, 50
Natale, L., 19, 20
nervous system, 18
network protocol
  Multicast, 38
  QNET, 38, 40
  TCP/IP, 38
  UDP, 38
neural network, 17, 45, 79, 81, 84, 85, 96, 106
neural networks, 84
  recurrent, 18
neuroscience, 20
Newton's third law, 103
Norvig, P., 15, 16
ocular system, 26
OKR. *See* opto-kinetic reflex
ontogenesis, 21
ontogeny, 21
open-source, 39
operating system, 37, 39
  Linux, 37
  Linux RT, 37
  QNX, 37
  UNIX, 42
  VxWorks, 37
  Windows, 37
  wrapper, 39
optical encoders. *See* encoders
optic-flow, 69, 82, 88, 89, 121
opto-kinetic reflex, 26, 69
opto-kinetic response, 26
orienting behavior, 19, 70
PAL, standard, 37
Panerai, F., 19, 20, 69
PCI, 35
Pentium, 35
  PIV, 35, 36
perception, 21, 24, 116
  motor theories of, 115
  speech, 119
periphery, 19, 33, 65, 66, 85
Pfeifer, R., 17, 19
phylogenesis, 21
phylogeny, 21
physiology, 17
PID, 50, 56, 57, 76, 77
planning, 20, 69, 100, 104, 105, 119
polar histogram, 101
position control, 50
posterior parietal cortex, 25
power grasp, 27
pragmatic representation, 24, 118
Pratt, G.A., 20
precision grip, 27
prediction
  hand, 87
  hand position predictor, 85
  hand shape predictor, 85
  predictive abilities, 23
premotor cortex, 25, 80
proprioception, 33, 80, 88, 101, 106, 115
PUMA, 31, 34, 35, 51, 77
pushing action, 100
PWM, 35
QNX, 37, 38, 39, 40
Quinn, R.D., 63
rationality, 15
reaching, 27, 95, 97, 100
real-time, 37, 38, 66, 96
reasoning, 15, 16, 17
recognition, 24
recurrent neural networks, 18
reference frame, 76
  arm-centric, 80
  body centered, 90
  bodycentric, 84

  egocentric, 84
reflexes, 21, 26, 69, 97
representation
  affordance, 118
  how, 25
  motor, 118
  motoric, 119
  pragmatic, 24, 118
  semantic, 25, 118
  what, 25
  where, 25
resting length, 48, 49, 52, 53, 54
retina, 19, 33, 65, 66, 67, 69, 116
retinal coordinate system, 76
retinal coordinates, 100, 101
retinal displacement, 104, 105
retinal error, 100
retinal plane, 84
retinal slip, 116
retinal slips, 90
Rizzolati, G., 119
Robinson, D.W., 20
robotics, 16, 17, 19, 20, 47, 48, 50, 63, 66, 76, 79, 81, 88
  cognitive developmental, 20
  developmental, 21
  epigenetic, 21
Rochat, P., 81
Ronnqvist, L., 23
Rosander, K., 23
Rucci, M., 19
run-time, 38, 40
Russell, S., 15, 16
saccade, 116
saccades, 26, 69, 70, 71
saccadic control, 69
Sandini, G., 44, 66, 139
Santos-Victor, J., 90
scheduling, 37
Schmidt, D.C., 39
Schwartz, E., 66
Searle, J., 16
segmentation, 82, 84, 85, 88, 90, 122
  color, 72, 101
  hand, 82
  motion-based, 83
  object, 24
self-knowledge, 80
self-organizing map, 110, 111, 113, 125
self-perception, 81
self-supervised learning, 85
semantic representation, 25, 118
sense of movement, 115
sensorimotor coordination, 17, 20, 26, 43, 44
series elastic actuator, 51
shared memory, 38, 40
situatedness, 17, 20
smooth pursuit, 69, 117
software architecture, 28, 31, 37, 38
sound card, 35
SourceForge, 39
spatial correlation, 81
speech perception, 119
spring, 28, 33, 48, 49, 50, 51, 52, 53, 54, 55, 57, 63, 64, 76
Stein, B.E., 20, 70
stereo fusion, 70
stiffness, 19, 22, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 76
  control, 51
Stone, L.S., 117
strain gauges, 52, 79
Streri, A., 22
Striano, T., 81
Stucchi, N., 116, 118, 119
subsumption architecture, 20, 45
sucking reflex, 22

superior colliculus, 70
Symbolic AI, 17
Symbolic systems, 16
synchronization, 37
tactile sensors, 79
Tagliasco, V., 66
task-coordinate frame, 48
TCP/IP, 38, 39, 40
template, 39, 41
thread, 40
tilt, 98
timing, 81
torque control, 50
touch, 33, 109, 116
  active, 24
  double, 80
touch-elicited grasp, 108
tracking, 26, 100
  behavior, 87, 96
  hand, 88
  smooth, 22
training set, 23, 43
trajectory generator, 98
Turing test, 16
Turing, A., 16
UDP, 38, 40
Unimation, PUMA 260, 31
UNIX, 42
Van der Meer, A.L.H., 22
Varela, F.J., 18
ventral pathway, 118
ventral stream, 24
vergence, 70, 71, 72, 74, 96, 98
Vernon, D., 17
version, 39, 68, 70, 71, 74, 96, 98
vestibular organ, 19
vestibular system, 33, 69, 115
vestibulo-ocular reflex, 26, 69, 71
Viola, P.A., 82
virtual device driver, 41, 42
virtual finger, 32, 107
virtual trajectory, 50
vision, 33, 116
  panoramic, 68
  space-variant, 68
visual pathways, 118
visual stabilization, 19, 26, 27, 69, 89
Viviani, P., 116, 118, 119
von Hofsten, C., 22, 23, 96, 108, 109
VOR. *See* vestibulo-ocular reflex
VxWorks, 37
walking, 119
what representation, 25
where representation, 25
Williamson, M.M., 20
Windows, 37, 39, 40
Wolpert, D.M., 116
wrapper, 39
YARP, 39, 40, 41, 42, 43
Yoshikawa, Y., 81, 88