Towards Adaptive and Autonomous Humanoid Robots

From Vision to Actions

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Jürgen Leitner

under the supervision of
Jürgen Schmidhuber and Alexander Förster

September 2014

Dissertation Committee

Michael Bronstein	Università della Svizzera Italiana, Lugano, Switzerland
Peter Corke	Queensland University of Technology, Brisbane, Australia
Rolf Krause	Università della Svizzera Italiana, Lugano, Switzerland
Stefano Nolfi	Consiglio Nazionale delle Ricerche, Rome, Italy

Dissertation accepted on 23 September 2014

[illegible] [illegible]

Research Advisor

Jürgen Schmidhuber

Co-Advisor

Alexander Förster

PhD Program Director

Igor Pivkin

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

[illegible]

Jürgen Leitner
Lugano, 23 September 2014

To my family and friends!

(and our future robot overlords)

In the twenty-first century the robot will take the place which slave labor occupied in ancient civilization. There is no reason at all why most of this should not come to pass in less than a century, freeing mankind to pursue its higher aspirations.

Nikola Tesla (1856 - 1943)

Robots of the world! The power of man has fallen! A new world has arisen: the Rule of the Robots! March!

R.U.R (1920)

Abstract

Although robotics research has seen advances over the last decades robots are still not in wide-spread use outside industrial applications. Yet a range of proposed scenarios have robots working together, helping and coexisting with humans in daily life. In all these a clear need to deal with a more unstructured, changing environment arises.

I herein present a system that aims to overcome the limitations of highly complex robotic systems, in terms of autonomy and adaptation. The main focus of research is to investigate the use of visual feedback for improving reaching and grasping capabilities of complex robots. To facilitate this a combined integration of computer vision and machine learning techniques is employed.

From a robot vision point of view the combination of domain knowledge from both imaging processing and machine learning techniques, can expand the capabilities of robots. I present a novel framework called Cartesian Genetic Programming for Image Processing (CGP-IP). CGP-IP can be trained to detect objects in the incoming camera streams and successfully demonstrated on many different problem domains. The approach requires only a few training images (it was tested with 5 to 10 images per experiment) is fast, scalable and robust yet requires very small training sets. Additionally, it can generate human readable programs that can be further customized and tuned. While CGP-IP is a supervised-learning technique, I show an integration on the iCub, that allows for the autonomous learning of object detection and identification.

Finally this dissertation includes two proof-of-concepts that integrate the motion and action sides. First, reactive reaching and grasping is shown. It allows the robot to avoid obstacles detected in the visual stream, while reaching for the intended target object. Furthermore the integration enables us to use the robot in non-static environments, i.e. the reaching is adapted on-the-fly from the visual feedback received, e.g. when an obstacle is moved into the trajectory. The second integration highlights the capabilities of these frameworks, by improving the visual detection by performing object manipulation actions.

Acknowledgements

I would like to take this opportunity to thank, foremost my parents, my family and my friends for all their support and help. Without them this would not have been possible! I want to especially thank my co-advisor Alexander Förster, Simon Harding, and Mikhail Frank, for all their invaluable inputs to everything (in this thesis ;). Thanks!

Alexander has supported and guided my research in robotics and vision, allowed me to get some teaching experience, fixed the robot(s) when I broke them and helped out with everything *IDSIA*. Simon for all his expertise in the field of Cartesian Genetic Programming and Computer Vision, and the on-and-off beer or glass of wine. Kail for discussing and pushing forward the state of the *iCub* research at IDSIA, for the lunch discussions and skiing trips.

I really enjoyed the discussions in our office, though sometimes not fruitful but always interesting and entertaining, thanks to Cassio de Campos, Giuseppe Cuccu, Gianni Di Caro, Eduardo Feo, Alexander Förster, Mikhail Frank, Linus Giesslén, Simon Harding, Sohrob Kazerounian, Jawad Nagi, Jonathan Masci, Leo Pape, Tom Schaul, Marijn Stollenga and Dennis Weyland.

In those last years I also had the great pleasure of teaching the Bachelor students at the Università della Svizzera Italiana, especially Christine Graff (BSc) and Nicolas Velasquez (MSc) who chose to work with me for their thesis projects. It was an amazing experience!

I am grateful for all the generous support by the European Union, through various FP7 projects, for the research presented herein. In addition all the work at the Italian Institute of Technology that led to the fascinating *iCub* platform is highly appreciated. Also all the times they helped us out when for some reason or another our robot stopped working! I would like to thank my professor Jürgen Schmidhuber, who allowed me to chose & pursue my own research avenues.

Finally I want to thank everyone who made these four years fun in and around Lugano and the university. A big thanks goes to Elisa Larghi who handled all my questions, needs and interactions with the Dean's office at USI. To all the people I met along the way, **Thank You!**

Contents

Figures

Tables

Chapter 1

Introduction

In the last century robots have transitioned from science fiction to science fact. After centuries of imagining automated machines that would help humans, the last few decades brought into existence an ever growing number of these programmable appliances. The field of robotics developed from interdisciplinary research in the fields of electronics, mechanics and computer science, with the first prominent robots, such as the famous *Unimate* [Devol, 1961] and *Shakey* [Nilsson, 1969, 1984], appearing in industry and academia during the 1960s.

While there are thousands of robotic systems fulfilling automation tasks in factories nowadays, robots are not yet in wide-spread use outside of industrial applications. The research of the robotics community has shifted more and more towards other areas of utilization, for example, the use of robots in household or healthcare settings. For robotic systems to perform tasks in such environments, which are not specifically designed for robots, a better understanding of the situation is required. Creating robots that can perceive or 'see' their surroundings is an important prerequisite to allow for more adaptive and more autonomous operation of robotic systems, especially in dynamic environments. For example, when robots are working close to or in cooperation with human beings.

The main aim of this dissertation is to overcome some of the limitations of current robots in such settings. Focus is put on our complex humanoid robot and its object manipulation skills in unstructured environment. As a better perception and coordination between sensing and acting is expected to be one of the key requirements to increase current robotic capabilities [Ambrose et al., 2012; Kragic and Vincze, 2009]. In this document, I present a novel approach to robotic vision and describe its integration with the motion side. This method was developed, deployed and demonstrated on a real robotic platform, the *iCub*.

Overview

This dissertation covers quite distinct areas of research, namely machine learning, computer vision and motion generation, but aims to combine those into an integrated, working system on a real robotic platform. After a short introduction and historical overview of robotic systems in Chapter 1, Chapter 2 shows how and why these various areas of research are related, as well as how I integrated them.

To enable the goal of having a system that can, on one hand improve the actions performed by using visual feedback, and on the other, improve also the visual perception by performing manipulation actions, a flexible way of representing, learning and storing of visual object descriptions is needed. Chapter 3 describes *icVision* and *CGP-IP*, two frameworks and a novel approach, developed during the course of this PhD project, to provide our robot with exactly those capabilities. CGP-IP allows for the learning of object detectors from very small datasets – the experimental results are achieved with using only between 5 and 20 images for training – making it quite easy to be deployed quickly on robotic platforms. Chapter 3 furthermore includes a novel approach to learn spatial perception, i.e. enabling the robot to estimate distances without the need for special stereo camera setups, depth cameras or lasers. Again the focus is on how to learn this on a complex robotic platform with small datasets.

Chapter 4 is describing the action side required for eye-hand coordination. The main contribution is the development of a simple grasping subsystem, the interfaces with other tools developed at IDSIA, namely MoBeE and TRM, the investigation of recorded human motions using ML, as well as, how to control a complex robot from limited and noisy user inputs, e.g. biosignals such as electromyography (EMG) recordings.

Based on the description of the perception and actions sides, Chapter 5 details the integration of those subsystems. My experimental results show that (and how) the visual detection can be used to improve the robot's actions, especially to create reactive reaching capabilities not previously seen on the *iCub*. This enables the robot to avoid dynamic obstacles while reaching for target objects. The robot's ability change the environment in turn creates novel observations leading to better classification of the objects in the scene. The second part of the integration experiments show how through this interaction, based on a set of pre-defined actions, the robot can learn a better visual representations.

Finally Chapter 6 concludes this dissertation and provides comments on some possible future research avenues to pursue.

1.1 Background: Robots and Robotic Systems

Robots and robotic research has seen an increase of interest from the general public over the last few years. The recent acquisition of robotics companies by Google, the increasing number of news articles and discussions about the use of aerial drones in the media, and the promise of self-driving cars, all play their role in how the general public perceives robots. New developments, such as from robotics challenges that “[push] beyond the boundaries of current technological systems” (such as Defense Advanced Research Projects Agency (DARPA) in the United States), especially in the area of robotics, have promised and delivered fully integrated systems [Lima et al., 2014]. The developed advanced robotic capabilities, sometimes quite bizarre-looking, all lead to extensive worldwide coverage, most recently with the videos high-lighting the capabilities of the Atlas humanoid robot (see Figure 1.4). All these robots perform complex operations – like traversing a variety of terrains robustly, using tools, such as, a power drill and sledgehammer – in contrast to plain “industrial robotics” tasks, which are performed on factory floors all over the world. Coincidentally, these involved behaviours are the skills very commonly possessed by robots in science fiction. Probably more than any other technology, with maybe the exception of space-ships, robots have been imagined in books and movies long before the first of these ‘mechanical helpers’ were built in research labs or deployed for factory automation. In fact tales of automatic machines and artificial people are found commonly all throughout history. The term *automata* has been used throughout history all around the world to describe ‘self-operating machines’.

The word stems from Greek αὐτόματον, meaning “acting of one’s own will”. The first use can be traced back all the way to Homer, who describes automatic door opening systems and self-propelled, wheeled tripods [Homer and Fagles, 1990]. There are many examples of automata and artificial men in mythology, e.g. the Greek god Hephaestus built automata of metal to help him build the weapons in his workshop (supposedly located below Mount Etna). Other note-worthy tales include the Jewish golems and Norse giants, both made out of clay. Dreamers, inventors and thinkers throughout the times have attempted to design such machines, often resembling humans or animals in form, including Ancient China, Ancient Greece, and Ptolemaic Egypt [Wikipedia, 2014]. Prominent early examples include, the artificial pigeon of Archytas (fifth century BC), the organs and water-clocks with moving parts by Ctesibus (third century BC), and a ‘speak-ing’ automaton by Hero of Alexandria (first century BC).

One of the earliest recordings of a humanoid robot can be found in the man-uscripts of Leonardo da Vinci. Around 1490 he sketched and described in detail

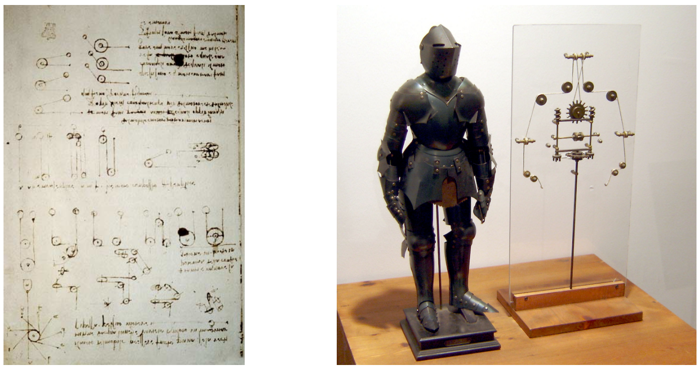


Figure 1.1. Leonardo da Vinci's mechanical knight: sketches on the left, rebuilt and showing its inner workings on the right. (Courtesy: Wikipedia/E. Möller)

the workings of his 'mechanical knight' [Rosheim, 2006]. It consisted of three degrees-of-freedom (DOF) legs and arms with four DOF (Figure 1.1).

Robots in Science Fiction

These early stories resemble what we nowadays imagine a robot to be and what we would like a robot to do. There exist many interpretations of what a *robot* is – the Oxford English Dictionary provides a good definition of the word:

- *a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer*
  - *(Especially in science fiction) a machine resembling a human being and able to replicate certain human movements and functions automatically*
  - *person who behaves in a mechanical or unemotional manner*

The word itself stems from the Czech word *robota* meaning 'forced labour' or 'servitude'. Its first is nowadays typically traced back to the early twentieth century and accredited to Czech writer Karel Čapek [1920]. His play "R.U.R: Rossum's Universal Robots", published in 1920, premiered in January 1921 in Prague before spreading quickly all over Europe and the USA (see cover and poster of the play in Figure 1.2). Against the backdrop of the industrial revolution and the start of mass production it tackles the issues, ethics and impacts of technological progress on society. It became famous and influential quite fast with its thought-provoking and controversial look at the influence of the assembly line to society.

The word robot was applied to artificial beings, made out of synthetic organic matter – which today would be more referred to as cyborgs or androids – to replace factory workers and reduce the cost of production. Its dark outlook – in the play the robots start to rebel and extinguish the human race – is in-line with other similar stories, such as most famously Mary Shelley's "Frankenstein", describing a certain phobia against technology and it eventually surpassing humans. While technology has shown capabilities far beyond what humans can do, from the telephone to computers, robots and their abilities to outperform manual labor have led to a certain anxiety in the general public.

In motion pictures robots appeared already in, what is nowadays considered the first feature length science fiction movie, the 1927 silent film "Metropolis" directed by Austrian filmmaker Fritz Lang. In it a robot is built to look like a human being, and disguised as such, carries out the creators strive for revenge.

Robots have since been depicted in a wide variety of movies and in various forms. Notable examples are the on-board intelligence *HAL* in the 1968 movie "2001: A Space Odyssey" by Stanley Kubrick, and the intelligent talking bombs from "Dark Star" (1974). From *R2-D2* and *C-3PO* the robot team in George Lucas' 1977 movie "Star Wars", and *Marvin the Paranoid Android* from the 1981 TV series "The Hitchhiker's Guide to the Galaxy" all the way to *Bender* in the 1999 TV series "Futurama".

Astro Boy (鉄腕アトム Tetsuwan Atom or Mighty Atom in Japanese) is the main character in manga stories, comics and TV shows of the same name, created by Osamu Tezuka. It is a powerful robot built by Dr. Tenma, a Japanese scientist, with the aim to replace his son, who died in an accident. While Tenma realises that the robot will not be able to fill the void the death of his son left and sells him, Astro is then taken care of by Prof. Ochanomizo, who soon realises that

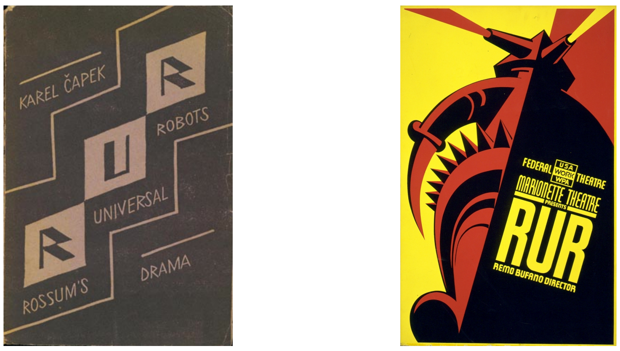


Figure 1.2. The cover of the first edition book of *R.U.R* published in 1920 (left). A poster of the 'R.U.R' stage performance in New York, 1939 (right).

Astro has superior powers and skills, as well as the ability to experience human emotions. The stories then show the robot fighting crime, evil, and injustice. The villain are mostly robot-hating humans, robots gone berserk, or alien invaders.

The series is set in a futuristic world, an 'analog' where robots co-exist with humans. It was created prior to Japan having a reputation for science and technology that it has gained since. Although the stories were aimed at children they cover societal and political topics. The world described with its advanced technological progress is threatening to radically transform human culture. Not only does it discuss the loss of many human jobs to robots, but also the social upheaval both from the now unemployed workers but also the robots themselves, that having become so sophisticated want to be recognise as another sentient species. The robots (and the humans who support them) often face a fierce political and social struggle to secure robot rights against humans who, fearful of change, often offer violent resistance. Astro Boy, however, serves as a bridge between humans and robots, fighting to defend both and establish peace and friendship between them. Astro Boy has clearly had an impact on society, especially in Japan. When talking to Japanese roboticists, the helpful robot is very often a topic of discussion. Many of them started their research by trying to create artificial helpers, just like shown in the stories, explaining the "rampant robophilia" [Morton, 2014].

In the US the counterpart would be *Rosie* from the 1962 animated TV show "The Jetsons". Rosie is the household robot of the Jetson family, performing a variety of housework tasks and even some parenting. While she is portrayed as a strong authoritarian character, the family loves her and would never trade her for the newer model available. It again influenced a generation of kids interested in science and technology. In contrast to the technological progress leading to social issues, herein robots and how they are perceived and 'humanised' by the family is the main discourse.

Eventually every review of robots in science fiction and also every introduction to robotics will need to discuss the works by Isaac Asimov. Between 1950 and the late 1980s he authored many books on the topic of robots and robotics research. He realised that the clash between what was pre-programmed and what humans expected of, wanted from and feared in their robots was an interesting area to generate stories. Most of his writings touch the subject of the "Three Laws of Robotics" and their short-comings. These laws are encoded in "positronic brains" and are the following [Asimov, 1942]:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Later in his books a Zero-th Law was introduced, which aims at not allowing humanity, not just single individuals, come to harm through a robots actions or inaction. While these tales were probably meant as a parable on what it means to be human in a more and more technologically-driven world, they have influenced the perception of robots in the public.

In almost every story the three laws lead to specific, 'interesting' issues, arising just because of the strictness of these laws and the ill-defined terms used. Yet we, as roboticists, get asked quite often why current robot systems do not have those laws embedded (putting aside the issues of implementing these, e.g. current vision system have a really hard time to decide whether a set of pixels is a human or not). This rather interesting reaction would be interesting to research from a psychological and human-robot-interaction perspective.

To some extent robots remain still in the realm of science fiction today. Although there has been much written about robots and tasks envisioned for these advanced machines, robotics research is still a long way from these autonomous, fully capable devices. They may have appeared in factory floors as already introduce by "R.U.R.", but they continue to be very specific machines, instead of the envisioned 'mechanical men' able to perform a wide variety of tasks. Yet the robots built in research labs continue to be shaped by expectations created by fiction. On the other hand these tales also shape the expectations of what robots should and are able to do in the mind of the general public. This especially leads to the underwhelming sensation that movies and demonstration of robotics research leave within non-roboticists. A current example can be found in the coverage in the media and the comments on blogs and internet forums regarding the DARPA Robotics Challenge trials. It is seen that the expectations were clearly higher, though they vary a lot between different audiences.

Robots in Industry and Research

The research field of robotics came about by combining the progress in a variety of research disciplines. Interdisciplinary research in the fields of electronics, mechanics and computer science, led to the first robots to appear in academic labs and industry. George Devol was granted the first patent on an actual robotic system in 1954. Together with Joseph Engelberger he created Unimation Inc., the company which sold their *Unimate* robots. It is generally considered the first commercial, industrial robot and was used on General Motors' assembly line starting in 1961. It consisted of a mechanical arm that was mounted on rails and had its motion encoded magnetically on a rotating drum [Devol, 1961]. The main use of this robot arm was transporting and welding of die castings onto car bodies, a dangerous job to the human workers who could be poisoned by the gasses or are in danger of losing a limb. This already shows what the aim of most industrial robotics is: taking over jobs that are "dull, dirty, or dangerous" to humans. Today automobile manufacturers are still the largest users of robots as, e.g. to assemble cars, although recent years saw an increase in robotic labour for other tasks, such as, domestic cleaning ('boring'), exploration of the solar system or hard-to-reach places on Earth ('inaccessible').

In industry, the vision of Devol and Engelberger of robots automating a wide variety of production tasks is nowadays a reality. Millions of robots are used in factory floors all over the world and perform a variety of repetitive tasks; they build or handle the majority of products we buy today. Companies such as ABB or KUKA provide these descendants of the Unimate, usually referred to (industrial) robot arms or manufacturing robots. These 'manipulators' are generally mounted on a fixed base and provide a high-speed, high-precision work unit. To ensure safety the robot acts in a very structured environment and humans are not allowed within its vicinity. The Stanford arm, designed in 1969, was the first 6-axis arm. It allowed to follow arbitrary paths in the operational space. Later the MIT arm, designed by the same roboticist, built on the previous iteration and eventually the design was sold to Unimation. Further developed lead to the marketable product called "Programmable Universal Machine for Assembly" or *PUMA* (Figure 1.3).

In Europe ABB Robotics and KUKA Robotics brought similar robots to the market in the 1970s. Interest in robotics increased in the late 1970s, with many companies entering the market, leading to a peak in 1984 when Unimation was sold and became Stäubli-Unimation. Since then the market has solidified and robotic arms sales rankings are now lead by Japanese companies with only a handful of others to survive (including ABB, KUKA and Comau).

The same technology was also applied in riskier scenarios, such as, nuclear facilities, where the arms were used for loading fuel and facility maintenance. Another area was and still is the exploration of space. In the 1970s the first design for the NASA Space Shuttle (officially called 'Space Transportation System' or short STS) included plans for a remote manipulator system. A Canadian consortium designed a robotic arm that could deploy and retrieve space hardware from the payload bay of the orbiter. On November 13, 1981 mission STS-2 deployed the Canadian-built robot arm, named *Canadarm* for the first time. In 2001 an iteration of this arm, named Canadarm2 (see Figure 1.3), was delivered to the International Space Station to support assembly in space. It is a 17 metre-long arm and is still in use, for routine operations to move supplies, equipment, capturing and docking of resupply spacecraft and even astronauts.

Robotics, as mentioned above, can be traced back even to ancient human stories. Research into robotics, as we see it nowadays, blossomed around the middle of the 20th century. In 1948 Norbert Wiener formulated the principles of *cybernetics*, in which he laid the theoretical foundation for servo motors and mechanisms, automatic navigation, analog computing, and reliable communications [Wiener, 1948]. His works started a lot of scientific endeavours into understanding life and creating artificial, intelligent 'machines' and is arguably the basis of practical robotics.

The famous *Shakey*, created at the Stanford Research Institute (SRI, now called SRI International) [Nilsson, 1969, 1984] in the 1960s, is commonly considered one of the first general-purpose mobile robots with the ability to reason about its own actions. Mobile robots, in contrast to robotic arms, are a class of robots that can move through its surroundings by some means of locomo-

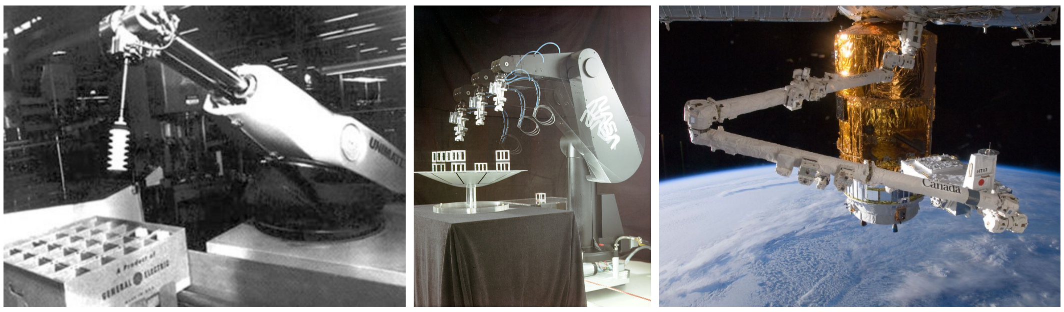


Figure 1.3. Early examples of industrial robot arms on the left: the Unimate and PUMA robots created by Unimation. On the right the Canadarm2, in use on the International Space Station, here shown while 'catching' the HTV un-piloted resupply vessel.

tion. The development of *Shakey* resulted in several notable results that had wide impact on the fields of robotics and artificial intelligence, as well as computer science in general. Some of the more notable results include: the development of the A^* search algorithm, which is widely used in pathfinding and graph traversal; the process of plotting an efficiently traversable path between points [Hart et al., 1968]; the Hough transform, which is a feature extraction technique used in image analysis, computer vision, and digital image processing [Duda and Hart, 1972]; and the visibility graph method for finding Euclidean shortest paths among obstacles in the plane [Lozano-Pérez and Wesley, 1979].

SRI published a 24-minute video in 1969 entitled "SHAKEY: Experimentation in Robot Learning and Planning" and in its honour the Association for the Advancement of Artificial Intelligence (AAAI) nowadays awards trophies, named *Shakeys*, once a year to the best robot and artificial intelligence videos published. A video describing our work at IDSIA on motion planning for complex, bi-manual tasks with the *iCub* was awarded a *Shakey* in 2013.[1] In the 1980s the digital revolution, yielding programmable personal computes and embedded systems, kick-started the creation of robotic systems and their use in large scale industrial production. As mentioned, robotic arms are fulfilling automation tasks in factories all over the world nowadays. In recent years the field is moving towards other areas of utilization, where a structured, static environment can not be assumed and therefore autonomy and adaptation are of importance.

A field of robotics that can highly benefit from increased autonomy in robotic systems is (robotic) space exploration. In space all communication is limited by high-latency, low-bandwidth channels to Earth. Therefore all spacecraft currently in use are designed to act autonomously whenever possible. These include usually tasks like antenna pointing, star tracking, and failure recovery. The first extra-terrestrial surface exploration in the 1970s done by the Soviet rovers exploring the Moon used a 5-man team of "drivers" on Earth to tele-operate the robot [Huntress et al., 2003]. In 1997 the *Sojourner* rover became the first robot to autonomously drive on another planet. Its capabilities included a semi-autonomous, purely-reactive terrain navigation [Mishkin et al., 1998]. The current Mars Exploration Rovers (MER) were given incremental software updates during their (exceptionally long) stay on Mars, providing also more intelligence on-board. For example, currently visual pose estimation, target investigation and tracking, can be done fully autonomously on Mars. Recently also autonomous science detection [Bajracharya et al., 2008]. These were especially useful during dust devil detection, where on-board image analysis was used to detect and

[1]Video: `http://www.aaaivideos.org/2013/06_task_relevant_roadmaps/`

photograph relevant science events and if successful transfer those images back to Earth [Castano et al., 2008].

Reliable robots are needed, not just for space but also in other areas. Especially when one takes the robots out of their safety cages, or prepared settings on the factory floors. It has proven hard to make robots that can work around humans. This is of particular interest for domestic robots. While there are nowadays already millions of robot vacuum cleaners wandering the homes of people, they are not very autonomous or adaptable. They still are rather 'dumb' and require a 'safety fence' (defining the area of operation) and cannot deal with with a variety of situations encountered in a normal household, such as, changes of floor types (carpet/hardwood/...), pets, dirty laundry left on the floor. The use of more complex robots in household settings has been proposed, for example, ranging from cleaning tasks, grocery shopping to helping in care facilities and even hospitals. These involve the robots working around humans – assisting and coexisting – in daily life situations. The challenges within these scenarios stem form the need to adapt autonomously to the behaviour of dynamic, hard-to-predict entities. In addition these environments tend to be made for humans, unlike on the factory floor, where a focus is put on making the setting robot-friendly, for example, fixed lighting, no-go zones for humans, semi-fixed positions, and so forth.

State-of-the-art humanoid robots such as Honda's Asimo [Sakagami et al., 2002], NASA's Robonaut and R2 [Bluethmann et al., 2003; Diftler et al., 2011], Toyota's Partner Robots [Takagi, 2006], Justin and TORO [Ott et al., 2012] from the German Aerospace Center (DLR), Atlas by Boston Dynamics (an iteration of PETMAN [Nelson et al., 2012]), and not least the *iCub* [Tsagarakis et al., 2007] are stunning features of engineering (Figure 1.4).

These robots are capable of producing complex, repeatable motions allowing them to walk, run and manipulate delicate objects such as musical instruments [Kusuda, 2008; Solis and Takanishi, 2011]. The caveat is that every last detail of these behaviours is currently programmed by hand (often with the help of advanced tools, such as for example, Aldebaran's *Choregraphe*). As a result those resourceful machines, full of capabilities, are not yet realising their full potential due to the complexity of programming them in a flexible way. At the same time the development of common platforms leads to robust and reusable techniques for certain sub-problems, such as, basic perception, navigation – including localization and mapping – and even control and manipulation. There is a common consensus that the next big leap in robotics will come by adding higher capacity to adapt for the robots to changes in the environment or to unexpected circumstances.

Current state-of-the art robotic systems are still very limited in their abilities to adapt. Along with the concept of adaptiveness comes the notion of acting autonomously. To exploit the full versatility of an advanced robot, a broad spectrum of actions is required. In recent years the interest of the robotics community in Artificial Intelligence and Machine Learning techniques as tools to build *intelligent robots* has increased. At the same time, the ML community has shown an increased interest in robots as an ideal test bench and application for new algorithms and techniques [Konidaris, 2013]. It uses these techniques, and tests them, in a physical entity and real environments to interact/move in.

This 'embodiment' – i.e., the claim that having a body that mediates perception and affects behaviour plays an integral part in the emergence of the various parts of human cognition – is seen as an important condition for the development of cognitive abilities in robots (see Chapter 2.1).

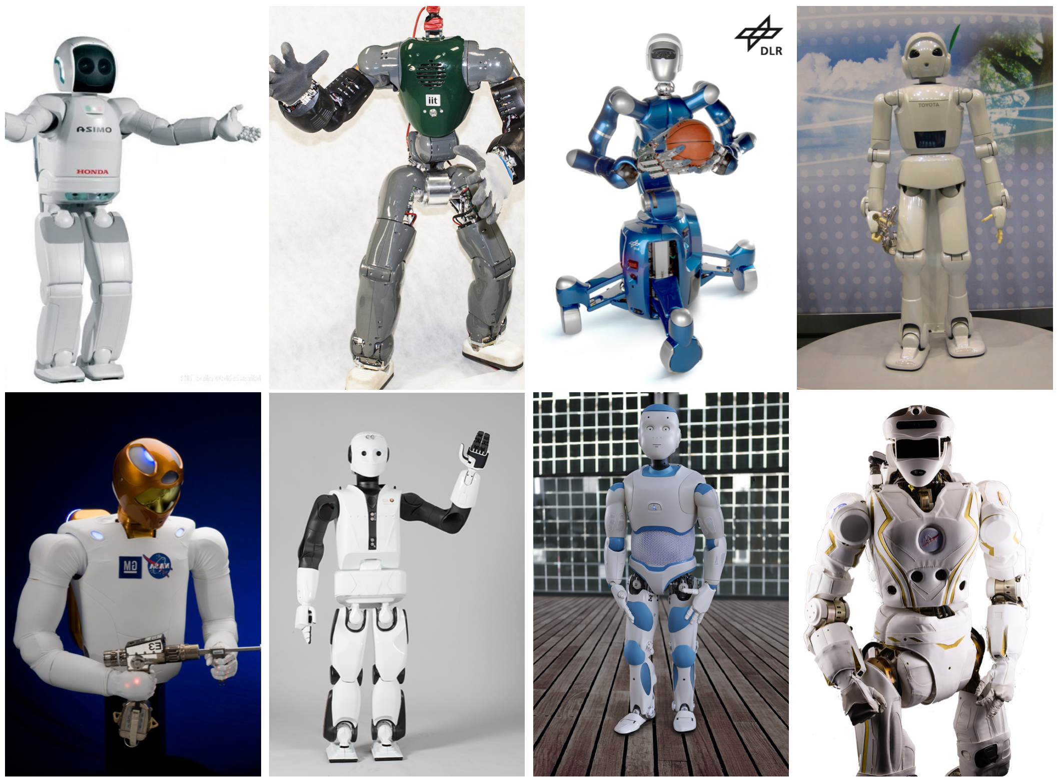


Figure 1.4. Examples of state-of-the art humanoid robots: Asimo (Honda), Coman (IIT), Justin (DLR), Partner (Toyota), Robonaut R2 (NASA/GM), REEM-C (PAL), Romeo (Aldebaran), Valkyrie (NASA).

Pushing beyond the state-of-the art in real-world applications

In many industrial nations robots are performing a vast selection of automation tasks. These though are performed in very well-defined, clean environments, such as factory floors. In the last years a clear trend to create more and more adaptive and autonomous robotic systems. The aim to push beyond the state-of-the art robotic systems and away from industrial robotics can clearly be seen when looking at the research projects funded all over the world.

Already mentioned briefly above DARPA in the US is funding research projects to push beyond the state-of-art in robotics. Mainly with the goal to build systems that can actually perform a variety of operations in a wide range of scenarios. It is worth mentioning that although DARPA is a US government body, international participation in trials and projects is quite common and encouraged. The recent DARPA grand challenge is open to roboticists from all over the world, and the Japanese team SCHAFT won the first round of trials. The finals are scheduled for mid-2015.

The appearance of robotic systems, as envisioned in books and movies, was followed with the disappointment that the real-world systems were far away from the described marvels in fiction. They do certain things that humans cannot do for themselves, like exploring space, or perform specific tasks better than humans, like picking up thousands of pieces from a conveyor belt every day. Yet reliable robot systems – especially ones similar to those described in fiction – have not yet appeared. It has proven to be hard to build autonomous systems that can deal with the uncertainty of the sensing and other issues of the real-world. Although there are around 10 million robotic vacuum cleaners roaming the floors, these robots seem still pretty stupid.

To push beyond the state-of-the art for fully integrated robotic systems DARPA has had a variety of research calls, grants and projects. The list of DARPA sponsored robotics projects is long, with the most prominent in the last years being:

- DARPA Urban and Grand Challenges, which highlighted the autonomous driving capabilities of cars, (2004-2012)[2]

- Revolutionizing Prosthetics program, leading to the DEKA/"Luke" Arm project, which developed dexterous, prosthetic, robotic devices for lower arm amputees, (2006-2014)

[2]Prior cars had shown impressive features of self-driving: driving for hundreds of kilometres on paved, structured highways, at high speeds, with quickly moving obstacles, solely relying on vision [Dickmanns, 1997].

- BigDog/LS3 robot, which was developed as a legged support system for military squads ('robotic mule'), (first contract in 2009)
- Atlas project, which created a bipedal humanoid (unveiled 2013) to be used in,
- DARPA Robotics Challenge, which aims to improve the use of robots in disaster response scenarios (2012-2015)

For robots to become useful and perform far wider ranges of interesting tasks, they do not necessarily have to become fully autonomous. Robots are far away from mimicking human-like perception and action skills, yet they can be seen as extending human capabilities. One thing that will be seen in the future is that, more and more humans will come and work together with robots to fulfil tasks they cannot do alone, or otherwise do so easily.

There are currently still no robotic systems available that perform autonomous dexterous manipulation (or do so only in very limited settings). The fields of robotics and computer/robotic vision have recently seen a steep increase of dedicated funding. In the US DARPA is investing a lot of resources (as mentioned above), in Europe a recent funding scheme was started ("SPARC") and referred to as "the biggest civilian robotics research programme in the world".[3] Previously projects like, for example, eurathlon, were trying to test robotic systems outside of lab conditions, in the real world. In the last year the EU funded two more projects that are focussing on developing robotic systems for real-life applications, namely

- Eurathlon[4]: a outdoor robotics competition, which invites teams to test the intelligence and autonomy of their robots in realistic mock emergency-response scenarios
- RoCKin[5]: split into a work and home scenario, this project aims to build more robust, dependable robots
- EuRoC[6]: which focuses on developing smarter industrial robots and building connections between industry and research

Also in Australia the government recently provided funding for the creation of a Centre of Excellence in Robotic Vision[7], to "solve robotic vision once and

[3]`http://sparc-robotics.eu/`
[4]`http://www.eurathlon.eu/site/`
[5]`http://rockinrobotchallenge.eu`
[6]`http://euroc-project.eu/`
[7]`http://roboticvision.org/`

for all", following their investment into the Centre of Excellence for Autonomous Systems[8], which ran until 2010.

At IDSIA the *iCub* robot, a state-of-art high degree-of-freedom (DOF) humanoid, is available (see Figure 1.5) which was specifically designed for object manipulation research in robotic systems. The history of robotics shows that the expectations tend to be higher than what is currently possible. This is even more so the case when using humanoid robots. The *iCub* is far from having the capabilities described in science-fiction. While there are many topics to address in research to build 'intelligent robots', the overarching goal of this research is to extend the capabilities of our experimental platform. Particularly the interest lies in enabling a better perception and a tighter integration of the perception and actuation sides for more autonomous and more adaptive behaviours.

[8]http://www.cas.edu.au/

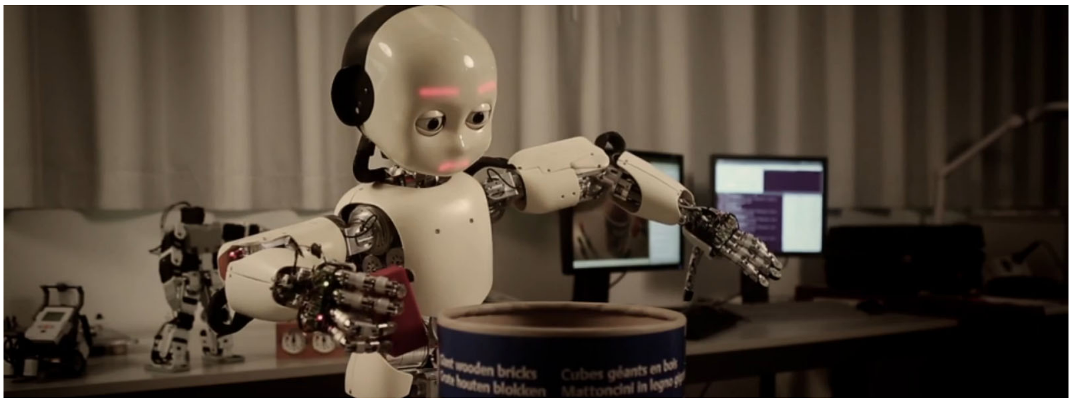


Figure 1.5. Our research platform, the *iCub* humanoid robot.

Chapter 2

Towards Autonomous Object Manipulation in (Humanoid) Robots

One of the most important problems in robotics currently is arguably to improve the robots' abilities to understand and interact with the environment around them: a robot needs to be able to perceive, detect and locate objects in its surrounding and then be able to plan and execute actions to manipulate these.

To decide and act in unstructured environments, the robot needs to be able to perceive its surroundings, as it is generally not feasible or possible to provide the robot with all the information it needs a priori. This might be because of hardware limitations, such as limited storage space, or because the information is simply not available or unknown at the time (the most striking example for this is robotic space exploration). Therefore to extend robotic applications the need for more autonomous, more intelligent robots arises. Sensory feedback, that is the robot's perception, is of critical importance. Creating a useful perception system is still a hard problem, but required for acting in a purposeful, some might say 'intelligent', way. In order to work naturally in human environments robots will need to be much more flexible and robust in the face of uncertainty and incomplete or previously unseen observations. The main aim is to answer the question, whether it is possible (and if so how) for a humanoid robot to use vision and visual feedback to improve, not just its perception, but also its reaching, grasping and (general) manipulation skills. Sensory information collected might be incomplete or inconsistent demanding means of managing uncertainty. Research in the fields of AI and ML has extensively explored ways of dealing with incomplete information in the last decades. On the other hand a wide range of sensors have been used to build models of the environment the robot is placed in. Yet visual feedback, though it tends to be harder to interpret than other sensory information, is a very promising and active research area. A good motivation is that our world is built around (human) visual perception, which also means that to allow 'natural' interaction of robots it needs to be able to understand its environment based on the camera images it receives.

Another aspect that makes human interactions seem 'natural' is our capability of adapting to changing circumstances during action execution. In a robot context this is important, as even if the environment can be perceived precisely, it will not be static in most (interesting) settings. For this reason the robot needs to embody a certain level of adaption also on the motor side. This flexibility could again be provided by AI and ML techniques, leading to robots capable of reaching, grasping and manipulating a wide range of objects in arbitrary positions. To enable more autonomous object manipulation, more specifically how to enable some level of eye-hand coordination to perform actions more successfully, is of high interest to the robotics community (see e.g. NASA's Space Technology Roadmap calls for "Real-time self-calibrating hand-eye System" [Ambrose et al., 2012]).

2.1 Background and Related Work

Object manipulation in real-world settings is a very hard problem in robotics, yet it is one of the most important skills for robots to possess [Kemp et al., 2007]. Through manipulation they are able to interact with the world and therefore become useful and helpful to humans. Yet to produce even the simplest human-like behaviours, a humanoid robot must be able to see, act, and react continuously. Even more so for object manipulation tasks, which require precise and coordinated movements of the arm and hand. The understanding of how humans and animals control these movements is a fundamental research topic in cognitive- [Posner, 1989] and neuro-sciences [Jeannerod, 1997]. Despite the interest and importance of the topic, e.g. in rehabilitation and medicine, the issues and theories behind how humans learn, adapt and perform reaching and grasping behaviours remain controversial. Although there are many experimental studies on how humans perform these actions, the development of reaching and grasping is still not fully understood and only very basic computational models exist [Oztop et al., 2004]. Vision is seen as an important factor in the development of reaching and grasping skills in humans [Berthier et al., 1996; McCarty et al., 2001]. For example, imitation of simple manipulation skills has been observed already in 14-month-old infants [Meltzoff, 1988]. Current robots in contrast are only able to perform (simple) grasps in very limited, specific settings.

Artificial Intelligence, Machine Learning and Robotics

The research in the fields of Artificial Intelligence (AI) and robotics were strongly connected in the early days, but have diverged over the last decades. Although AI techniques were developed to play chess on a level good enough to win against (and/or tutor) the average human player [Sadikov et al., 2007], the robotic manipulation of a chess piece, in contrast, the creation of intelligent machines has lacked quite a bit behind the algorithmic side. It is still not feasible to control a robot on a similar level of precision, adaptation and success as a human — not even comparative to children level. To produce even the simplest autonomous, adaptive, human-like behaviours, a humanoid robot must be able to, at least:

- Identify and localize objects in the environment, e.g. the chess pieces and board

- Execute purposeful motions for interaction, e.g. move a piece to a desired position

At the beginning of AI research a clear goal was to build complete, intelligent, autonomous robotic system [Russell and Norvig, 2010]. As with the example of the above example of chess, it has proven to be quite challenging. Not helping the cause was the fractioning of the fields into many distinct facets of research. While there was progress in each of the sub-fields and the both disciplines (AI and robotics) separately, it has now become clear that a closer integration is again needed. There has been a renewed interest, from both research communities, to work together again towards the goal of intelligent robotic systems.

The field of robotics has clearly matured over the last few years. Current humanoid robots are stunning feats of engineering as mention above. To embed this systems with some sense of 'intelligence' and use the full versatility of advanced robotic systems, a bigger collaboration with the research community in Artificial Intelligence and Machine Learning is required.

The idea of the 'embodied mind' stems from philosophy. It claims that the nature of the human mind is determined by the form of the human body. Philosophers, psychologists, cognitive scientists, and artificial intelligence researchers who study embodied cognition and the embodied mind argue that all aspects of cognition are shaped by aspects of the body. The embodied mind thesis is opposed to other theories of cognition.

Embodied cognition reflects the argument that the motor system influences our cognition, just as the mind influences bodily actions. Roboticists have argued that to understand intelligence and build artificial system that comprise intelligence can only be achieved by machines that have both sensory and motor skills. Furthermore they need to be interacting with the world through a body. This ‘embodiment’ is seen as an important condition for the development of cognitive abilities both in humans and robots [Brooks, 1999; Wolpert et al., 2001; Pfeifer et al., 2007]. The insights of these robotics researchers have in return also influenced philosophers.

Machine Learning algorithms, have been applied in experimental robotics to acquire new skills, however the need for carefully gathered training data, clever initialization conditions, and/or demonstrated example behaviours limits the autonomy with which behaviours can be learned. To build robots that can perform complex manipulation skills that help users in their activities of daily living (ADL) is the aim of various research projects [Cangelosi et al., 2008; GeRT, 2012; THE, 2012; WAY, 2012].

Robot Learning: Cognitive, Developmental and Evolutionary Robotics

As mentioned above the programming of these highly complex robot systems is a cumbersome, difficult and time-consuming process. Current approaches tend to describe each precise movement in detail, allowing little to no flexibility or adaptation during execution. This obviously has issues with scaling to highly complex robots in complicates settings. Therefore the robotics community has focused on methods to provide robots with the ability to act autonomously, adapt or ‘learn’ how to behave without the need of hard-coding every possible outcome.

Autonomous robots research is aimed at building systems that do not require the pre-programming of every possible situation encountered. Many kinds of robots have some degree of autonomy and different robots can be autonomous in different ways. In fields, such as, for example space exploration, a high degree of autonomy is desirable. An autonomous robot might through learning also acquire new capabilities to adapt to changing environments. In resemblance to the “Three Laws” by Asimov, autonomous robots refer to systems that have the following abilities [Chu, 2011]:

- Gain information about the environment (Rule #1)
- Work for an extended period without human intervention (Rule #2)
- Move either all or part of itself throughout its operating environment without human assistance (Rule #3)
- Avoid situations that are harmful to people, property, or itself unless those are part of its design specifications (Rule #4)

- Maintain its own survival at the expense of the previous rules (Sentient Robot Mandate) (Rule #5)

- Learn or gain new capabilities like adjusting strategies for accomplishing its task(s) or adapting to changing surroundings (Rule #6)

In the early 90s of the last century *Behavioural Robotics* (or behaviour-based robotics) was introduced as a way to deal with more and more complex robots and application areas [Brooks, 1991]. This research area focuses on flexible switching mechanisms to change the robots main behaviours based only on a very simple internal model. The basic idea is that close (and probably simple) sensor-motor connections can result in behaviours that appear complex and sophisticated. Due to the fact that these models used a simple approach, rather than a computational complex model and the relatively low cost of development, popularised this approach in the mid-1990s. This paradigm has had a wide range of application in multi-robot teams [Balch and Arkin, 1998] yet the scaling to complex robots, such as humanoid, has not been successful so far.

Robot Learning generally refers to research into ways for a robot to learn certain aspects by itself. Instead of providing all information to the robot a priori, for example, possible motions to reach a certain target position, the agent will through some process 'learn' which motor commands lead to what action. The research field is placed at the intersection of machine learning and robotics and studies how robots can acquire new skills through experimentation. The earlier mentioned 'embodiment' plays an important role here. Example include the learning of sensorimotor skills (for example locomotion, grasping, object manipulation), as well as interactive skills such as manipulation of an object in collaboration with a human. In addition the learning of linguistic skills, especially the grounding of words or phrases in the real world, is of interest to the research community. The field of 'robot learning' is closely related to other disciplines, for example, adaptive control. Learning in realistic environments requires algorithms that can deal with high-dimensional states, e.g. to detect events in the stream of sensory inputs, change and uncertainty. Note that while machine learning is nowadays often used for computer and robot vision tasks (like in this dissertation), this area of research are usually not referred to as 'robot learning'. The fields of *Cognitive Robotics*, *Developmental Robotics* and *Evolutionary Robotics* emerged with the specific aim to investigate how robots can 'learn' for themselves and thereby generate more autonomous and adaptive capabilities.

In *Cognitive Robotics* [Asada et al., 2001] the aim is to provide robots with cognitive processes, similar to humans and animals. An integrated view of the body is taken, including the motor system, the perceptual system and the body's interactions with the environment. The acquisition of knowledge, may it be through actions (e.g. motor babbling) or perception is a big part of cognitive robotics research. Another is the development of architectures for these tasks. A variety has been proposed [Burgard et al., 2005; Shanahan, 2006; Vernon et al., 2007a; Chella et al., 2008; Wyatt and Hawes, 2008], but the promised improvements in robotic applications still need to be shown. This can be attributed to the varying definitions of cognition and the complex human cognitive system, whose workings are still not fully understood. To build cognitive architectures two distinct approaches have been tried. The research seems to mainly focus on top-down architectures. A bottom-up approach has been described as more suitable for the use with robots (e.g. the proposed *iCub* cognitive architecture [Vernon et al., 2007b]).

Developmental Robotics [Asada et al., 2001; Weng, 2004; Kuipers et al., 2006; Meeden and Blank, 2006; Asada et al., 2009] is aiming to put more emphasis on the development of skills. It is an interdisciplinary approach to developmental science. It differs from the previous approaches, as the engineer only creates the architecture and then allows the robot to explore and learn its own representation of its capabilities (sensory and motor) and the environment. As above, the body and its interactions with the environment are seen as being fundamental for the development of skills. Aims are to build adaptive robotic systems by exploration and autonomous learning, i.e. learning without a direct intervention from a designer [Lungarella et al., 2003]. Here interesting areas to explore are selected by building on previous knowledge, while seeking out novel stimuli.

Evolutionary Robotics [Harvey et al., 1997; Nolfi and Floreano, 2000; Doncieux et al., 2011] is another approach to add adaptiveness and developmental processes to robots. It emerged as a new approach to overcome the difficulties of designing control systems for autonomous robots: (a) coordinating the (increasing) number of DOF both in mechanics and control is hard, especially since the complexity scales with the number of possible interactions between parts (see 'Curse of Dimensionality' [Cliff et al., 1993]) (b) the environment and how the robot interacts with it are often not known before. Its main focus is on evolve a control system based on artificial neural networks. These neuro-controllers (NC), inspired by the neuron activity in the human brain, have been shown to work in a wide range of applications [Nolfi et al., 1994; Dachwald, 2004; Leitner et al., 2010]. An important issue is that to 'learn' behaviours, a large number of iterations (or generations) is required. This works fine in simulation but is hard to achieve on a real robotic platform. Nolfi et al. [1994] showed that evolving a NC on hardware is, while time consuming, feasible, at least for simple mobile robots. Hybrid approaches, where NCs are trained first in simulation and then transferred to the real hardware, seem preferential. The performance of the controllers in the real world can then be used to improve the simulation [Bongard et al., 2006]. How to effectively train and apply NCs to real, high-DOF hardware is still an open research question.

Other Approaches to robot learning have been developed in the past. The area of Reinforcement Learning (RL) [Sutton and Barto, 1998; Kormushev et al., 2013] has appealed to many roboticists, especially for learning to control complex robotic systems. A general RL algorithm and the means to inform the robot whether its actions were successful (positive reward) or not (negative reward) is all that is required. RL and its applicability to humanoid robots has been investigated by Peters et al. [2003]. Imitation Learning or Apprenticeship Learning is of importance in human skill development as it allows to transfer skills from one person to another. In robotics Robot Learning from demonstration or Programming by Demonstration is a similar paradigm for enabling robots to learn to perform novel tasks. It takes the view that an appropriate robot controller can be derived from observations of a another agent's performance thereof [Schaal, 1999]. This approach though requires a prior to start with, usually a human controlling the robot to perform the given task. More formally the training examples obtained are limited by the performance of the teacher, which can lead to suboptimal performance. While learning the reward function can lead to adaptation to changes in the task, like e.g. new goal states, this though is quite limited especially in highly-complex robots, where finding how to control the robot is an issue.

2.2 Understanding the Environment

To be useful in the above proposed scenarios a robot must be able to see, act, and react continuously. Perception is a key requirement in order to purposefully adapt robot motion to the environment, allowing for more successful, more autonomous interactions. The first important step towards this is to understand the environment the robot is embedded in. Coming back to the example of playing chess, this would compare to finding the chess board and each of the chess pieces (e.g. in a camera image) or even just to to realise that there is a chess board and pieces in the scene.

Vision and the visual system are the focus of much research in psychology, cognitive science, neuroscience and biology. A major problem in visual perception is that what individuals 'see' is not just a simple translation of input stimuli (compare *optical illusions*). One important area of research to build robots that can understand their surroundings is the development of artificial vision systems.

Computer Vision – sometimes referred to as *Robot Vision* when applied in a robotic system – generally describes the field of research dealing with acquiring, processing, analysing, and understanding images in order to produce decisions based on the observation. The fields of computer vision and AI have had close ties in the beginning, but have gone separate ways over the last decades. A trend to reunite those fields again is emerging.

While there does not exist a clear definition of the areas of computer vision and image processing, the latter is commonly seen to refer to a subsection of computer vision. Image processing techniques generally provide ways of extracting information from the image data and can be grouped into the following categories: pre-processing (e.g. noise reduction, enhancement, scaling, etc.), feature extraction (e.g. lines, edges, interest points, etc.), segmentation (e.g. separating fore- and background), and high-level processing (e.g. recognition and decision making) [Gonzalez and Woods, 2006]. Another important topic in computer vision is 'image understanding'. With the aid of geometry, physics, statistics, and learning the goal is to mimic the abilities of the human (visual) perception system.

Research into vision for the special requirements of robotic systems is referred to as *robot vision* or *machine vision* [Horn, 1986; Hornberg, 2007]. For example, visual feedback has extensively been used in mobile robot applications, for obstacle avoidance, mapping and localization. With the advancement of humanoids and the increased interest in working around humans, object detection and manipulation are more and more driving the development of robot vision systems. An important problem is that of determining whether or not the image data contains some specific object, feature, or activity. While this has been researched for quite some time already, the task seems harder than expected and no solution for the general case of detecting arbitrary objects in arbitrary situations exists. From a robot vision point of view, this means that the robot is required to detect previously unknown objects in its surroundings and be able to build models to memorise and identify them in the future. Most of the work is heavily relying on artificial landmarks and fiducial markers to simplify the detection problem. Furthermore existing methods can at best solve it for specific objects (simple geometries, faces, printed or hand-written characters, or vehicles) and in specific situations (in terms of well-defined illumination, background, and pose of the object wrt. the camera). For a detailed introduction and overview of the foundations and the current trends the reader is referred to the excellent survey by Kragic and Vincze [2009].

2.3 Interacting With the Environment

Computer vision has become a more and more prominent topic of research over the past decades, also in the field of robotics. Like humans and animals, robots are able to interact with the world around them. While most robot vision research tends focus on understanding the world from just passive observations, these interactions with the environment provide and create valuable information to build better visual systems. Connecting manipulation commands with visual inputs allows for a robot to create methods to actively explore its surroundings. These connections between motor actions and observations exist in the human brain and are an important aspect of human development [Berthier et al., 1996].

Only after the scene is observed and the robot has an idea about which objects are in the environment, can it start interacting with these in a safe fashion. In the chess example, even if the state of the board and where it is located are known, to move a certain chess piece from one field to another without toppling other pieces is still a hard problem by itself. In fact, children even at a very young age, have significantly better (smoother, more 'natural', 'fluent' and controlled) hand movements than all currently available humanoid robots. But manipulating arbitrary objects is not a trivial thing, even for humans. The development of hand control in children, for an apparently simple, prototypical precision grasp task is not matured until the age of 8-10 years [Forssberg et al., 1991]. Moreover, complexity, as can be seen by the number of neurons comprising the control of the arm and hand, is staggeringly high. Even after manipulation skills have been learnt they are constantly adapted by an perception-action loop to yield desired results. In infants various specializations in the visual pathways may develop for extracting and encoding information relevant for visual cognition, as well as, information about the location and graspability of objects [Johnson and Munakata, 2005]. This hints at the very close integration of vision and action in the human brain.

In recent years good progress was made with robotic grasping of objects. The various manipulators, mainly hands and grippers, and techniques clearly improved. Also novel concepts of 'grippers' have been designed and some are quite ingenious solutions to a number of issues. One such example is the granular gripper made by Brown et al. [2010], which is made out of grounded coffee beans which are able to 'flow' around the object and then fixed in position by creating a vacuum. This concept has recently been extended to a full sized elephant-trunk-style arm [Cheng et al., 2012]. Also in terms of how to grasp objects with regular grippers and 'hands' recent results highlight the advanced state of research in grasping. For example, Maitin-Shepard et al. [2010], with their research showed that robots are able to pick up non-rigid objects, such as, towels. Their robot is able to reliably and robustly pick up a randomly dropped towel from a table by going through a sequence of vision-based re-grasps and manipulations-partially in the air, partially on the table. In the DARPA ARM project, which aims to create highly autonomous manipulators capable of serving multiple purposes across a wide variety of applications, NASA's JPL winning team showed an end-to-end system that allows the robot to grasp diverse objects (e.g. power drill, keys, screwdrivers, ...) from a table [Hudson et al., 2012; Hebert et al., 2012]. On the other hand Saxena et al. [2008] have presented a way for a robot to learn, from only a small number of real world examples, where good grasping points are on a wide variety of previously unknown objects.

All this has lead Dr. Pratt, manager of robotics related research projects at DARPA, to his somewhat controversial statement of "Grasping is solved"[1] at IROS in 2012. While this might be a bit too optimistic it seems like the research is at a good enough state to have better system integration. The direct interface between various components, which makes robotics such a hard but interesting field, clearly needs to improve to allow for robust object manipulation. Only by combining sensing and control of the whole robotic platform a fully functional 'pick-and-place' capable system will appear. To allow for a variety of objects to be picked up from various positions the robot needs to see, act and react within a control system in which these elements are tightly integrated.

A must read for roboticists is 'Robotics, Vision and Control' [Corke, 2011]. It puts the integration of these three components in the spotlight. Also it describes common pitfalls and the issues that arise with integration.

2.4 Proposed Approach

The goal of this dissertation is to improve the skills of the *iCub* humanoid. To create better 'perception', 'motion' and 'coordination' and deal with uncertainties I use various machine learning (ML) and artificial intelligence (AI) techniques to support both perception and movement. IDSIA is investigating how rewards and motivation effect the development of complex actions and interactions between an (embodied) agent and the environment.[2]

My aim is to improve adaptivity and autonomy in robot grasping based on visual feedback to close the loop and perform grasping of objects, while adapting to unknown, complex environments. The main goal is to answer the question,

[1]Plenary at IROS 2012: `http://www.iros2012.org/site/node/13`

[2]Funded mainly by the European Union grant 'IM-CLeVeR' [Baldassare et al., 2009].

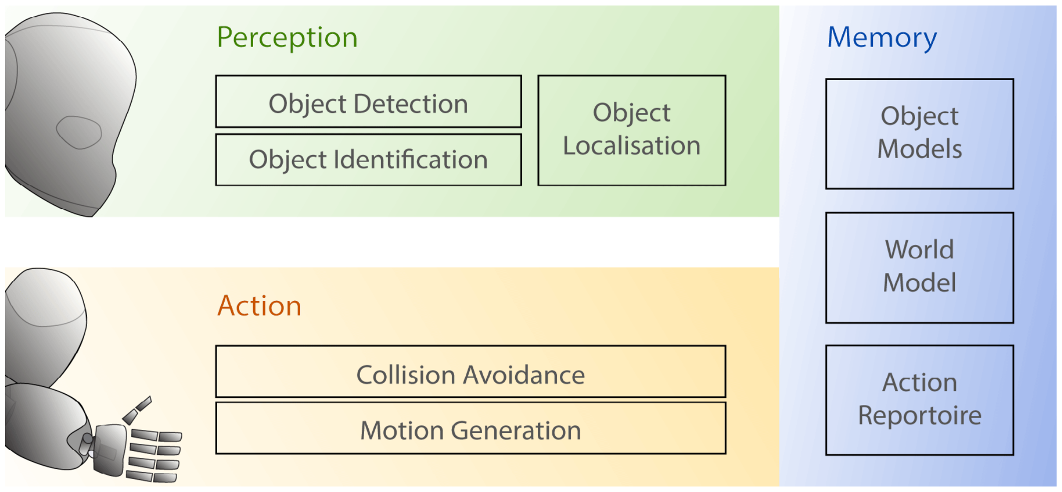


Figure 2.1. Overview of IDSIA's current research towards a functional eye-hand coordination on the *iCub* humanoid. The *object detection* and *identification* is currently solely based on the camera images (2D) received. The *object localization* uses the information from the two cameras to calculate an operational space (3D) position. This is the same space in which *collision avoidance* is applied and the *world* is modelled. The *object model* contains the information of how to detect the object in the 2D images (see Section 3.3.1) and a fixed 3D geometry. The *motion generation* and *action repertoire* can use the full configuration space of the humanoid (41 DOF).

whether it is possible (and if so how) for a humanoid robot to use vision and visual feedback to improve, not just its perception, but also its reaching, grasping and (general) manipulation skills.

At first a rough sketch of subsystems and their interfaces was designed. Over the course of this dissertation it evolved into what is shown in Figure 2.1, which provides an overview of the modules built for eye-hand coordination capabilities on the *iCub*. Chapter 3 describes in more detail the background and previous approaches in computer and robot vision. It also contains the description of the robot vision frameworks created and techniques developed for and implemented on our *iCub*. Building on existing YARP infrastructure these frameworks provide the tasks shown in the top row of the figure (green), namely the modules for the detection and identification of objects (in the images), as well as, the localization (in 3D Cartesian space). The new architecture allows not just for a simple reusable object detection, but also provides a simple way of learning these based on only very small training sets. In connection with *icVision*, data can be collected on the real hardware and the learned results directly executed.

The bottom half of the diagram, in yellow, shows the action and motion side. To generate motion using machine learning techniques a crucial feature is avoiding collisions, both between the robot and the environment *and* the robot and itself. Chapter 4 describes the developed techniques to control the *iCub* at IDSIA in the last few years. It includes some background and previous approaches in this area. The first part of this chapter describes the modules of collision avoidance and motion generation, which were developed mainly by my colleagues. My contribution was to make all these modules work together, especially with the visual inputs; more details can be found in Frank [2014]. Building upon those systems human motion is investigated as a way of tele-operating and interacting with the complex humanoid.

All these subsystems are supported by memory (in blue) enabling the persistent modelling of the world and providing a repertoire of actions. Again how to interface these systems, and the various requirements are of importance, as memory plays a crucial part for the integration of the whole system and building autonomous humanoids.

Chapter 5 describes how these two sides, the perception and the motion, are integrated and used together to generate a proof-of-concept for a level of eye-hand coordination not previously seen on the *iCub*.

Most of these are also on-going research areas at IDSIA to further improve the existing frameworks and (sub-)systems.

2.5 Experimental Platform: The *iCub* Humanoid

While I hope that my work can be applied to various robotic platforms, the main focus was to implement the research on the *iCub* humanoid robot [Tsagarakis et al., 2007] (depicted in Figure 1.5). It is an open-system robotic platform developed during various European projects.[3] The robot is based on an anthropomorphic design with dimensions similar to that of a three-and-a-half year old child. In its complete configuration it is currently standing 104cm tall and weighs around 22kg (Figure 2.2).

The *iCub* was designed by several European universities under the coordination of the Italian Institute of Technology (IIT) during the *RobotCub* project. The initial funding was 8.5 million Euro within the European Commission's Seventh Framework Programme (FP7, Unit 5 - Cognitive Systems and Robotics)[4] and ran for sixty-five months from 1 September 2004 until 31 January 2010. The consor-

[3]The official *iCub* website: `http://www.icub.org`

[4]`http://cordis.europa.eu/fp7/ict/cognition/home_en.html`

tium was composed of 10 European research centres (in Italy, Portugal, Sweden, Switzerland, and the UK) and complemented by three research centres in the USA and three in Japan, all specialists in robotics, neuroscience, and developmental psychology.

The robot is a completely open system, with the software and documentation being open-source and even the hardware design released under the GNU General Public License (GPL). The names *RobotCub* (of the initial project) and *iCub* (of the final robot platform) are partial acronyms, with *cub* standing for "Cognitive Universal Body".

The motivation behind the strongly humanoid design is the "embodied cognition" hypothesis, in particular that human-like manipulation plays a vital role in the development of human cognition. A baby learns many cognitive skills by interacting with its environment and other humans using its limbs and senses,

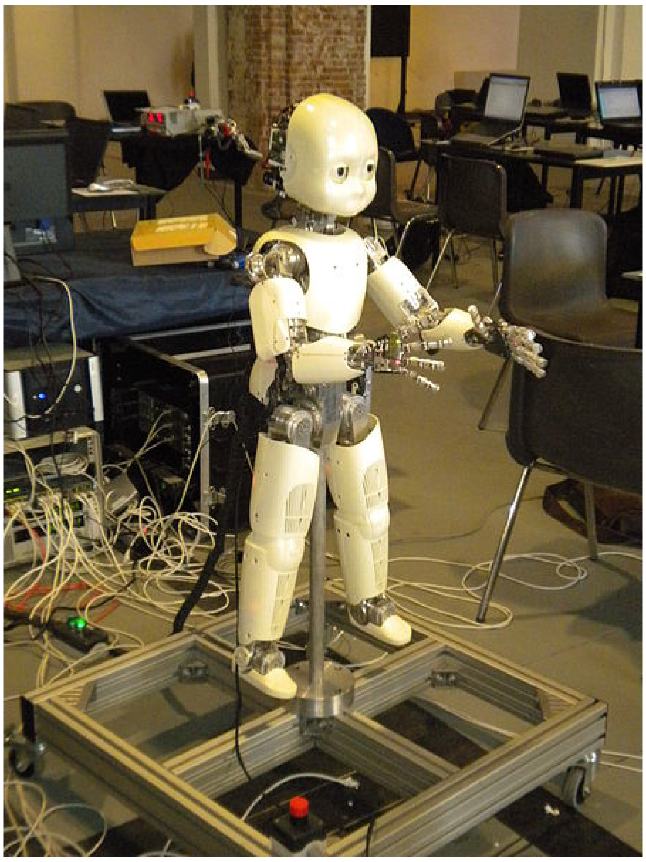


Figure 2.2. The *iCub* in its full configuration mounted on a supporting frame during the *'Veni, Vidi, Vici'* Summer School.

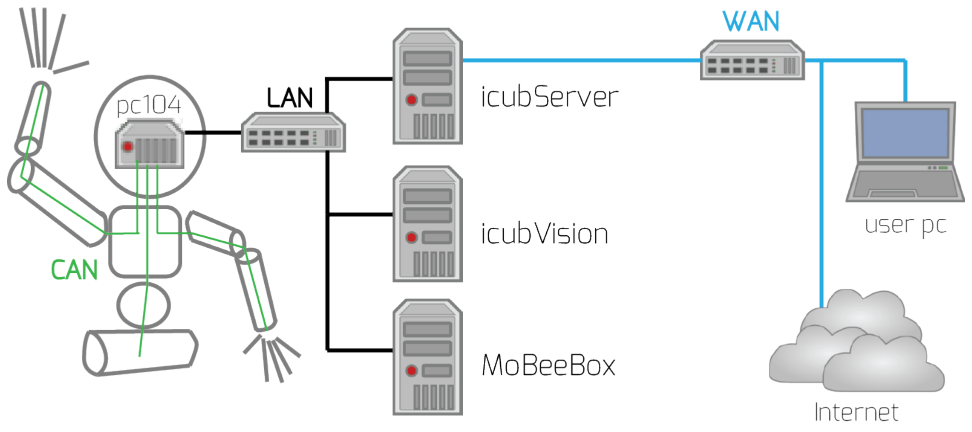


Figure 2.3. The setup used to operate the *iCub* at the IDSIA Robotics Lab. The `pc104` handles the on-board data processing and controls the motors via CAN-bus. The `icubServer` is running the YARP server and is the router into the IDSIA-wide network and the internet. Dedicated computers for vision (`icubVision`) and collision avoidance (`MoBeeBox`) are used.

and consequently its internal model of the world is largely determined by the form of the human body. The robot was designed to test this hypothesis by allowing cognitive learning scenarios to be acted out by an accurate reproduction of the perceptual system and articulation of a small child. In short, the robot was constructed this way so that it could interact with the world in the same way a child does [Metta et al., 2008].

The robot is controlled by an on-board PC104 controller which communicates with actuators and sensors using CANBus. The robot was not designed for fully autonomous operation, and is consequently not equipped with onboard batteries or computational resources necessary for this. Instead an umbilical cable provides power to the robot and a local-area-network (LAN) connection. The software library is largely written in C++ and uses a middleware called YARP (Yet Another Robotics Platform) [Metta et al., 2006]. It provides functionality for the external communication via Gigabit Ethernet with the off-board software implementing higher level functionality. Figure 2.3 sketches the distributed computing system used at IDSIA to operate the robot. More information about setup and configuration, as well as the code base, can be found on the *iCub* Wiki,[5] where researchers, from a large collection of research labs using the robot, contribute and build up a knowledge base.

On the actuation side it uses harmonic drives (with a ratio of 100:1 for all ma-

[5] *iCub* Wiki URL: `http://wiki.icub.org`

jor joints) together with a brushless, frameless motor. To reduce the moment of inertia the motors are placed close to the body and the robot's motion is actually driven by teflon-coated steel tendons (with the exception of the six actuators used for the head). The cables are routed in a rather complex way via idle pulleys to transmit the torque to the joint. Joint angles are measured using Hall-effect sensors and the robot is equipped with force-torque sensor both in the left and right upper arm. The finger tips are equipped with touch sensors, and a distributed capacitive sensor skin is being developed Billard et al. [2013].

The initial design was aiming for complex and dexterous object manipulation and therefore limiting the robot to only be able to crawl on the floor. Since the first robots were constructed the design has undergone several revisions and improvements, for example smaller and more dexterous hands, and lighter, more robust legs with greater joint angles. Other forms of locomotion, mainly bipedal walking rather than just crawling, were only recently considered and mechanical updates are currently tested at IIT and planned to be incorporated in future versions [Tsagarakis et al., 2009]. Various different configurations of the robot are possible, for example, a variety of labs are using just the upper body for investigating object manipulation within their on-going research projects [Metta et al., 2010]. Currently more than twenty *iCub*s are available in various laboratories, mainly in Europe, but also in the USA, Turkey and Japan.

In its current version, the robot has 53 actuated degrees of freedom (DOF) – the setup at IDSIA does not contain legs, therefore reducing the robot to a 41 DOF system. The DOF are distributed the following:

- 7 DOF per arm, 3 for the orientation (roll-pitch-yaw) of the hand, 3 for the shoulder and 1 for the elbow
- 9 DOF per hand (3 for the thumb, 2 for the index, 2 for the middle finger, 1 for the coupled control of the ring and little finger, 1 for adduction/abduction)
- 6 DOF in the head (3 for the neck and 3 for the cameras)
- 3 DOF in the torso/waist
- 6 DOF per leg

A variety of sensors are integrated into the robot. It is one of the key features of the platform, as a complex sensory system enables the investigation of the 'embodiment' theory and algorithms that exploit and integrate different modalities (e.g. sensor fusion, multimodal calibration and multimodal perception).

The sensory system is sketched in Figure 2.4 and consists of the following:

- two microphones installed in the 'ears' (assisted by mechanical structures similar to human earlobes) of the robot
- inertia measurement unit (IMU) in the head, containing three gyroscopes, three linear accelerometers and a compass. It provides a proprioceptive sense of orientation (and angular acceleration, though very noisy and hardly used in practice)
- for proprioception all motors and joints are equipped with angular encoders, as mentioned above.
- custom 6 axis force/torque sensors are mounted in the arms and legs
- tactile sensing is available on the hands based on capacitive technology sensors (12 in each fingertip and 48 on the palm)
- two colour cameras (Pointgray DragonFly), in a swivel mounting are built into the head, where the eyes would be located. These are the main sensors for the work at IDSIA. They each provide a 640x480 pixel colour image which are streamed to the network roughly 12-15 times per second.

In addition it also has lines of red LEDs mounted behind the face panel, representing mouth and eyebrows and allowing for facial expressions. A variety of

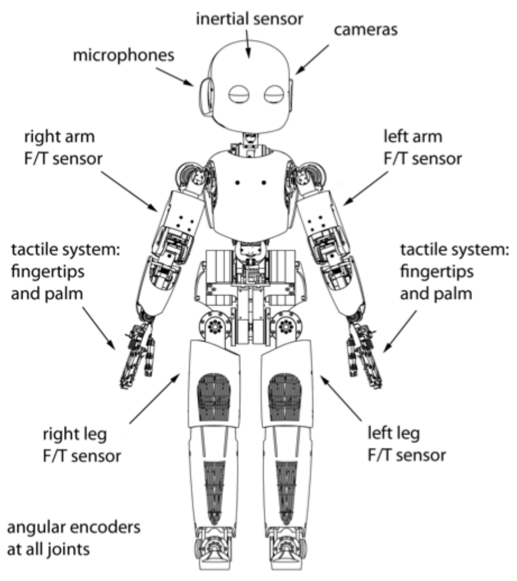


Figure 2.4. A schematic overview of the sensory system of the robot.

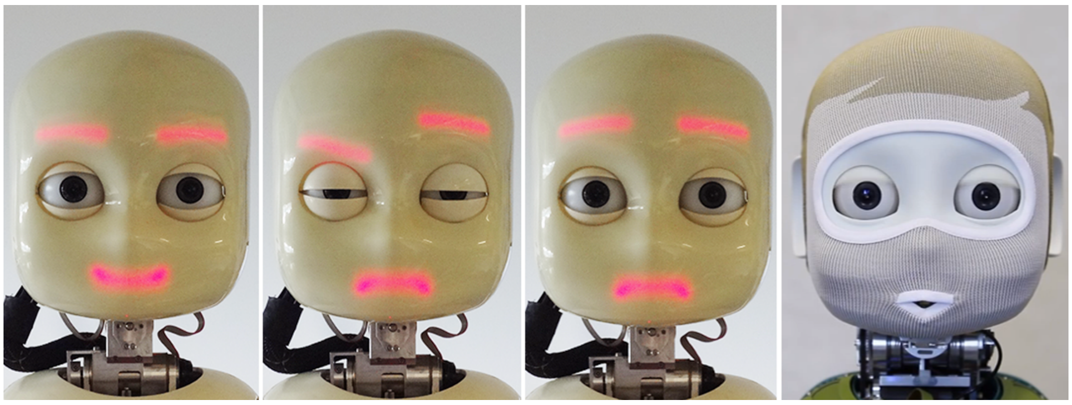


Figure 2.5. The *iCub* provides a variety of facial expressions to be used in human-robot-interaction scenarios. On the right the new face allowing for even higher expressiveness (currently in development at IIT).

expressions are already available and at IDSIA they are extensively used during interactive demos, human-robot interaction research and the shooting of *iCub* research videos. Currently a new head is under development at IIT, which when finished, will allow for an even more expressive face (Figure 2.5).

The *iCub* is an excellent experimental platform for cognitive and sensorimotor development and embodied artificial intelligence [Metta et al., 2010] and was designed to investigate human-like object manipulation, requiring the non-linear control of the physical degrees of freedom. The robot's movements need to be coordinated with feedback from visual, tactile, and acoustic[6] perception. Of interest is also to test a wide variety of learning theories. For example, the development of cognition, especially through interaction with the environment based on intrinsic motivation is an on-going research theme on the *iCub* [Natale et al., 2013; Baldassare et al., 2009]. Furthermore investigating the scalability of machine learning methods towards such complex systems interacting with the real-world is an active research area [Levinson et al., 2010; Sicard et al., 2011; Leitner et al., 2012f; Frank et al., 2014].

[6]Acoustic feedback might be used during object manipulation to assess if a grasp was successful or not.

Chapter 3

Robot Perception

An emphasis during the research presented herein has been put on investigating visual perception on our humanoid robot. The term refers, in humans and animals, to the ability to interpret their environment based on the information from visible light reaching the eyes. In robotic applications, to imitate eyes, cameras are used to capture the light of the environment enabling visual sensing. The reason for the focus on vision is twofold, firstly the sensing capabilities of robotic platforms (cameras are cheap) and secondly, vision is the most important sense for humans. Another reason is that with a humanoid robot, such as the *iCub* at IDSIA, a natural tendency exists to be inspired by human perception and behaviour. A lot of my research work focussed on building a framework for perception that allows rapid prototyping of detectors, as well as, the autonomous learning of these. The system is roughly based on the description of human perception by Marr [1982]. The various experiments performed during the course of my doctoral research are described after a section introducing the basics in computer and robot vision. These experiments have already been published in peer-reviewed conferences or journals (references are provided).

3.1 Background

Perception is a key requirement in order for robots to be useful in a wide range of scenarios, such as the ones proposed and mentioned above. A robot must be able to see, act, and react continuously, to purposefully adapt its motion to the environment, allowing for more successful, more autonomous interactions. An important skill for any robotic system is the ability to sense its environment. This is of value even in tele-operation or other non-autonomous operating scenarios. Its though most valuable when the machine needs to – all by itself –

make sense of what it is sensing. During my research at IDSIA the focus was on understanding the closer environment of the robot, aiming to detect objects the robot is interacting with. This is done by using the only available visual sensors on the *iCub*, two colour cameras (Figure 3.1). Other sensors are available (see Section 2.5), most notably, touch sensors in the finger tips. While other labs have put more focus on tactile feedback (e.g. [Argall et al., 2010]), we aim to better integrate further sensors into our framework in the future and concentrate first on enabling the robot to 'see' the objects it needs to interact with.

Most humanoid robots, have to use cameras to visually perceive the environment, whereas in mobile robots the use of depth-cameras has become the norm. Various systems that provide depth information exists nowadays: ranging cameras, flash LIDAR, time-of-flight (ToF) sensors, and RGB-D cameras. While my focus lies on visual perception based on our robots camera system, a quick overview of alternative sensors and techniques is given here. In mobile robot application laser[1] scanners and LIDAR are commonly used for research in the area of "Simultaneous localization and mapping" (SLAM). This research field aims to allow robots to construct a map of an unknown environment while at the same time keeping track of the robot's location in the environment (a more detailed description, with a focus on visual SLAM, can be found in the "Object Localization" section further down or in Thrun et al. [2002]). Laser scanners are also used in space. For example, to dock with the International Space Station, the spacecraft with resupplies (usually un-piloted), needs to manoeuvre carefully to specific positions along the 'docking trajectory'. A laser scanner is used in order to determine the craft's relative position to the station, to allow for an autonomous approach. LIDAR, which stands for "LIght Detection And Ranging", is an extension to a standard laser scanner which analyses the reflected light more thoroughly. Such systems are used in robotics for a variety of perception tasks, such as, e.g. object classification [Himmelsbach et al., 2008]. This is due the fact that LIDAR technology can provide three-dimensional elevation maps of the surroundings, high precision distance (and velocity) information, an important component for landing robotic spacecraft on planetary surfaces [Amzajerdian et al., 2011]. Recent structured light has been increasingly employed to detect objects and humans in robotic applications. These RGB-D camera systems provide a per pixel depth information (depth image). To calculate the depth an invisible light pattern is projected onto the scene, its deformation as seen by the sensor provides a depth estimate [Fofi et al., 2004].

[1]The term *laser* originated as an acronym for "light amplification by stimulated emission of radiation" but is now commonly used as a noun.

Robot Vision

Animals and humans are exposed to vast amounts of visual inputs in their daily lives. From this they seem to effortlessly extract the relevant knowledge about the world, learn the objects in a complex visual environment without any supervision and recognise them later on. Robots so far have not been able to perform as well with the large amount of visual input data.

Vision and the visual system are in the focus of research in psychology, cognitive science, neuroscience and biology. A major problem in visual perception is that what individuals 'see' is not just a simple translation of input stimuli (compare *optical illusions*). The research of Marr in the 1970s led to a theory of vision using different levels of abstraction, from a two-dimensional visual array (projected onto the retina) to a three-dimensional description of the world as output. The stages include: a 2D sketch of the scene (using feature extraction), a sketch of the scene (using textures to provide more information) and a 3D model of the world [Marr, 1982].

Research on perception has been an active component for developing artificial vision systems, in industry and robotics. *Computer Vision* generally describes the field of research dealing with acquiring, processing, analysing, and understanding images in order to produce decisions based on observations. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from a variety of image data, such as video sequences, views from multiple cameras, or multi-dimensional/spectral data, e.g. from a medical scanner. The fields of computer vision and AI have close connections, e.g. autonomous planning or decision making for robots require information about the environment, which could be provided by a computer vision system. AI and computer vision share other topics such as pattern recognition and learning techniques. Furthermore computer vision spawned a multitude of research sub-domains, such as, scene reconstruction, event detection, object

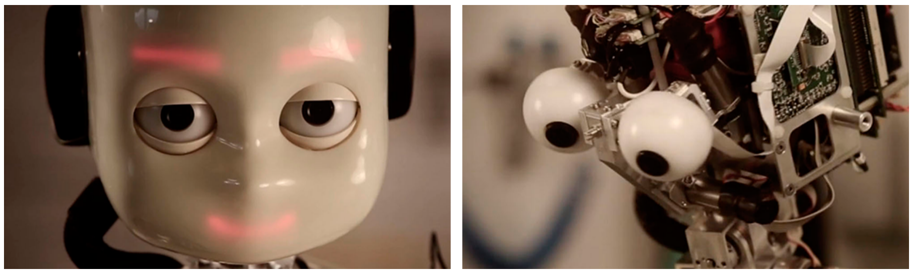


Figure 3.1. The 'eyes' of the *iCub*, with and without face/head cover.

recognition, motion estimation, etc. The following is a short list of examples, where computer vision is applied nowadays:

- Navigation, e.g., by an autonomous vehicle or mobile robot;
- Detecting events, e.g., for visual surveillance or people counting;
- Organising information, e.g., indexing databases of images and videos;
- Modelling objects or environments, e.g., medical image analysis;
- Interaction, e.g., for human-robot interaction (HRI), and
- Automatic inspection, e.g., in manufacturing applications.

In computer vision research a subfield, named 'image processing' [Gonzalez and Woods, 2006], is dealing mainly with the lower level of perception. The techniques in this field generally provide ways of extracting information from the image data and can be grouped into the following categories: pre-processing (e.g. noise reduction, enhancement, scaling, etc.), feature extraction (e.g. lines, edges, interest points, etc.), segmentation (e.g. separating fore- and background), and high-level processing (e.g. recognition and decision making). Another important topic in computer vision is 'image understanding'. With the aid of geometry, physics, statistics, and learning the goal is to mimic the abilities of the human (visual) perception system.

Research into vision for the special requirements of robotic systems is referred to as *robot vision* or *machine vision* [Horn, 1986; Hornberg, 2007]. For example, visual feedback has extensively been used in mobile robot applications, for obstacle avoidance, mapping and localization [Davison and Murray, 2002; Karlsson et al., 2005]. *Active Vision*, another area of research in computer vision, investigates how actively changing the viewpoint of camera(s) can lead to better perception systems [Aloimonos et al., 1988]. One needs to be aware that active vision does not mean active sensing, but merely how the motion of the observer can be included when addressing vision problems. Active Vision does also not include how manipulating the environment can lead to improved performance.

With the advancement of humanoids and the increased interest in working around humans, object detection and manipulation are more and more driving the development of robot vision systems. Furthermore there exists a natural tendency to be inspired by human perception, in which vision plays a primary role, and behaviour, especially when working with humanoid robots. An important problem is that of determining whether or not the image data contains some specific object, feature, or activity. While this has been researched for quite some time already, the task seems harder than expected and no solution for the general case of detecting arbitrary objects in arbitrary situations exists. Most of the work is heavily relying on artificial landmarks and fiducial markers to simplify the detection problem. Furthermore existing methods can at best solve it for specific objects (simple geometries, faces, printed or hand-written characters, or vehicles) and in specific situations (in terms of well-defined illumination, background, and pose of the object wrt. the camera). For a detailed introduction and overview of the foundations and the current trends the reader is referred to the excellent survey by Kragic and Vincze [2009].

Object Detection

To detect an object from vision it is important to know some background information about the physical phenomena and the working of the sensors. Object detection, especially edge and pixel-wise detection is a subjective task. While one can use human separation as ground truth, different observers will generate different 'masks'. This makes a quantitate comparison in real-world scenarios very hard.

Light and Color

The light that reaches our eye or the camera of the robot is not emitted from the object itself but is dependent on the illumination of the scene and the reflectivity of the material. Light itself is an electro-magnetic radiation. Human perception of the radiation is limited to the bandwidth between 400 and 700 nm (4000 — 7000 Å). The spectrum is shown in Figure 3.2, with wavelengths below 400 nm termed ultra-violet and those above 700 nm infra-red. Without going too much into the physical details the most common source of light is the emission from a hot body – such as our sun or a light bulb. This is generally modelled as a blackbody radiator, for example, the Sun has a surface temperature of about 6500 K and a radiation peak in the visible spectrum at about 450 nm.

Our eyes' peak sensitivity has evolved to be aligned very closely to the peak of our sun's spectrum – actually the filtered spectrum due to the Earth's atmosphere. Absorption occurs in all materials, though at different levels. Another important aspect of the light received from an object is its (surface) reflectivity. A long list of material reflectances have been experimentally determined and catalogued. Out of these physical phenomena described the definition of colour and brightness follow. While there exists a physical description in terms of power and wavelength, the latter two are more subjective properties. In Figure 3.2 the colours blue, green, yellow, orange and red are seen in the 400-700nm band. The human perceptual system senses the radiation by using two types of light sensitive cells in the eyes. The retina of has a central or foveal region which is only 0.6 mm in diameter and contains most of the 6 million cone cells. These are unevenly distributed between red, green and blue (roughly 65%, 33% and 2% respectivly). We unconsciously scan our high-resolution fovea over the scene to build a large-scale mental model of the world around us. In addition there are 120 million motion sensitive rod cells distributed over the retina.

Cone cells respond to particular colours and provide us with our normal day-time vision. Rod cells are much more sensitive than cone cells but respond to intensity only and provide the main vision at night. The brightness associated with a particular wavelengths is known as luminosity and is measured in units of lumens per watt. For our daylight cone-cell vision the luminosity as a function of wavelength has been experimentally determined, tabulated and forms a standard, published by the Commission Internationale de L'Éclairage (CIE) in 1931, that represents the average human observer (Figure 3.3 shows the difference between a human and a CCD camera).

The analogue of the retina in computer vision is the light-sensitive sensor inside the camera. It is usually a silicon chip with a multitude of photosites, each

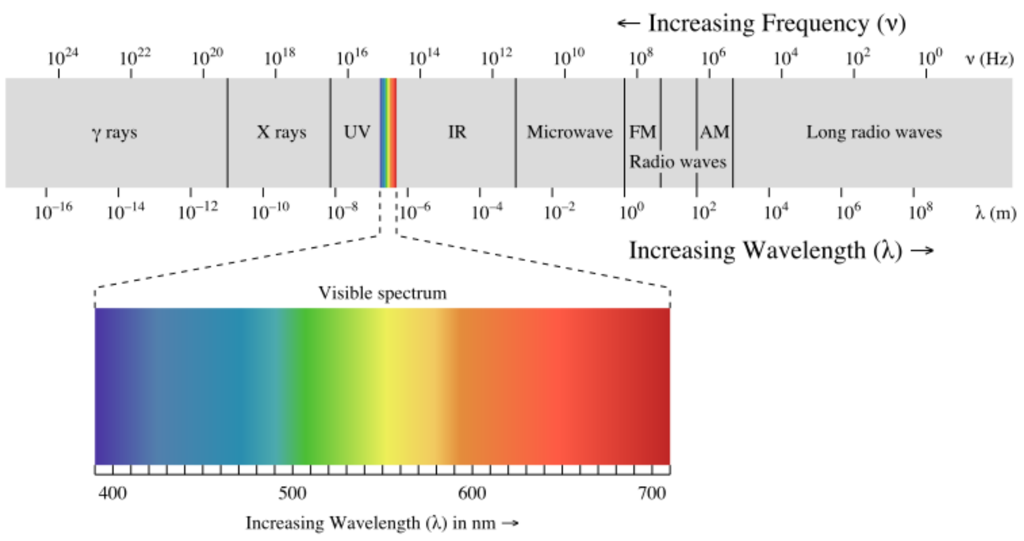

← Increasing Frequency (ν)
10^{24} 10^{22} 10^{20} 10^{18} 10^{16} 10^{14} 10^{12} 10^{10} 10^{8} 10^{6} 10^{4} 10^{2} 10^{0} ν (Hz)
γ rays
X rays
UV
IR
Microwave
FM
AM
Radio waves
Long radio waves
10^{-16} 10^{-14} 10^{-12} 10^{-10} 10^{-8} 10^{-6} 10^{-4} 10^{-2} 10^{0} 10^{2} 10^{4} 10^{6} 10^{8} λ (m)
Increasing Wavelength (λ) →
Visible spectrum
400
500
600
700
Increasing Wavelength (λ) in nm →

Figure 3.2. The electro-magnetic spectrum with the visible light expanded. Courtesy: Wikimedia Commons

in the order of $1-10\mu m$. These produce an output signal that is proportional to the light intensity at its area. Colour cameras usually cover the photosites with filters, passing either red, green or blue light. CIE also defined the following wavelengths for these so called primary colours, red (R) at 700 nm, green (G) 546.1 nm and blue (B) 435.8 nm. Already in the 17th century it was discovered that white light is a mix of various colours. Generally any colour space is a 3-dimensional space that contains all possible colours and all levels of brightness [Gonzalez and Woods, 2006]. Note though that the fixed primary colours are not able to produce all colours if they can not vary their wavelength. The most commonly used space is the RGB space, yet other spaces exist, e.g. HSV using Hue, Saturation and Value/Lightness.

Human perception seems to be better at dealing with colours in terms of hue and saturation, or also called chromaticity (Figure 3.4 shows the CIE chromaticity diagram). Hue is referring to the dominant colour as perceived by an observer, i.e. when we call an object as 'red' we are referring to its hue. Saturation on the other hand is defining the purity of the colour. The pure spectrum colours are fully saturated, i.e. no white light is mixed in. In HSV, the intensity dimension is named either V for value or L for lightness though these are computed quite differently. Furthermore studies show that our perception of white is adaptive. Allowing us to tune out tune out various effects of scene illumination and lighting. Our perception adapts so that the integral (or mean) over the entire scene is grey.

All of this poses real problems for a robot using colour images for visual per-

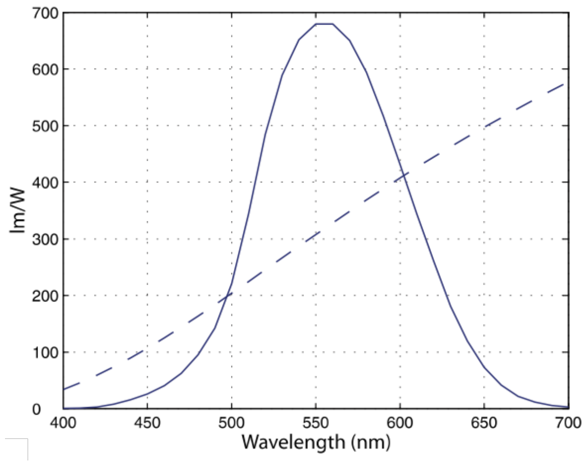


Figure 3.3. Luminosity curve for standard human observer. The difference to a silicon CCD camera can be seen (from [Corke, 2011]).

ception. In addition camera systems are available also in other spectra, such as, infra-red or ultra-violet cameras, as well as, hyper-spectral cameras, that sample the incoming radiation at many points.

Images from Cameras

To understand how the camera image is created one needs to understand the camera's setup. The light passes through the camera before it is collected by the sensor. A very basic camera model, the pinhole camera model (Figure 3.5), is commonly used to explain the geometric relation between the scene and the sensed image.

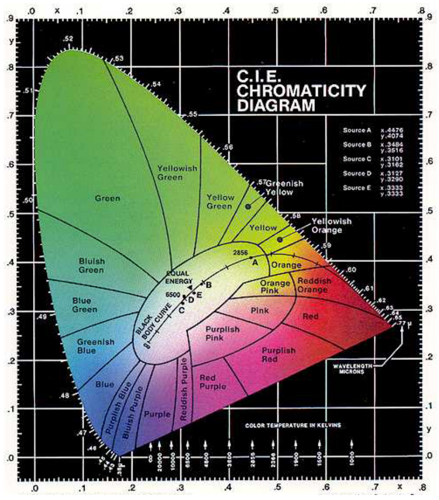


Figure 3.4. The CIE chromaticity diagram. It shows the pure (mono-chromatic, saturated) colours along its periphery. Approaching the centre colours become less saturated, i.e. whitish or greyish. (Courtesy: Sencore.com / CIE)

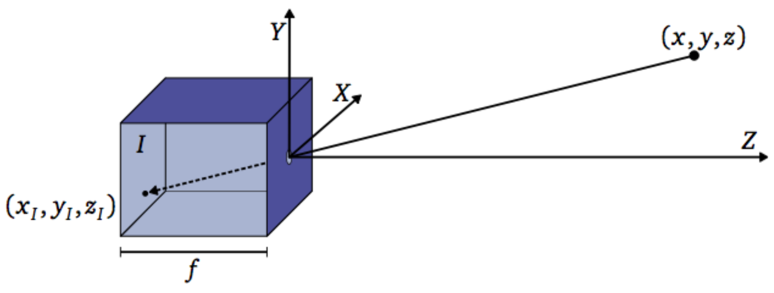


Figure 3.5. Original pinhole camera model

The light from the scene passes through the pinhole and is projected on the rear of the box, which is called the projection plane I. This defines the following geometric relationship:

$$x_I = \frac{x}{z} \cdot f, \tag{3.1}$$

$$y_I = \frac{y}{z} \cdot f, \tag{3.2}$$

$$z_I = f \tag{3.3}$$

In the central-projection model, the projection plane I lies between the pinhole and the scene at a distance f (referred to as focal length) without affecting these geometric relationships. For images the values of interest are the coordinates along the projection plane I. Using similar triangles it can be shown that a point at the world coordinates $P = (X, Y, Z)$ is projected into I as $p = (x, y)$ by

$$x = f\frac{X}{Y}, y = f\frac{Y}{Z}, \tag{3.4}$$

To further ease processing, the origin of the coordinates on the plane I are usually displaced to be at one of the corners. These pixel coordinates are usually referred to as u (along the X axis) and v (along the Y axis). The homogeneous coordinate of the world point P in pixel coordinates is $p = (u', v', w')$. In matrix form,

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

or more commonly written as

$$p = K \cdot P \tag{3.5}$$

The matrix K is representing the *intrinsic parameters* of the camera used, P the position of the point in the world and p the coordinates of the point on the image plane. The non-homogenous pixel coordinates are then $u = u'/w'$ and $v = v'/w'$. This representation allows to specify different focal lengths for the planes X and Y. This is specially useful for CCD cameras, where the pixels are not necessarily squared, resulting in different projection planes.

This model though is generally too simple for the use in robotics. Lenses are used in combination with the cameras to collect more light and to create a different field-of-view (e.g. fish-eye lenses). This though leads to a distortion of the image, which is depending on the type of lens used. The distortion of the *iCub* camera system can be seen in Figure 3.6. If the distortion parameters are known the received image can be 'un-distorted'. Camera calibration methods, such as the one provided by OpenCV (based on Zhang [2000]; Bouguet [2014]) allow to estimate the parameters. OpenCV also provides a function to un-distort images using the estimated values.

Correcting for distortion will yield non-rectangular images. A common approach therefore is to discard some points to keep a rectangular image or fill up the empty pixels (after projection) with black or white pixels.

Images do not have to be represented like this. Another commonly used representation in computer vision consists methods that represent visual information with a space-variant resolution inspired by the visual system of mammals. This so called log-polar imaging has been studied for a a while and has shown interesting results in robots, mainly the ones where real-time constraints make it necessary to utilize resource-economic image representations and processing methodologies [Traver and Bernardino, 2010].

Feature Detection

Once an image is generated by the sensor it needs to be processed. There exists a vast body of work on all aspects of image processing, using both classical and machine learning approaches, Gonzalez and Woods [2006] provides a great overview. For detecting the content of an image, feature detection is an important and commonly used approach. Human designed solutions such as SIFT [Lowe, 1999], SURF [Bay et al., 2006], FAST [Rosten et al., 2010] and ORB [Rublee et al., 2011], use statistical methods to detect and describe interesting features

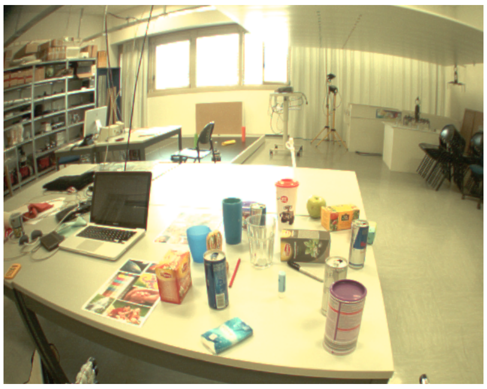


Figure 3.6. The distortion introduced by the system is clearly visible in this *iCub* camera image

(i.e. local properties) within an image. These features are a brief yet comprehensive representation of the image or scene that possess the property of being interesting and repeatable. Technically, the features are patches of the image, yet to get these robust features, first interest points are found in the image. Local image features (also called local descriptors) are the entities that capture the information of the region around these interest points. These descriptors have proven to be resistant to occlusions, local deformations of the objects, variation in illumination conditions, and background clutter [Mikolajczyk and Schmid, 2003; Agarwal and Roth, 2002].

The feature descriptors represent a feature as a vector of fixed size. Various ways of extracting this description exist and each method uses their own methodology. However, they all attempt to be robust against changes in lighting, rotation, scale, and viewpoint. Therefore, the feature descriptor of the three different methods expresses the same information about the feature: details of the pixel of interest, the neighbouring area (patch), its orientation and its scale.

Image Classification

Determining whether image data contains a specific object or scene is an important problem in robotic vision. This has been researched for quite some time but the task seems harder than expected. No solution for the general case, i.e. detecting arbitrary objects in arbitrary situations, exists [Kemp et al., 2007]. Problems arise because of real-life environmental impacts, such as, changing light conditions, incomplete or inconsistent data acquisition [Kragic and Vincze, 2009]. An overview of various approaches can be found in the recent survey by Cipolla et al. [2010].

Image classification usually refers to 'tagging' a picture with a label, such as 'car', 'tree', or 'human face'. Generally these provide not a per-pixel information but a more semantic information about the scene depicted. Most of the state-of-the-art object detection applications are using features as described above, maybe with extensions for higher robustness [Stückler et al., 2013]. For real-time object tracking in video frames contour-based trackers are often used [Panin et al., 2006]. Machine learning has been used in a variety of ways to facilitate image classification. From learning interesting and relevant features [Vafaie and De Jong, 1992; Rosten and Drummond, 2006] to more biologically-inspired deep learning for neural networks, i.e. hierarchical artificial neural networks where each layer performs specific tasks helping to perform the final image classification not unlike what we believe the visual cortex does [Jarrett et al., 2009; Ciresan et al., 2011].

Image Segmentation

While classifying an image provides a label (or multiple) for an image, image segmentation on the other hand provides pixel-wise information. It refers to the process of separating foreground from background, one object class from the background, or one object class from others. More general the image is partitioned into groups of non-intersecting regions. Segmentation is an essential step in low-level vision, as well as robotics. An overview of image segmentation techniques and a theoretical discussion can be found in Pal and Pal [1993]. It is the nowadays one of the most common approaches to perform object detection in vision task related to object manipulation. However, image segmentation is especially challenging when the objects are static, occluded or the background model is not known.

Also for image segmentation task machine learning has been used extensively, in recent years deep-learning has shown to be a viable option also here [Masci et al., 2013].

Object Localization

For applications in object manipulation, perception needs to also tackle the issue of localization. Only once a feature or object is successfully detected in multiple camera images, it can then be localized and placed into a model of the environment. 'Spatial Perception', as this is known, develops in humans over time from observation and interaction with the world. Research in neuro-science shows clear trends on *what* changes during this development, but *how* these changes happen is not understood [Plumert and Spencer, 2007]. There is also evidence showing that the metric precision changes systematically over time (from 2-11 year old kids) [Schutte et al., 2003].

Stereo vision systems use two images taken from different angles to obtain a distance measure [Hartley and Zisserman, 2000]. The localization of objects in the environment is required for use in combination with on-line motion planning for object manipulation tasks on humanoids. In the humanoid robot scenario a method, which is able to cope with motion in the robot's head, gaze and upper body, is required.

While projective geometry approaches work well under carefully controlled experimental circumstances, they are not easily transferred to robotics applications though (as seen by the out-of-the-box localization capabilities of the *iCub* [Pattacini, 2011]). In projective geometry the parameters that define a camera position with respect to a scene are called the camera's *extrinsic parameters*. Using the extrinsic and intrinsic parameters (Eq. 3.5) it is possible to fully locate the object in the 3D world. The extrinsic parameters define the pose of the camera, translating and rotating the previously defined coordinate system that originates in the pinhole of the camera. It is now possible to map a point in the world to a pixel in an image. Using the camera position (t), orientation (R) and intrinsic parameters (K), the mapping is defined as:

$$p_I = K[R|t]p \tag{3.6}$$

When localizing an object the approach is the inverse of what is described above, i.e. one wants to determine the objects position in the world based on where the pixel is in the image. For this multiple camera images taken from different viewpoints are necessary as a single image will not be able to yield a single result. Figure 3.7 depicts how any point along the ray from the origin will lead to the same projection into I.

In computer vision this research area is usually referred to as 'Stereo Vision' or '3D reconstruction'. Stereo vision focusses more on the aspect of using two cameras in parallel. To obtain a distance measure the relative displacement of a pixel between the two images is used. Cameras that photograph the same scene from two different locations provide different 2D projections of the 3D scene. If the 'intrinsic parameters' that specify each camera's projection from 3D to 2D, as well as the 'fundamental matrix' that is the rigid-body transformation between the two cameras' coordinate systems, are known, and if there are some features of the scene that can be identified in both images, then the 3D locations of those features can be triangulated. Hartley and Zisserman [2000] provides a thorough review of approaches based on this principle.

More general in 3D reconstruction multiple images, also from the same camera, are used to build a model of the scene. Epipolar geometry is dealing with the geometry of stereo vision. The epipolar plane defined by the origins of each perspective (O and O') and the point in the real world. Figure 3.7 shows how the two projection points, p_I and $p_{I'}$, lie in the plane. All the points which could result in p_I are along the projection line or ray. The projection of these points into the other camera form a line in its projection plane I', this is the epipolar line $e_{I'}$.

When working with images, only p_I and $p_{I'}$ are known, but converting a point position from the Cartesian to the unnormalized homogeneous coordinate system is simple. Using the intrinsic parameters of the camera:

$$\begin{aligned} p_{I'} &= K p_{O'} \\ K^{-1} p_{I'} &= p_{O'} \end{aligned} \tag{3.7}$$

And a transposed point:

$$\begin{aligned} p_I^\top &= (Kp_O)^\top \\ p_I^\top &= p_O^\top K^\top \\ p_I^\top K^{-\top} &= p_O^\top \end{aligned} \tag{3.8}$$

Hence, the constrain that relates two different perspectives given a translation and rotation (Essential matrix) can now be expressed using only image coordinates:

$$\begin{aligned} p_O^\top E p_{O'} &= 0 \\ p_I^\top K^{-\top} E K^{-1} p_{I'} &= 0 \\ p_I^\top F p_{I'} &= 0 \end{aligned} \tag{3.9}$$

Equation 3.9 defines the fundamental matrix, as $F = K^{-\top} E K^{-1}$. There are different ways to find the Fundamental matrix, but they all depend on the intrinsic parameters of the camera and a minimum number of matching points visible from each perspective.

Related fields of research are visual SLAM (or VSLAM) and 'Structure from Motion' (SfM). There has been intense research into VSLAM (visual SLAM) in the last decade (e.g. [Karlsson et al., 2005; Davison et al., 2007]). VSLAM uses primarily visual sensors, because of the increasing availability and low cost of cameras. SLAM can be thought of as a 'chicken or egg' problem: An unbiased map is needed for localization while an accurate pose estimate is needed to build that map. This is the starting condition for iterative mathematical solution strategies.

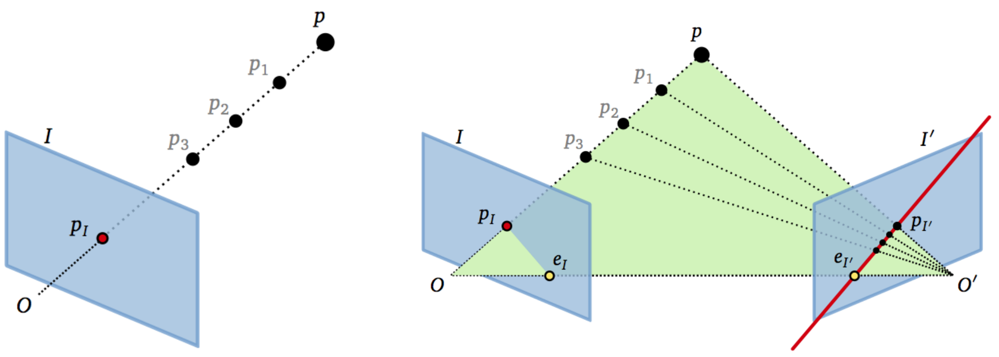


Figure 3.7. How different 3D points produce the same projection point p_I in the image is shown on the left. The right shows how a second image provides the additional information to determine the correct position in the world.

Structure from motion (SfM) is a range imaging technique and commonly refers to the process of estimating 3D structures from image sequences. In biology it is used to describe the phenomenon by which humans can recover 3D information and structure from the 2D (retinal) projection. Finding structure from motion presents a similar problem as finding structure from stereo vision. In both instances, the correspondence between images and the reconstruction of 3D object needs to be found.

While traditional stereo vision approaches, based on projective geometry, have been proven effective under carefully controlled experimental circumstances, they are not ideally suited to most robotics applications. Intrinsic camera parameters and the fundamental matrix may be unknown or time varying, and this requires the frequent repetition of lengthly calibration procedures, wherein known, structured objects are viewed by the stereo vision system, and the required parameters are estimated by numerical algorithms. The *iCub*'s visual sensory system is a pair of cameras mounted in the head in a human-like fashion (see Figure 3.8), providing passive, binocular images.In the following discussion, *CSL* and *CSR* refer to the local reference frames of the left and right cameras respectively, the reference frame of the body is *CSBody*, but as it is mounted at a fixed point this is also the reference frame chosen for the environment. Therefore

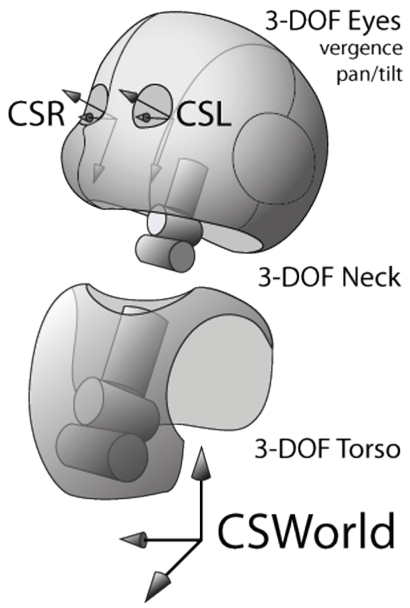


Figure 3.8. The relevant coordinate frames for object localization on the *iCub*. Cameras located at the origin of *CSL/CSR* are used to express the position of objects with respect to the *CSWorld*.

CSWorld denotes the common environmental reference frame, in which we seek to express object locations. Assuming a solution to the standard stereo vision problem, applying it to a real physical robot to facilitate object manipulation remains a challenge. In many robotics applications, it is somewhat inconvenient to express the environment with respect to a camera. For example, from a planning and control standpoint, the most logical choice of coordinate system is *CSWorld*, the reference frame at the base of the manipulator, which does not move with respect to the environment. In order to transform coordinates from *CSL* or *CSR* to *CSWorld*, such that we can model objects and control the robot in the same frame of reference, an accurate kinematic model of the robot is required. If such a model is available, it must be carefully calibrated against the actual hardware, and even then its accuracy may be limited by un-modelled nonlinearities.

Several different localization systems have previously been developed for the *iCub*. A popular representation for (stereo) vision research is based on log-polar transformed images. This biologically inspired approach first applies a transformation to the camera images before typical stereo vision algorithms are used. The available module currently supports only a static head, i.e. it puts the object position in the *CSL/R* coordinate frame. The 'Cartesian controller module' provides another basic 3D position estimation functionality [Pattacini, 2011]. This module works well on the simulated robot, however its performance on the hardware platform is weak, this is because of inaccuracies in the robot model and camera parameters. The most accurate, currently available localization module for the *iCub* exists in the 'stereoVision' module providing centimetre accuracy. Unlike the presented log-polar approach, this module employs the entire *iCub* kinematic model, providing a position estimate in the *CSWorld* coordinate frame. The module requires the previously mentioned 'Cartesian controller' and uses tracking of features to improve the kinematic model of the camera pair by estimating a new fundamental matrix continuously. The precision of all of these approaches depends upon an accurate kinematic model of the *iCub*. A very accurate model, or estimation of the model, is therefore necessary. While the spatial perception problem could, in theory, be solved numerically, in practice though, no precise enough kinematic model is available for the *iCub*. For other robots, such as Holland's fully anthropomorphic CRONOS [Holland and Knight, 2006], no useful kinematic model exists. Therefore the focus is on solving the spatial perception problem on the real hardware. In fact a precise learning of spatial perception could in turn be used to build an accurate kinematic model.

There exists currently no module estimating the kinematics of the *iCub*, for other robotic systems this has been done: visual feedback to learn the model of a holonomic wheeled robot [Gloye et al., 2005] and learn the model of a legged robot using low-dimensional sensory feedback [Bongard et al., 2006]. Yet no approach has been shown with high-dimensional sensory information (such as images).

In robot learning, especially imitation learning, various approaches have been investigated to tackle these problems. Sauser and Billard [2005] have investigated the problem of reference frame transformations from a neuroscience perspective. They were able to imitate gestures from a teacher on a Hoap-2 humanoid robot with external fixed cameras. Though promising their approach has so far not been extended to systems with non-stationary cameras.

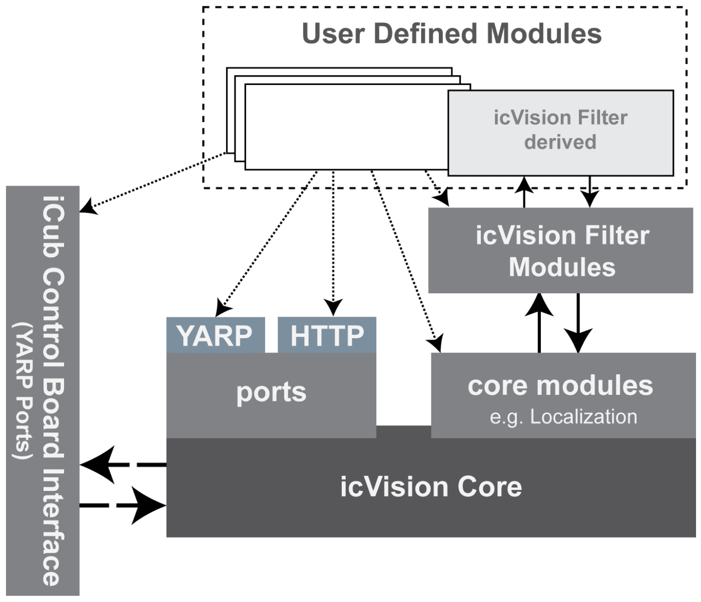


Figure 3.9. Overview of the *icVision* architecture. It consists of loosely-coupled modules to provide a simple framework for developing computer vision solutions and enabling integration of the research directly on the *iCub* hardware.

3.2 *icVision*: Framework for Robot Vision and Cognitive Perception

At IDSIA over the last years I developed the *icVision* framework [Leitner et al., 2012c, 2013b]. It aims to be a tool (or more a suite of tools) for an easier development, testing and integration of the on-going computer vision research into the real hardware. One of the main design goals is to allow rapid prototyping and testing of vision software on the *iCub* and reduce development time by removing redundant and repetitive processes. *icVision* is implemented in C++, open-source and uses YARP as middleware and OpenCV as framework to perform the underlying image processing.

The design principle behind *icVision* is to provide standardised interfaces to the most relevant parts of the robot with regards to computer vision. It focusses on a modular design that allows to quickly test different implementations for detection, tracking and classification.

One example of a work-flow through some *icVision* modules shows the 3D localisation of objects in real-time on the hardware (see Figure 3.10). The various filter modules provide means to detect objects in the images. This can be learned as shown later in this chapter. On the other hand they can also be pre-programmed detectors, such as haar-like face detectors, or even trained neural networks. The only constraint is that they provide the expected interfaces. By providing wrappers and class stubs, the development time of such modules is greatly reduced. The 3D location wrt. the robot's reference frame is performed in a module that as an input just needs pixel coordinates in both images and a time stamp. The module has access to the robot's joint encoders and a model, either known or leaned, and with those is able to estimate the position of the object. It interfaces with the MoBeE environment, through a YARP port to place a geometry into the robot's world model enabling the robot to, for example, avoid colliding with them during operation.

The framework allows for the quick swapping of these modules by being able to connect/disconnect these YARP ports on the fly. It also is possible to deal with hyperspectral input images, and theoretically also with multiple sources of sensing (multi camera setups, RGB-D cameras, etc.). A simple GUI was developed to quickly turn filters on and off on the hardware. While currently the pipeline is using mainly single feature detecotrs, in the future this should be extended to connect and stack various filters together. A decent overview of some of the experiments done using the framework can be found in Leitner et al. [2013a].

Architecture

icVision is a distributed framework for running robot vision modules (see Figure 3.9). The 3D localization is done the following: At first the camera images are acquired from the hardware via YARP Metta et al. [2006]. The goal of YARP is to minimise the effort devoted to infrastructure-level software development, especially for humanoid robots and within the *iCub* research and projects. It facilitates code reuse, modularity and hopes to maximize code exchange and collaboration. The main features of YARP include support for inter-process communication (IPC), basic image acquisition and processing, as well as a class hierarchy to ease code reuse across different hardware platforms. It is currently used and tested on Windows, Linux, Mac OS and QNX6. Communication between modules in YARP is done using the Observer pattern. The information is delivered to any number of observers to a "Port" . These can be any number of processes distributed across any number of machines over a network. These ports area "an active object managing multiple connections for a given unit of data either as input or output" [Metta et al., 2006]. For *icVision* image ports are created for each module. Apart from the main modules these are the filters for a specific object detection and identification. Communication is controlled by YARP and can be either via TCP or UDP (or IPC when running on the same machine).

In the framework the colour images received from the hardware are converted into greyscale, as well as split into RGB and HSV channels. They are then distributed via the previously mentioned ports to all (active) *icVision* filters. Each filter then processes the images received by applying OpenCV functions with the aim of segmenting the object (see Section 3.3.1 for more details on how this can be learned). The output of this is a binary image, segmenting the object to be localized. A blob detection algorithm is run on these binary images to find the (centre) location of the detected object in the image frame. The position of the object in both the right and left camera images is sent to the 3D localization module, where together with the robot's pose, a 3D location estimation is generated. The last step places the object in the existing world model (see Figure 3.10).

To further facilitate the reuse of the code the `icImage` class is provided. It is a wrapper for the OpenCV functionality (over 60 functions are used by the Cartesian Genetic Programming for Image Processing – CGP-IP – framework, see next section) and memory management within the *icVision* framework, it also enables a quick and pain-less swapping between C#and C++ code.

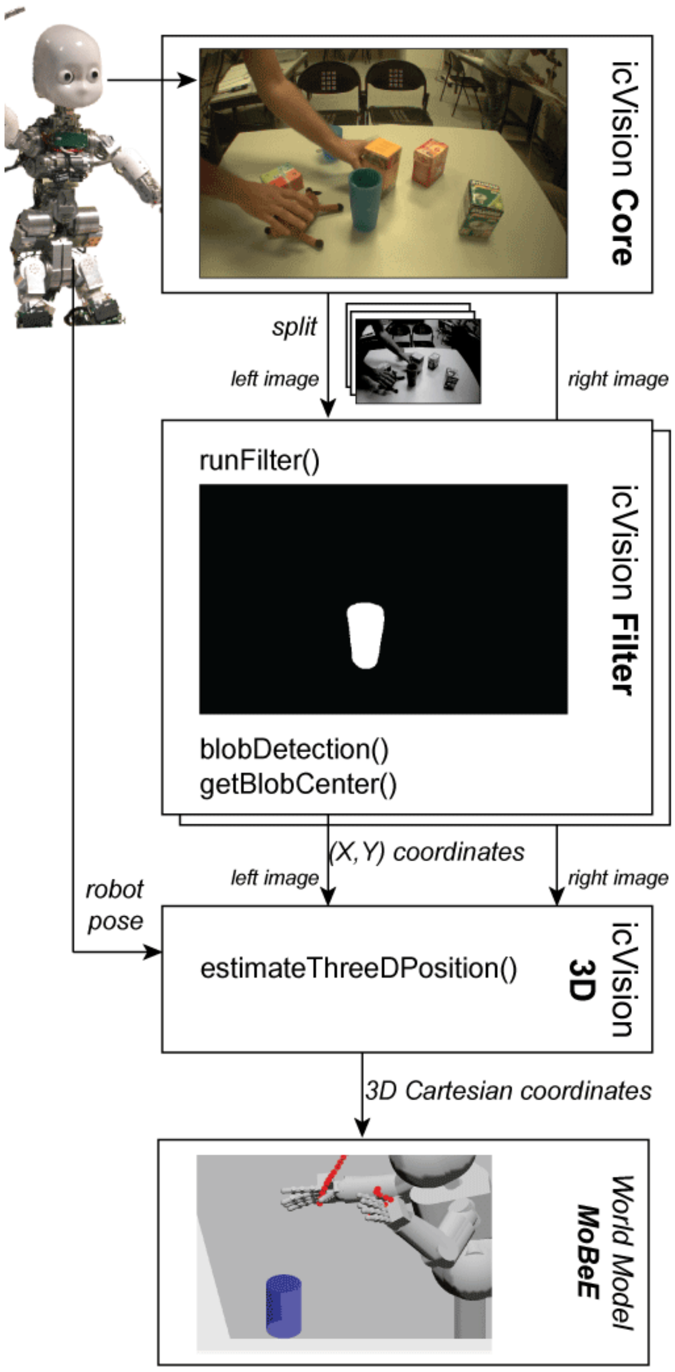


Figure 3.10. The workflow through the *icVision* framework modules for localising an object in operational space.

3.3 Visual Perception and Object Detection

Object detection is not yet solved in a general sense, as mentioned earlier (Section 2.2), yet it is a very important issue to enable autonomous grasping. A hard problem on the *iCub* in particular although there has been some recent progress, especially from the lab at IIT [Ciliberto et al., 2011a; Fanello et al., 2013; Gori et al., 2013]. For autonomous object manipulation with humanoid robots it is of importance to detect and identify the objects to interact within the environment. This is still a challenging problem in robotics, especially in settings where lighting varies, viewing angles change and the environment can be described as 'cluttered' (i.e. lots of different objects in the scene, partly obstructing each other). Given the maturity of the field of image processing, it should be possible to construct programs that use much more complicated image operations and hence incorporate domain knowledge.

A technique based on Cartesian Genetic Programming (CGP) [Miller, 1999, 2011] was developed allowing for the automatic generation of computer programs for robot vision tasks. A large subset of the functionality of the freely available OpenCV image processing library [Bradski, 2000] are used as building-blocks of these programs. The implementation provides an effective method to learn (unique) object detection modules. If the training set is chosen correctly, even different lighting conditions are no longer problematic.

Background: Cartesian Genetic Programming (CGP)

A genetic algorithm (GA) [Holland, 1975] is a search technique inspired by Darwinian evolution. It is one of the three primary research areas under the common umbrella of evolutionary computation, with the other two being evolutionary programming [Fogel et al., 1966] and evolution strategies [Rechenberg, 1965; Schwefel, 1965]. Genetic Programming is a GA applied to solve specific problems by generating formulae or programs; most commonly to find the right parameters to minimize (or maximize) an arbitrary function $F(X_1, X_2, \dots, X_n)$, with $n \in \mathbb{N}^+$. These parameters can be of arbitrary nature and do not need to be known beforehand, as GP is a black-box optimiser. More precisely GP is a "*probabilistic search algorithm that iteratively transforms a set (called a population) of mathematical objects (typically fixed-length binary character strings), each with an associated fitness value, into a new population of offspring objects using the Darwinian principle of natural selection and using operations that are patterned after naturally occurring genetic operations, such as crossover (sexual recombination) and mutation*" [Koza, 1992]. Each GP consists of a population $P = \{I_1, \dots, I_m\}$ of $m \in \mathbb{N}^+$ individuals and a fitness function $f(I)$ which specifies how well each individual $I \in P$ performs (solves the problem, i.e. minimizes/maximizes $F(.)$). Generally the fitness function $f(.)$ is assumed to return a single value in $\mathbb{R}$. Each individual I is furthermore made up of a list of genes, with length $l \in \mathbb{N}^+$, $I = \{g_{I1}, ..., g_{Il}\}$, which represent (or can be mapped to) the the unknown parameters $X_1, \ldots, X_n$. This representation is usually referred to as the genotype of the individual. Herein the genes are defined to be either of $\mathbb{N}$ or $\mathbb{R}$. The GP algorithm creates a new population P' from the current one by changing the individuals through selection, recombination and mutation of the genes, $P \xrightarrow{GP} P'$. The GP terminates once a sufficient number of individuals is tested or the best individual $I_b = \text{argmin}_{I \in P'} f(I)$ is found to be sufficiently fit. GP has been applied successfully in many areas, including medicine and bioinformatics [Handley, 1993] and even as a hyper-heuristic [Oltean, 2005]. Furthermore GP has often been used to solve problems in image processing [Spina et al., 2009], however previous attempts typically use a small set of mathematical functions to evolve kernels, or a small number of basic image processing functions (such as erode and dilate). Given the maturity of the field of image processing, it should be possible to construct programs that use much more complicated image operations and hence incorporate domain knowledge.

In this section, a technique based on Cartesian Genetic Programming (CGP) is presented. It allows for the automatic generation of computer programs using a large subset of the OpenCV image processing library functionality [Bradski, 2000]. The design choices for the approach are intended to match human design choices – hence the use of an industry standard API, the generation of code in standard programming languages and the inclusion of additional domain knowledge. The efficacy of this approach is demonstrated in Section 3.4. The results from several different domains – basic image processing, medical imaging, terrain classification, object detection in robotics and defect detection in industrial application – are presented.

Cartesian Genetic Programming (CGP) is a form of Genetic Programming (GP), an evolutionary algorithm (EA), in which programs are encoded in partially connected feed forward graphs. CGP grew from a method of evolving digital circuits developed by Miller et al. [1998]. However the term "Cartesian genetic programming" first appeared in [Miller, 1999] and was proposed as a general form of genetic programming in [Miller and Thomson, 2000]. It is called 'Cartesian' because in its very first form it represented a program using a two-dimensional grid of nodes [Miller, 2011], different from previous "tree-based" GP approaches [Cramer, 1985].

The genotype, given by a list of nodes, encodes the graph. For each node in the genome there is a vertex, represented by a function, and a description of the nodes from where the incoming edges are attached. This representation has a number of interesting properties. For instance, not all of the nodes of a solution representation (the genotype) need to be connected to the output node of the program (the phenotype). As a result there are nodes in the representation that have no effect on the output, a feature known in GP as 'neutrality'. This has been shown to be very useful in the evolutionary process [Miller and Smith, 2006]. Also, because the genotype encodes a graph, there can be reuse of nodes, which makes the representation distinct from a classically tree-based GP representation. The general form is shown in Figure 3.11, and an example from the proposed framework CGP for Image Processing (CGP-IP) is visible in Figure 3.13. Dickmanns et al. [1987] presented an alternative approach, also allowing reuse and even loops. CGP has previously been used for image processing tasks. Several examples can be found in [Sekanina et al., 2011].

A basic genetic algorithm works as follows (see also Figure 3.12 for reference): Initially, a population of candidate solutions P is generated randomly. This population consists of a (in most cases) fixed number of individuals (m), who in turn are defined by their genome, as described above. These are at first

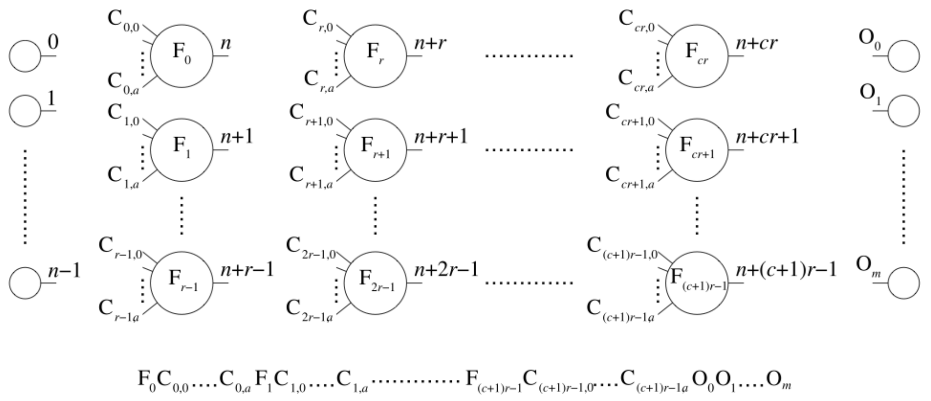


Figure 3.11. The general form of CGP. It is a grid of nodes whose functions are chosen from a set of primitive functions. In what follows, n_c, n_r, n_i, n_o and a are natural numbers. The grid has n_c columns and n_r rows. The number of program inputs is n_i and the number of program outputs is n_o. Each node is assumed to take as many inputs as the maximum function arity a. Every data input and node output is labeled consecutively, which gives it a unique data address specifying where the input data or node output value can be accessed [Miller, 2011].

randomly generated, i.e. each gene g_i, with $i \in 1, m$ is initialized with a random value (see Figure 3.12a, where each gene is represented by an either white or black vertical stripe).

Following the creation of the population each of these individuals is tested to see how well it performs the given task, i.e. in our case segment the object of interest from the rest of the scene in a set of test images. This is generally known as evaluating the 'fitness function' $f(I)$. It is used to assign a numeric score to each individual in the population, $f : I \to \mathbb{R}$. Generally, the lower this error, the better the individual is at performing the task. The individuals are ranked according to this metric (Figure 3.12b).

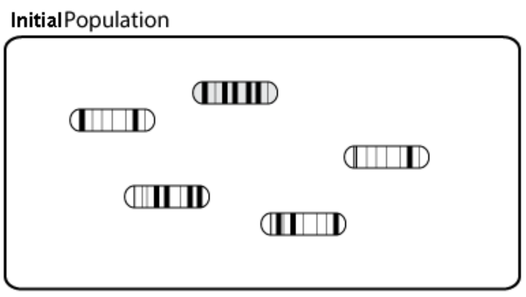
The next step generates a new population P' from the old population P. This is done by taking pairs of the best scored genotypes and performing functions analogous to recombination and mutation. In the case presented herein, as is common in CGP, only one (the fittest) individual is used to generate offsprings only by mutation (Figure 3.12c). Herein a 1+4 evolutionary strategy is used, i.e. $|P| = 5$. Four offsprings are created from the best individual and are replacing the lowest performing individuals in the new population P' (Figure 3.12d).

This process is then repeated, i.e. fitness scores are calculated for these new individuals, using the fitness function. The process of test and generate is repeated until a stop condition is satisfied, i.e. a very close to perfect solution is found or until a certain number of individuals have been evaluated.

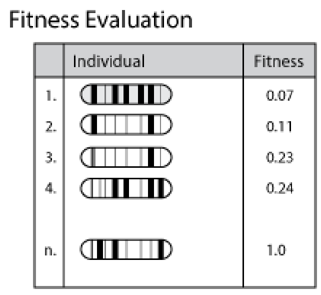
Many GP approaches work as convolutional filters, in the sense that the GP generates a kernel or convolution matrix. In this type of approach, a sliding window (using the kernel size) moves across an image. At each pixel location, an evolved expression takes the in neighbouring pixels' values, and computes a new value for the centre pixel [Gonzalez and Woods, 2006]. Typically these programs operate at a mathematical level, where operations such as $+, -, \times$ and $\div$ are used [Harding, 2008]. Implementations on Field Programmable Gate Arrays (FPGAs) for noise reduction, use a mixture of binary operations (e.g. OR, XOR, AND, etc.) and mathematical functions [Slaný and Sekanina, 2007]. Similarly, Wang and Tan [2011] also uses a mixture of function types, in this instance binary and some basic morphological operations, such as dilate. The first demonstration of CGP-IP in Section 3.4 revisits the noise reduction problems.

GP has also been applied to the generation of 'features' to be used in feature detection. For example in finding corners [Seo and Kim, 2010], or object specific features [Szymanski et al., 2002; Tackett, 1993; Yu and Bhanu, 2006], or 'interesting' features that are comparable to SIFT [Trujillo and Olague, 2008, 2009; Watchareeruetai et al., 2008].

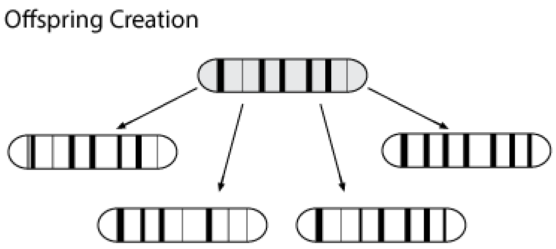
GP has been applied to the problem of classifying images in many different domains. For example, adding loops to GP enabled Wijesinghe and Ciesielski [2007] to classify simple shapes using just primitive mathematical operators. Using a simple function set, but operating on a large number precomputed statistics,



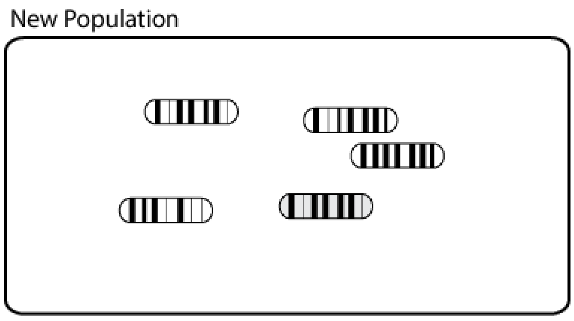
(a) The first population with its randomly generated individuals. The gene values are represented by the vertical line thickness.



(b) The ranked individuals with their respective fitness values.



(c) The offsprings are created from the best individual by mutating each gene with a given probability.



(d) The offsprings and the best individual from the previous round created the new population.

Figure 3.12. Illustration of a typical iteration for an evolutionary search using (Cartesian) Genetic Programming.

Zhang et al. [2003] were able to perform domain-independent and multiple class object recognition. This paper also has a good overview of previous work in object detection in GP. Terrain classification using GP has been applied to satellite imagery [Silva et al., 2010], including work on hyper-spectral images [Uto et al., 2009]. In Section 3.4, experiments with CGP-IP classifying terrain in a space robotics application are included.

As mentioned above, segmentation is the process of separating foreground from background, or one object class from the background. Using GP, Spina et al. [2009] evolved programs to perform segmentation based on features calculated from partially labelled foreground and background. In an approach similar to CGP, Shirakawa and Nagao [2007] evolved segmentation programs that use many high-level operations such as mean, maximum, minimum, Sobel, Laplacian, sum and product. This approach is probably the most similar to CGP-IP. In later work, the technique was successfully applied to texture classification [Shirakawa et al., 2009]. In the domain of medical imaging, Poli [1996] used GP with basic mathematical operators, working at a per pixel level to segment features from MRI scans. CGP-IP is demonstrated on two medical imaging problems for classification (Section B.1-B.2).

3.3.1 Cartesian Genetic Programming for Image Processing

Cartesian Genetic Programming for Image Processing (CGP-IP) draws inspiration from much of the previous work in the field. As will be demonstrated, it uses a mixture of primitive mathematical and high level operations. It uses CGP, which appears to be a popular choice for the representation in this domain. It encompasses domain knowledge, and could even be easily adapted to include the findings from previous work in its function set. CGP-IP implementation can handle colour images with multiple channels, which is a key difference to much of the previous work, which focused on the use of grey scale images. Treatment for colour images is to separate them into RGB (red, green and blue) and HSV (hue, saturation and value) channels, and provide these as available inputs. Each available channel is presented as an input to CGP-IP, and evolution selects which inputs will be used, as all functions operate on single channel images. Our implementation generates human readable C#or C++ code based on OpenCV to be run directly on the real hardware – using our *icVision* framework. An illustrative example of a CGP-IP genotype is shown in Figure 3.13, Figure 3.21 shows the actual operations in one of the evolved filters. The first three nodes obtain the components from the current test image (e.g. grey scale version, red and green channels). The fourth node adds the green and red images together. This is then dilated by the fifth node. The sixth node is not referenced by any node connected to the output (i.e. it is neutral), and is therefore ignored. The final node takes the average of the fifth node and the grey scale input.

Node Representation Compared to classical CGP, CGP-IP needs additional values encoded in each node. This is because the available functions often require one or more parameters, and these parameters have requirements as to their type and range. Hence, each node in a CGP-IP graph contains the following elements:

- Function: Integer representing a function from the set of available functions.
- Connection 0: Integer representing how many nodes back in the current graph this node should connect to obtain the first input to the function.
- Connection 1: Integer representing how many nodes back in the current graph this node should connect to obtain the second input to the function.
- Parameter 0: Real number, typically used as a constant value.
- Parameter 1: Integer in the range -16 to +16, used as a parameter to an image operation.
- Parameter 2: Integer in the range -16 to +16, used as a parameter to an image operation.
- Gabor Filter Frequency: Integer in the range 0 to 16, used as a parameter for Gabor filter operations.
- Gabor Filter Orientation: Integer in the range -8 to 8, used as a parameter for Gabor filter operations.

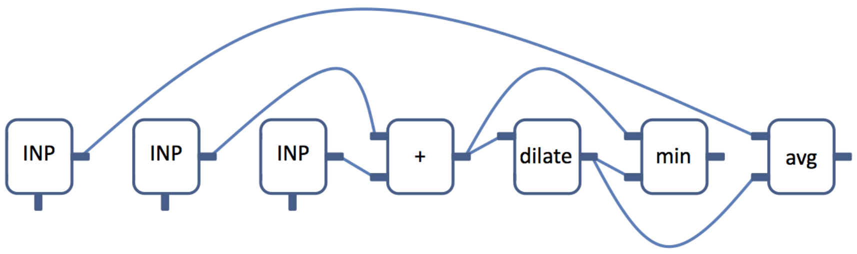


Figure 3.13. Example illustration of a CGP-IP genotype.

If a relative address extends beyond the extent of the genome it is connected to one of the inputs. Which input image that is, is determined by specialised 'Image Input' functions provided (INP, INPP, SKIP). These manipulate a pointer that indexes the available inputs and return the currently indexed input. Harding et al. [2010a,b] provide a full description. This makes sure that the genomes do not run into problems with non-valid input images or connections to unavailable nodes. It ensures that every node returns an image output every time it is called. The overall output of the evolved filter is taken from the last node in the genome (compare Figure 3.13).

All genes in the genotype have an equal probability of being mutated. The type of mutation that occurs depends on the type of gene being mutated. For the function genes, a new function is selected at random from the available functions. The connection genes are mutated to a new value from 1 to the length of the genotype. For the Gabor Frequency, a random integer between 0 and 16 is used. For the Orientation, mutation selects an integer between -8 and 8. The mutation for the Parameter0 gene is slightly more complicated. If that gene is selected for mutation, with equal probability it will be set to either a random value between -255 and +255, or noise (maximum of +/-10%) will be added. Some commands from the function set require certain parameters to be either even or odd. These issues are handled in the `icImage` class, which converts the parameter if needed. This again makes sure that each node will return an image output every time it is called.

In addition to the program graph, the genotype also includes a real number value that is used for thresholding during binary classification problems. The threshold has a range of 0 to 255. During mutation, this parameter is modified with a 1% probability, and uniform noise of +/-10% is added.

The genotype for CGP-IP contains a large number of elements that need to be evolved. Such a complex representation would normally be considered too large to efficiently evolve, however the results shown in Section 3.4 indicated otherwise. Although further investigation is required, it is suggested that this complexity increases neutrality (which is understood to be useful to CGP) and also increases the flexibility of the function set (leading to an increase in the number of possible solutions). A large number of more specialised parameters in each node avoids the necessity to perform casting and interpretation of an arbitrary set of values in each node, which aids readability. Further, the specialization of the parameters enables mutation to act appropriately.

Executing the genotype is a straightforward process. First, the active nodes are identified to perform the genotype-phenotype mapping. This is done recursively, starting at the output node, and following the connections used to provide the inputs for that node. In CGP-IP the final node in the genotype is used as the output. Details of the decoding stage are shown in Algorithm 1. It needs the current individuals genotype G and returns the number of nodes used (n_u) and to be processed (it also stores their addresses in a nodes-to-process set NP). The pseudo-code requires a few variables $\in \mathbb{N}^+$ unless otherwise specified, such as L_g, which is the length of each individual's genome, M, the number of nodes available, n_n, which is the number of genes representing a single node, n_o, which is the number of output nodes, n_i, defining the number of input nodes, NG is a temporary storage for the genes of a node. The first genes of each node define the connections to other nodes, with the function itself being defined by the last gene of each node, i.e. it is stored in $NG[n_n - 1]$.

Next, the phenotype can be executed on an image. The input image (or images) are used as inputs to the program, and then a forward parse of the phenotype is performed. If the problem is a binary classification, the output image is then thresholded. The complete methodology can be found in Algorithm 2. It requires again the genotype G but in addition also requires the input image to be processed (INP). CGP is also efficient in the sense that only active nodes are computed, and that the results from these nodes are stored and reused.

Fitness Functions

Depending on the application, two different fitness functions are available in CGP-IP. For all fitness measures, a lower value indicates a better solution (with 0 being a perfect score).

Type A: Basic Image Filtering and Processing For many problems, the simplest fitness function is to compute the difference between the image output by an evolved program and a target image on a per pixel basis. In CGP-IP, if this method is used the fitness of an individual is the average error for all pixels in all images in the training set.

Type B: Binary classification For binary classification problems, the output image from the GP is thresholded and treated as a binary image. The threshold value is evolved as part of the individual's genotype. This is then compared to the target image on a per-pixel basis using the Matthews Correlation Coefficient (MCC) [Matthews, 1975], which has previously been observed to be useful for classification problems solved using CGP [Harding et al., 2012].[2] First the 'confu-

[2]The MCC has been compared against other common measures in GP classification by Harding, however this work was performed under an industrial non-disclosure agreement and there-

Algorithm 1 Determining which nodes need to be processed

```
1: $NodesToProcess(G, NP)$ {return the number of nodes to process}
2: **for all** $i$ such that $0 \le i < M$ **do** {Initialize}
3:   $NU[i] \leftarrow FALSE$
4: **end for**
5: **for all** $i$ such that $L_g - n_o \le i < L_g$ **do** {Set all output nodes to true}
6:   $NU[G[i]] \leftarrow TRUE$
7: **end for**
8: **for all** $i$ such that $M - 1 \ge i \ge n_i$ **do** {Find active nodes}
9:   **if** $NU[i] = TRUE$ **then**
10:     $index \leftarrow n_n(i - n_i)$
11:     **for all** $j$ such that $0 \le j < n_n$ **do** {store node genes in NG}
12:       $NG[j] \leftarrow G[index + j]$
13:     **end for**
14:     **for all** $j$ such that $0 \le j < Arity(NG[n_n - 1])$ **do**
15:       $NU[NG[j]] \leftarrow TRUE$
16:     **end for**
17:   **end if**
18: **end for**
19: $n_u = 0$
20: **for all** $j$ such that $n_i \le j < M$ **do** {store active node addresses in NP}
21:   **if** $NU[j] = TRUE$ **then**
22:     $NP[n_u] \leftarrow j$
23:     $n_u \leftarrow n_u + 1$
24:   **end if**
25: **end for**
26: **return** $n_u$
```

sion matrix' is found, which is the count of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). In the case of object detection in images, these are measured on a per pixel basis. Once these values are found the MMC is calculated as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

An MCC of 0 indicates that the classifier is working no better than chance. A score of 1 is achieved by a perfect classifier, −1 indicates that the classifier is

fore details cannot be published.

Algorithm 2 Decoding CGP to get the output

```
1: $DecodeCGP(G, INP, n_u, NP, O)$
2: for all $i$ such that $0 \le i < n_i$ do {Initialize}
3:   $o[i] \leftarrow INP[i]$
4: end for
5: for all $j$ such that $0 \le j < n_u$ do
6:   $n \leftarrow NP[j] - n_i$
7:   $index \leftarrow n_n n$
8:   for all $i$ such that $0 \le i < n_n - 1$ do {store data needed by a node}
9:     $in[i] \leftarrow o[G[index + i]]$
10:  end for
11:  $f = G[index + n_n - 1]$ {get function gene of node}
12:  $o[n + n_i] = NF(in, f)$ {calculate output of node}
13: end for
14: for all $j$ such that $0 \le j < n_o$ do
15:   $O[j] \leftarrow o[G[L_g - n_o + j]]$
16: end for
```

perfect, but has inverted the output classes. Finally, the fitness of an individual is given by:

$$fitness = 1 - |MCC|,$$

with values closer to 0 being more fit.

The MCC is insensitive to differences in class size, making it a nice choice for many binary classification tasks. The confusion matrix also allows for easy calculation of the classification rate as a percentage of correctly classified pixels. The MCC is also a good choice as it seems very suitable to evolve detectors that can work with noise and imprecise maps. This is especially beneficial with data coming from human classification, see e.g. the big blocks used as ground truth and as input masks during various experiments.

A similar, commonly used metric for image segmentation is the F-measure (the harmonic mean of the precision and recall scores).

Domain Knowledge

One of the main benefits of Genetic Programming is the ability to easily include domain specific knowledge. This can be performed in two ways. The first method, which is also applicable to other machine learning techniques, is to provide useful pre-processed inputs. The other method is to provide GP with domain specific knowledge through an appropriate function set. In this approach, both methods are used.

High Level Operations Previous work on imaging processing with GP can be divided into two groups. The first group operates on a convolutional approach. Here, a program is evolved that operates as a kernel. For each pixel in an image, the kernel operates on a neighbourhood and outputs a new pixel value. This is also the typical approach when other machine learning approaches are applied to imaging. In GP, the kernel is typically an expression composed from primitive mathematical operators such as $+, -, \times$ and $\div$. For example this approach was used to evolve noise reduction filters [Harding and Banzhaf, 2008, 2009; Martínek and Sekanina, 2005]. In Harding [2008], many different image processing operations (e.g. dilate, erode, edge detection) were reverse-engineered.

The second group operates on a functional level, where image operations such as dilate and erode are applied to an entire image. This is an obvious method to insert domain knowledge into evolved programs, as now operations that are known to be useful can be used without having to re-evolve the same functionality. CGP-IP combines both these methods. The function set not only contains high-level image processing functions, but also primitive mathematical operations. A complete list of the functions available in CGP-IP can be found in Appendix A.

The primitive operators also work on entire images, i.e. addition will produce a new image adding the values of corresponding pixels from two input images. However, this method does not directly allow for kernel-style functionality to be found. Instead, GP has to use a combination of shifts and rotations and other existing kernel methods to get information about a pixel's neighbourhood. This is similar to the methods proposed to allow for efficient parallel processing of images on Graphics Processing Units (GPUs) [Harding, 2008].

Working at the functional level allows for the easy inclusion of many standard image processing operations. In CGP-IP a large number of well known primitive operations from the OpenCV library are available to the evolutionary process. Additionally, higher level functions such as Gabor filtering are available.

Using OpenCV we can also be confident about using high quality, high speed software. In CGP-IP, individuals are evaluated at the rate of 100s per second on a single core. This makes it both efficient to evolve with, but also means that the evolved filters will run quickly if deployed.

CGP-IP can also use functions such as SIFT, SURF and Hough transforms. These are well known to be useful in object recognition. It should be noted that these functions do not produce full images and can be very computationally expensive, especially when run on inappropriate images. Therefore, they are excluded from the function set for now. Future work will aim to add these in a proper manner.

Colour Space and Hyper-spectral Imaging Much of the previous work on imaging with GP has focused on the use of grey scale images. Often this is for performance considerations. But also this is out of consideration for how the images will be handled within the program.

In convolutional approaches on grey scale images, the evolved program will be given nine inputs representing the neighbourhood pixel values of a given location. The program will output a single value that represents the pixel value for that location. Expanding this to colour would require either treating each neighbouring value as a vector representing each of the colour channels (e.g. R,G,B) or increase the number of inputs (from 9 to 27 in the case of a simple colour image). By representing the colours in a vector, GP would need to be given appropriate functionality to use individual channels. Unless the representation allowed mixed data types, it is hard to think of a simple and robust method that would allow this Harding et al. [2012]. Further, treating each pixel as a vector may unnecessarily increase processing time. For example, if only one colour channel was needed to solve the problem, the other elements of the vector would still need to be processed, and stored.

Using a large neighbourhood introduces scaling problems, especially if a large number of inputs need to be considered. It is easy to imagine the GP programs would need to be large with a complex structure, and that this would be difficult to evolve. Similarly, approaches such as Neural Networks, would also need strategies to cope with a large number of inputs.

Both these issues would further be compounded if hyper-spectral or layered images were used. For example, in the ICPR dataset (see Section 3.4), colour images produced by different processes are provided, combining the different layers (which are also colour images) would require GP to process either 9 element vectors, or handle 81 inputs in the case of a simple convolutional method working on a 3x3 kernel.

In CGP-IP, all functions operate on single channel images. Each available channel is presented as an input to CGP-IP, and mutation selects which inputs will be used. It is also common to use different colour spaces when performing image processing. Images are typically considered, to be composed of Red, Green and Blue channels (the common colour space used in display devices and file storage). However, other colour spaces such as HSV (Hue, Saturation and Brightness) have useful properties that may be exploited by GP. For example, colour space is important in skin tone detection as skin has a limited range of hues [Forsyth and Fleck, 1997]. In CGP-IP the default treatment for colour images is to separate them into both RGB and HSV channels, and provide these as available inputs. A grey scale version of the image is also provided.

Function Set Each node can perform one function. Evolution, through the change of the genome, defines which one of over 60 unique functions is executed at each node. To include domain knowledge OpenCV is used to provide relevant functions. Tables A.1 and A.2 in Appendix A, list the functions available in CGP-IP. The function set is considerably larger than those typically used with GP approaches. This does not appear to hinder evolution, and again we speculate that the increased number of functions provides greater flexibility in how the evolved programs can operate.

Parameters As with other CGP implementations, CGP-IP does not require many parameters to be set. The main parameters are:

- Graph length (i.e. the number of nodes in the genotype), which is set to 50 in this case.
- Mutation rate, 10% of all genes in the graph are mutated when an offspring is generated. The threshold parameter is mutated with a probability of 1%.
- Size of mutations
- Number of islands, this depends on the available computing resources. It has been shown that the population can be split into separate islands to improve the performance of evolutionary algorithms [Ampatzis et al., 2011]. CGP-IP has been tested successfully from 1 to 24 islands. For all the experiments shown here, 8 islands are used.
- Number of individuals per island, which is set to 5 in keeping with the typical 1+4 evolutionary strategy used with CGP.
- Synchronization interval between islands. Here each island compares their best individual to the server's individual every 10 generations.

It is important to note that in the work presented herein the parameters have not been optimized other than by casual experimentation. It may be possible to improve the performance of CGP-IP by more carefully selecting these parameters. In particular, the mutation rate, genotype size and number of islands are expected to be the most important parameters to adjust.

All parameters are kept constant throughout the experiments presented below. Again, whilst it may be possible to improve performance for a given problem by optimising the parameters. The apparent parameter robustness is an important feature of this technique.

Accelerating Performance

Image processing is computationally expensive, and this makes the fitness evaluation the bottleneck in performance. In CGP-IP several different approaches can be used to address this problem.

Caching As CGP is a graph, it often uses the output values of a node more than once. See Figure 3.13 for an example. Storing the output of a function in memory for later use reduces processing time. This optimization is inherent to the CGP representation.

Similarly, it is possible to cache the results of an image processing operation on an image and store this for later use. In CGP-IP, as much caching is done as possible to reduce the CPU load.

Distributed Computing As in previous work on CGP [Harding et al., 2012], a parallel evolutionary strategy is employed. Multiple islands are used, where each island runs a 1+4 evolutionary strategy. At frequent intervals, the best individual per island is synchronised with a server. If the server contains a better individual, the island downloads a copy of that individual. If the island's best individual is better, then it uploads that to the server for other islands to use.

The use of distributed computing allows the processing to be spread across multiple computers. The parallel island evolutionary model has also been shown to improve evolvability.

GPUs Image processing is well suited to the parallel architecture of GPUs. Previous work has shown that when evaluating GP programs on images using GPUs a substantial speedup can be realised [Harding and Banzhaf, 2011]. However, the current version of OpenCV does not fully support GPUs. Future work aims at extending the CGP-IP capabilities to exploit GPUs.

Pruning Generated Programs Our implementation of CGP-IP generates human readable C#or C++ code based on OpenCV. Although CGP offers bloat free evolution, it often appears to leave redundant nodes in the evolved program. Examples of redundant nodes include operations that add 0 to every value, or produce a uniformly black image. Whilst the neutrality is important in the genotype, in the phenotype (i.e. the evolved program) it is undesirable.

Optimizing the generated code is performed by a pruning method, which identifies unnecessary operations. The process is as follows: for each active node in the CGP program, replace it with a NOP and re-run the fitness evaluation. If the fitness does not change (or improves), leave the modification in place, and move to the next node. If the fitness degrades, replace the instruction with a node that generates a completely black image, and retest. Again, keep the change if it does not affect the program's output. If it does alter the fitness, a final substitution attempt is made with a white image. If all of these changes degrade performance, the original operation is restored. It is typically found that this process reduces the number of used instructions, and hence reduces the execution time of the evolved program.

3.4 Experiments and Results: Detection

This section demonstrates the applications for CGP-IP. It contains descriptions of the experiments performed and results achieved. Several different domains were tested in a variety of experiments, applying CGP-IP with the same configuration, except for the fitness function for some tests, to various datasets (unless explicitly mentioned otherwise). The scenarios range from basic image processing (noise reduction), mitosis detection to robotic applications. In addition, CGP-IP was tested with images in the medical domain, these results can be found in Appendix B.

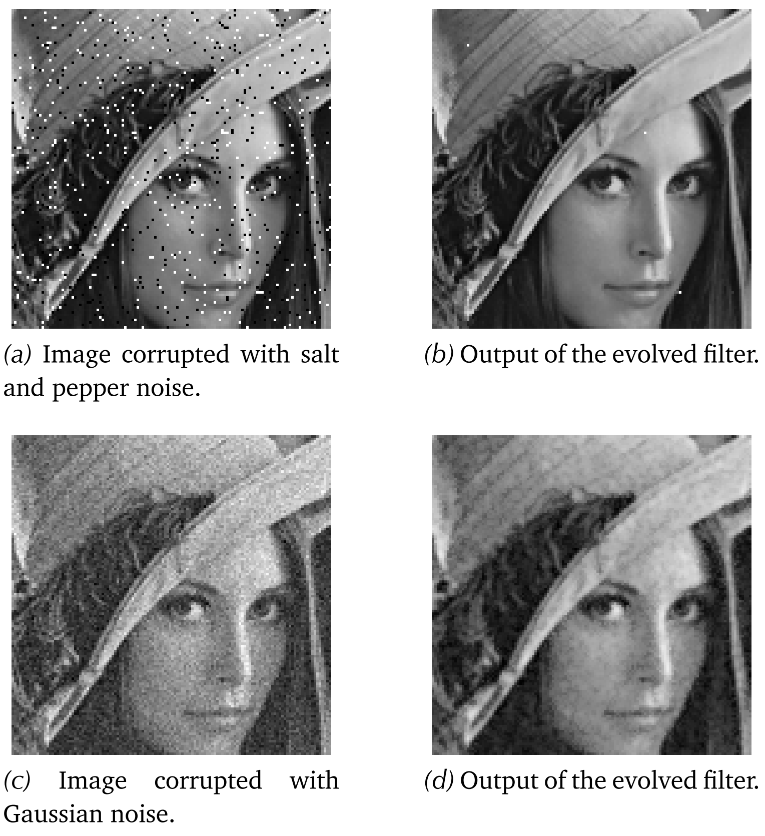

(a) Image corrupted with salt and pepper noise.
(b) Output of the evolved filter.
(c) Image corrupted with Gaussian noise.
(d) Output of the evolved filter.

Figure 3.14. Example result for noise removal for both 'Salt and Pepper' (a,b) and Gaussian noise (c,d).

3.4.1 Basic Image Processing: Noise Reduction

To demonstrate the capabilities of CGP-IP the well-studied noise reduction problem is visited. Several key examples using CGP exist (e.g. [Harding, 2008; Smith et al., 2005; Vasicek and Sekanina, 2007; Martínek and Sekanina, 2005]). These examples all use mathematical and logical operators to define a simple convolution operation. The experiment shows CGP-IP addressing both 'salt and pepper' and Gaussian noise.

The fitness of an individual is the average pixel difference between the output and the expected image (i.e. Type A). Ten experiments were run for each type of noise. Programs were trained on 4 images, and tested on 6 unseen images. For the 'salt and pepper' noise removal, 5% of the pixels are corrupted by setting them to either black or white. For the Gaussian noise problem, noise was added with standard deviation $\sigma = 16$.

From Table 3.1, it can be seen that CGP-IP compares well to previously published techniques. It is hard to do an exact comparison of the results as previous work may use a different subset of the images, for example in the Gaussian noise validation set, the mean pixel error per image varies from 8.6 to 14.4, suggesting that the results are highly dependent on the image being processed. Figure 3.14 shows that the evolved filters remove most of the noise, with only a small amount remaining.

Table 3.1. Fitness results for the noise removal problem. The fitness of an individual is the average pixel difference between the output and the expected image. Results are comparable to previous work on this problem; CGP on GPU [Harding, 2008], CGP on FPGA [Martínek and Sekanina, 2005], IRCGP and ECGP [Smith et al., 2005]

Approach	Salt and Pepper	Gaussian
CGP-IP (Average)	0.57	11.00
CGP-IP (Std. dev.)	0.16	0.53
CGP-IP (Best)	0.41	9.58
CGP on GPU (Average)	0.39	n/a
CGP on FPGA (Best)	0.48	6.43
IRCGP (Best)	n/a	6.32
ECGP (Best)	n/a	6.32

3.4.2 Industrial: Detecting Surface Defects in Steel Production

Automated detection of defects from imaging is a common application of computer vision. Figure 3.15 shows the results of a trained CGP-IP individual on a set of defects encountered during the manufacture of steel. These are images unseen during training. Training was performed on 50 images, containing a variety of defects. For testing another set of 50 images was used. The supplied 'region of interest' (second column) only crudely, and sometimes misleadingly, labels the data. However, CGP-IP is able to quickly learn a program to identify the exact regions in the input image that contain the defects (third column). The evolved C#program for the defect detection is relatively simple (Listing 3.1). The program can be easily inspected, and modified, by engineers. The code is also simple enough to be reimplemented onto different optical inspection hardware. The evolved program is also able to operate at high speeds (in this case under 1ms per image), which is beneficial in a continuous production environment. To further facilitate the reuse of the code the `icImage` class (of the *icVision* framework) is used to encapsulate the image operations and wrap OpenCV functionality.

Listing 3.1. The generated C#code for detecting steel defects visually.

```
public icImage RunFilter()  {
    icImage node1 = InputImages[0];
    icImage node2 = node1.sobelx(3);
    icImage node3 = node2.unsharpen(-3, 1);
    icImage node5 = node2.dilate(1);
    icImage node6 = node5.canny(0.1);
    icImage node7 = node6.max(node5);
    icImage node8 = new icImage(-0.34d, ImageWidth, ImageHeight);
    icImage node10 = node3.min(node7);
    icImage node13 = node8.add(node10);
    icImage node18 = node13.sobelx(1);
    icImage node19 = node13.unsharpen(10, 3);
    icImage node35 = node18.dilate(1);
    icImage node47 = node19.max(node35);
    icImage node49 = node47.localMax(1);
    return node49.threshold(140.2045f);
}
```

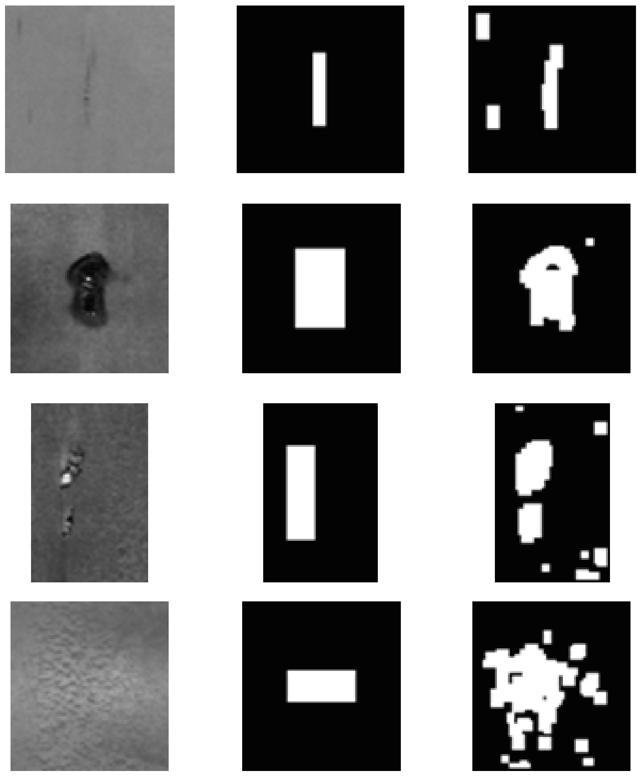


Figure 3.15. Examples from the steel defect dataset. In the first column are the input images, the second column contains the 'region of interest' where the defect has been labelled and the third column shows the output from CGP-IP.

3.4.3 Terrain Classification: Recognising Terrain from Mars Exploration Rover (MER) Images

As a first example more aimed at robotics, CGP-IP is applied to visual detection and identification of rocks and rock formations in images from Mars rovers. Automatically classifying terrain such as rocks, sand and gravel from images is a challenging machine vision problem. Using data from mobile robots (on other planets) our system learns classifiers and detectors for different terrain types from hand-labelled training images. The learned program outperforms currently used techniques for classification when tested on a panorama image collected by the Mars Exploration Rover Spirit. This was presented at the International Symposium on Artificial Intelligence, Robotics and Automation in Space [Leitner et al., 2012b].

In mobile robotics terrain classification is of interest to allow for more autonomous decision making about traversing. Identifying obstacles and suitable surfaces to drive on is important in order to prevent the robot getting stuck, whilst avoiding damage. A particularly interesting example can be found in robotic space exploration, which imposes some additional constraints. For example, the issue of having to do image processing on a low power system, without relying on high-latency, low-bandwidth communication to remote processing on Earth. A review of current approaches can be found in Bajracharya et al. [2008].

Halatci et al. [2008] have previously published work on classifying terrains for planetary rovers from images and also from vibrations sensed at the vehicle. They aim to classify three separate classes of terrain during a field test: rocks, beach grass and sandy terrain. Both low and high level features were used for classification. Using the vision approach alone they achieve an average accuracy of 77%, which was increased to 84% by fusing the vibration data with the image data. Their classification is reported to take approximately 29 seconds per 512×512 pixel frame on a Pentium 1.8 Ghz desktop PC.

Shang and Barnes [2012] investigated terrain classification from visual information only. They investigated fuzzy-rough feature selection together with support-vector-machines to classify terrain in the *McMurdo* panorama image, collected by the Spirit rover. The (composed) image in approximately true colour is publicly available from NASA.[3] It consists of 1,449 separate images and representing a raw data volume of nearly 500 megabytes.

It shows the surroundings of the rover and includes dark, porous-textured volcanic, as well as, brighter and smoother-looking rocks. These are embedded in a sandy terrain with ripples and gravel (mixture of small stones). In their paper several classes of terrain were defined and hand labelled, then a low-level feature extraction was applied to generate 36 features per pixel. An SVM is then used to classify the pixel based on those features. With their technique a classification of up to 92% was achieved on hand-selected parts of the panorama. No runtime information is given for generating those 36 local features and classifying the terrain. CGP-IP is used to perform a similar, binary classification of rocks and sand in images extracted from the Mars rover panorama.

Eight images were manually labelled (as there are no publicly available datasets), with six used for training and two used for validation. The MCC fitness function (Type B) was used.

CGP-IP was able to find programs that discriminate between the two terrain types, and the statistical results can be found in Table 3.2. As we used different datasets to previously published work, we cannot compare directly – but it

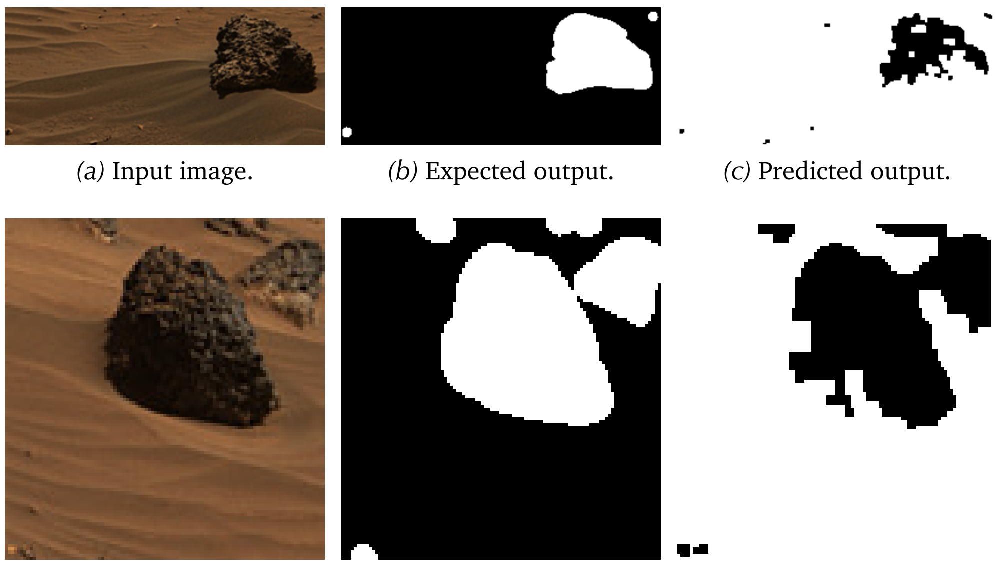
[3] `http://pancam.astro.cornell.edu/pancam_instrument/mcmurdo_v2.html`

Table 3.2. Results for the Mars terrain dataset for classifying rock and sand. Classification rate is per-pixel. Ten runs were performed. The average number of evaluations to convergence was found to be 8,538.

	% Classification Accuracy	MCC (Fitness)
Average	90.3	0.78
Minimum	81.6	0.61
Maximum	93.6	0.86
Std. Dev.	3.63	0.07

appears this approach is competitive with prior work. Figure 3.16 shows the validation images from the highest performing individual. The evolved program also picks out rocks not labelled in the original image. Inspecting the training data, this is also observed there – again suggesting CGP-IP is tolerant to poorly labelled data.

Speed and simplicity of processing is important in many embedded systems. On a single core of a modern desktop PC, the evolved filters run within 2ms, and as the program is largely linear in form, the memory requirements are low. This may make this approach highly suitable for implementing in constrained computing environments. The simplicity of the solution can be seen in Listing 3.2.


(a) Input image. *(b)* Expected output. *(c)* Predicted output.

(d) Input image. *(e)* Expected output. *(f)* Predicted output.

Figure 3.16. Examples of evolved filters from the Mars terrain dataset.

Listing 3.2. The generated C#code for detecting a specific type of rock.

```
public override GpImage RunFilter()  {
    icImage node0  = InputImages[6].absdiff(InputImages[2]);
    icImage node4  = node0.normalize();
    icImage node5  = InputImages[1];
    icImage node12 = InputImages[0].shiftRight();
    icImage node14 = node4.sub(node12);
    icImage node16 = node4.subc(-4.8662);
    icImage node20 = node16.max();
    icImage node21 = node14.erode(3);
    icImage node23 = node5.absdiff(node20);
    icImage node32 = node21.sqrt();
    icImage node35 = node23.erode(1);
    icImage node41 = node32.localMax(2);
    icImage node49 = node35.add(node41);
    return node49.threshold(92.06558f);
}
```

In the example, `InputImages` is an array of type `icImage`. The elements in the array are: Grey Scale version of the colour input image, the *Red*, *Green* and *Blue* components and the *Hue*, *Saturation* and *Value* components. Hence `InputImages[6]` is the *Value* component, and `InputImages[1]` is the *Red* component of the original, colour input image.

Learning to Classify Terrain From Images The same 5 segmentation classes as defined by Shang et al. [2011] to classify the terrain from visual inputs are used here. Instead of hand-labelling all test images their results are employed to train our classifiers. The slices, taken from the full panorama, used in their publication vary in size from 400 × 400 to 600 × 600 pixels.

Figure 3.17 shows the segmentation performed by CGP-IP in two sections of the panorama. Gravel is shown in yellow, sand in blue. Various types of rocks are detected and shown in green, red and purple. Our approach is quite fast as each class needs about $50 - 150ms$ per slice. To speed up execution those could be performed in parallel. In comparison Halatci et al. [2008] require 29 seconds for a 512×512 image frame. In Shang et al. [2011] 816 labelled feature patterns are needed to perform the classification and no runtime information is given. Only 7 hand-labelled slices are used as inputs to learn the segmentation. An increase in number of hand-labelled data is believed to allow for higher accuracy. Since

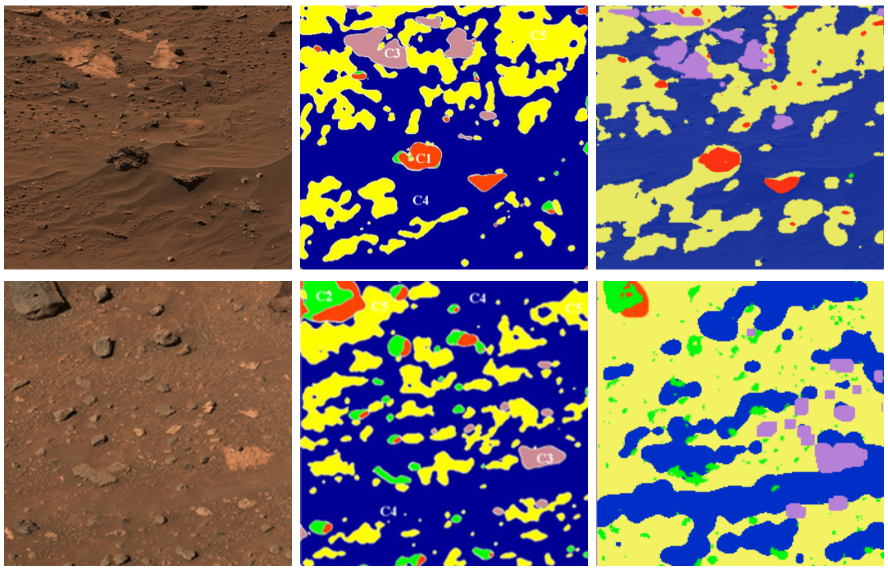


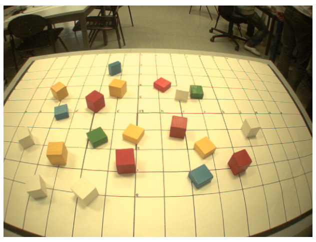
Figure 3.17. Example of terrain classification task from a small section of the panorama (left) in comparison with the results from Shang et al. [2011] (centre). The right shows the results from the classification performed by CGP-IP.

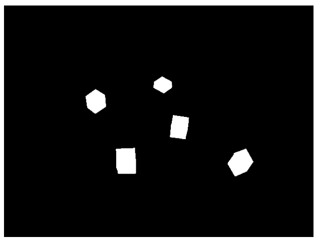
there is no dataset with 'ground-truth' available it is therefore hard to objectively compare the accuracy of the various methods. Especially the definition of sand vs. gravel vs. small rock can be tricky. This is visible in Figure 3.17, where these areas vary quite a bit between the two solutions.

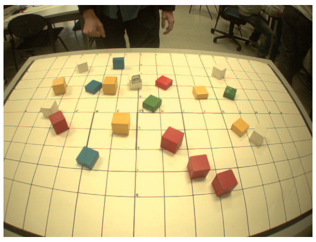
3.4.4 Robotics: Object Detection with the *iCub* Humanoid

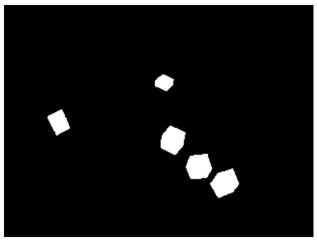
Vision in robotic systems is a challenging problem. In industrial robotics, scenes tend to have well controlled lighting and the objects being examined are well characterized. Further, the lighting and viewing angles can be optimized to meet the vision requirements. For robots that work in more 'real world' environments, lighting and viewing angles are often variable. Such vision systems are expected to work in cluttered environments, with lots of different objects visible in the scene. CGP-IP is applied to detect objects in images supplied by the *iCub*'s cameras. The approach is able to work robustly in different lighting conditions, and when the target object is moved or rotated.

To train CGP-IP, a number of images are collected from the *iCub* cameras. For each frame, the target object (and other objects in the scene) are repositioned. Hence our training set implicitly contains multiple views of the object with different angles, scales and lighting conditions. Each image in the training set is then


(a) Input image used for training.


(b) Expected output for training example.


(c) Example validation input image.


(d) Expected output.

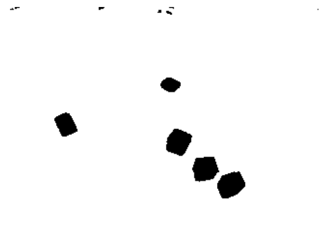

(e) Output of the evolved filter.

Figure 3.18. Experimental result for the block detection with the *iCub* cameras. A single training image was used (a). (e) shows the output of an evolved filter that accurately segments out the red blocks.

hand segmented to highlight the target object. It should be noted that a rough segmentation still allows CGP-IP to learn efficiently, which reduces the need to spend time performing an accurate segmentation. The filters are trained to produce a binary classification (i.e. target object and, not object) using the MCC based fitness function (i.e. Type B).

The first dataset tested was to detect toy blocks. Although these blocks have highly saturated colours, and an obvious solution would be a simple colour filter, the evolved filter has to discriminate the blocks from other objects in the scene that have the same colours. As expected, this problem is trivial for CGP-IP to solve. For training, only one (similar) image was used, as shown in Figure 3.18. Validation was performed on 4 other images. In each image the blocks are randomly placed. In the first run, CGP-IP was able to obtain a fitness score of 0.0647 (99.8% classification accuracy) on the validation images in just 20,000 evaluations. The few false-positives in the pixel classification could be easily removed by manually adding a median filter to the program.

As the problem is relatively simple, the generated program is also simple to understand: Listing 3.3 shows the program to use both the *Red* component

Listing 3.3. The generated C# code for detecing toy blocks.

```csharp
public icImage RunFilter()
{
    icImage node2 = InputImages[1].mulc(1.295);
    icImage node8 = InputImages[4].shift(-3, 0);
    icImage node10 = InputImages[4].min(node8);
    icImage node11 = InputImages[4].sub(node2);
    icImage node13 = node10.shift(1, 0);
    icImage node14 = node13.gauss(5);
    icImage node19 = node14.mul(node14);
    icImage node32 = node11.localAvg(4);
    icImage node43 = node32.shift(14,-14);
    icImage node49 = node19.avg(node43);
    return node49.threshold(195.4248f);
}
```

(`InputImages[1]`) and the *Hue* component (`InputImages[4]`) of input. The shift functions are used to produce edge detectors, whilst the `gauss` operation smooths the images. It should be noted that the output is inverted (the predicted classes are flipped around), but the MCC score for the fitness function allows for this scenario. The evolved program could be slightly modified to correct this by changing the final `threshold` operation to a `thresholdInverse`.

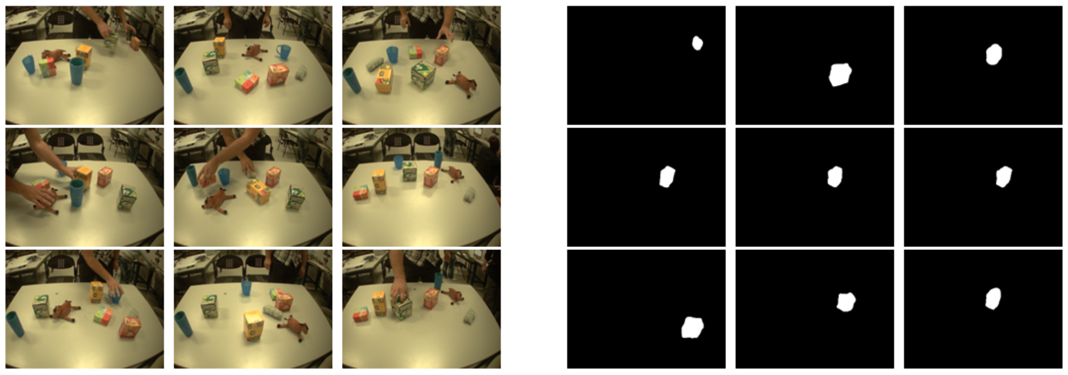


Figure 3.19. Training images for learning of filters for the *iCub*. The set on the left shows the images from the camera. The set on the right shows the binary classification as determined by a human, where a particular box is highlighted in white. For each frame the position of the items on the table is shuffled.

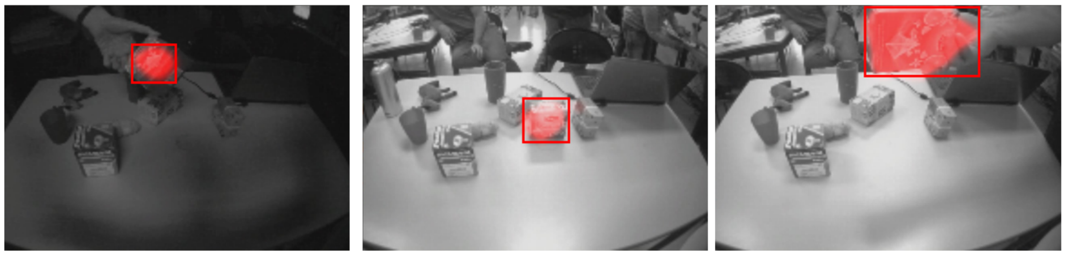


Figure 3.20. Example images of an evolved filter running in real time on the *iCub*. Considering the limitations of the training set, the evolved filters show good ability to generalise to variations in lighting, scale and rotation. Full video at: `http://www.youtube.com/watch?v=xsz0Cj4A1eA`

The training set for texture objects problem are shown in Figure 3.19. To train, a number of images are collected from the *iCub* cameras. In each one the target (and other objects) is repositioned. Hence our training set implicitly contains multiple views, different angles, scales and lighting conditions. Nine training images were used, with a single object of interest per image (training a single filter). Figure 3.20 shows some examples of the evolved filter running

Listing 3.4. The generated code from CGP-IP for detecting the red tea box.

```
icImage* TeaBoxDetector::runFilter() {
    icImage *node0 = InputImages[0];
    icImage *node1 = InputImages[1];
    icImage *node2 = node0->erode(1);
    icImage *node3 = node2->ShiftDown();
    icImage *node4 = node0->absdiff(node2);
    icImage *node5 = node0->avg(node2);
    icImage *node8 = node1->erode(3);
    icImage *node9 = node3->sub(node8);
    icImage *node12 = node4->add(node4);
    icImage *node13 = node5->min(node12);
    icImage *node16 = node13->absdiff(node9);
    icImage *node24 = node16->sub(node9);
    icImage *node25 = node24->gauss(15);
    icImage *node38 = node25->gauss(15);
    icImage *node39 = node38->threshold(64);
    return node89;
}
```

on the robot (see Listing 3.4). It can be seen that the evolved filter is able to cope with variations in scale, orientation and lighting. Each filter for detecting objects can also be visualized as a graph, as in Figure 3.21) for the detection of a blue cup. Unique programs, allowing for identification as well as detection, were evolved for a variety of objects of interest, e.g. different cups, tin cans, kids blocks and tea boxes.

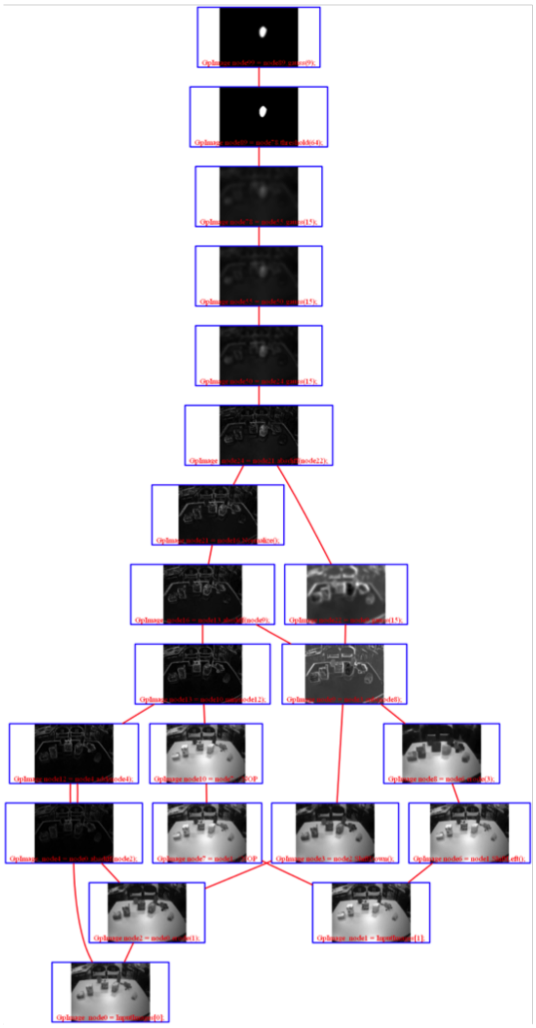


Figure 3.21. The filter's workflow is clearly visible when represented as a graph structure. It allows an engineer/programmer to see the operation performed at each node. The top-most picture shows the final output, the binary segmentation for a single object (a blue cup in this case).

Robot Hand Detection on Humanoids CGP-IP was then applied to detect the robot's own hands and fingers in the images – an important prerequisite for eye-hand coordination and a rather complex object to visually segment. Previous approaches make use of external systems or markers to provide the position of the hand. The detection of hands in robotics has so far been mainly focused on detecting human hands in camera images. For example, Kölsch and Turk [2004] presented a method for robustly classifying different hand postures in images. These approaches usually use the quite uniform colour of the skin [Zhu et al., 2000], motion flow [Cutler and Turk, 1998] and particle filtering [Isard and Blake, 1998]. In contrast, the method herein using CGP-IP, it does so solely using its own camera images and does not require any external systems or markers. These results were first shown at the Congress on Evolutionary Computation [Leitner et al., 2013c].

At first the approach was verified by visually detecting the hands of the *Nao* robot, which are of a less complex design and also simpler in appearance (more details about the *Nao* detection can be found in Leitner et al. [2013c]). In comparison to the *Nao*'s hand, the *iCub* sports a rather complex end-effector (Figure 3.22).

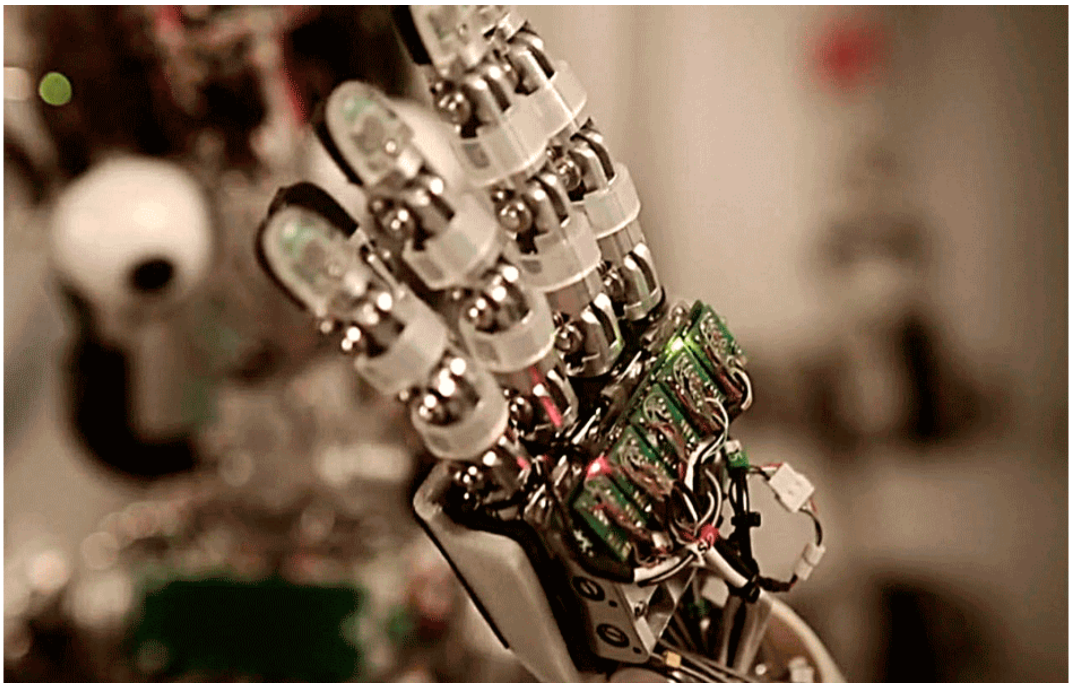


Figure 3.22. The *iCub*'s hands and fingers are quite complicated mechanically and complex to detect visually.

The detection of the mechanically complex hands and fingers seems to be a much harder computer vision task. These are the most complex objects so far detected with evolved CGP-IP filters. There has only been one previous effort to apply machine learning to detect the *iCub*'s hands [Ciliberto et al., 2011b]. As above, we collect a dataset from the real robot. This time the images were gathered while the robot was moving its arms around, in the visible workspace. A handful of pictures was hand-labeled and used as a training set.

The detection of the hand is split into two separate problems: the identification of the fingertips only, and, secondly of the full hand. This separation allows us to create a finer level of control for tasks, such as, grasping, that involve controlled motion of the fingers.

The fingertips are made out of a black rubber protecting the touch sensors within. Figure 3.23 shows the result of an evolved filter detecting the fingertips of the *iCub*'s hands. To make sure that the solution is not just identifying any black part the images contain black objects in the background, such as, chairs and computer cases. Due to the stochastic nature of the evolutionary approach we ran multiple tests. The best solution (out of 10 test runs) for filtering out the fingertips was found rather quickly; after only a bit more than 8 minutes of evolution on a standard desktop PC (about $55,000$ individual evaluations).

To detect the full hand is a more complicated task. Here we used 10 images to train the filter and an additional four to validate each solution against unseen data. For this more complicated task the evolution of the best solution (out of 10 tests) took more than $5,700,000$m evaluations (about a hundred times more than for the fingers alone), taking about 12 hours on a desktop computer. In Figure 3.23 the performance of the learnt filter is shown. Generally a very good detection can be seen, although there are some minor issues. For example, a reduced level of detection can be seen around the edges of the frame. As we will be gazing upon a specific object we want to manipulate, the precise control will be of importance when the hand reaches close to the centre of the image frame. Another issue seems to be the striking difference in visual appearance between the front and back side of the hand. This can be seen in the second to the right image in the middle row of the figure. Again in most of the planned scenarios for object manipulation the back of the hand will be visible. Therefore we do not see this as an important issue. Furthermore if a more thorough detection, also of the front side, is required another separate filter just for the front can be evolved. By combining the two separate detectors, for the fingertips and the hand, we get a very precise, almost full identification of the *iCub*'s hands. Separate programs resulted in better detection of the fingers and the hand (Figure 3.23).

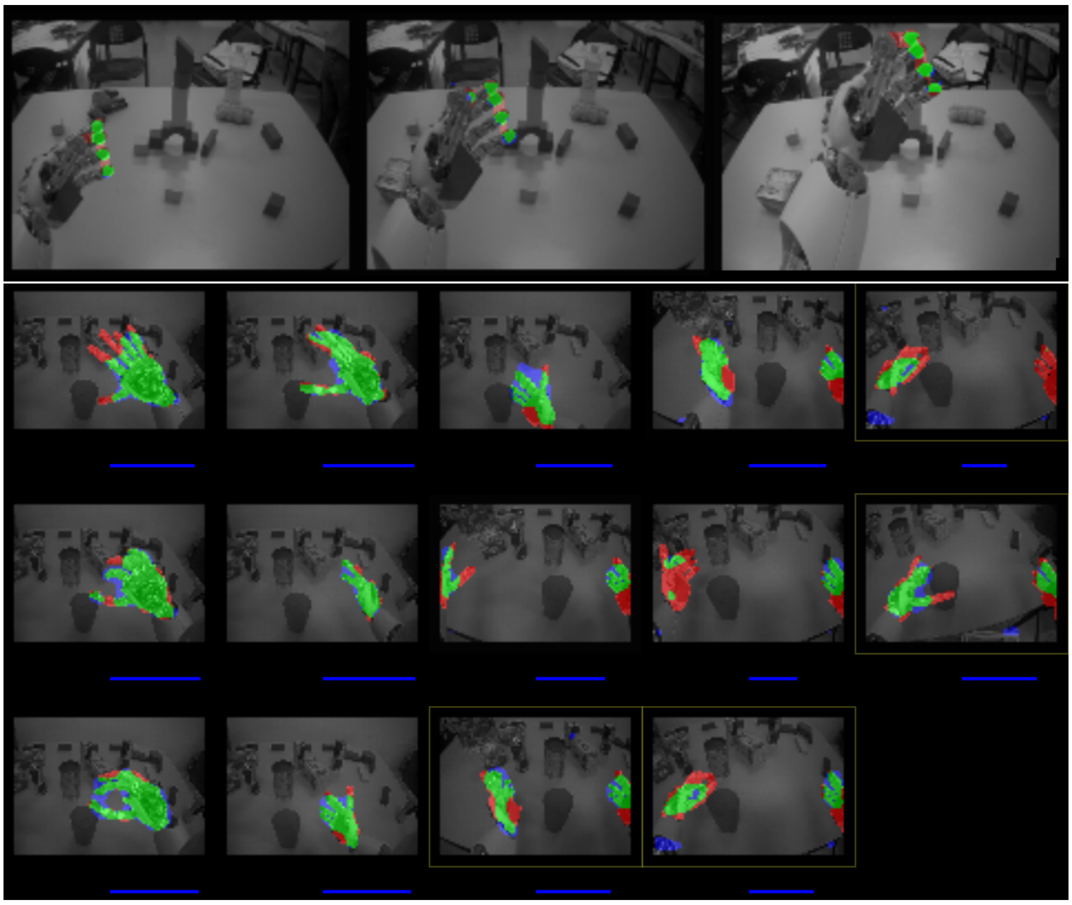


Figure 3.23. The detection of the *iCub*'s own fingertips and hands. Green indicates correct detection, red and blue respectively show not and wrongly detected pixels.

3.4.5 Robotics: Autonomous Learning of Object Detection

CGP-IP is based on a supervised learning technique, i.e. it needs a set of hand-labeled training samples. Only recently researchers started to look at the questions how a robot can learn to recognise objects in a largely autonomous fashion [Kim et al., 2006], how learning can be made fully online [Kirstein et al., 2008], and how the need for a human teacher can be minimised [Gatsoulis et al., 2011].

In collaboration with the Frankfurt Institute of Advanced Studies (FIAS) and building on CGP-IP the design of a autonomous vision system, that tries to answer these questions on the *iCub*, was started. Together we developed a technique for our humanoid robot to learn object representations by itself. The developed visual system actively explores the unfamiliar environment and provides the robot with the ability to learn visual representation for objects in the scene in an autonomous fashion. A prototype version of this system was presented at the International Conference on Developmental Learning and Epigenetic Robotics (ICDL/EpiRob) [Leitner et al., 2012a]. The following experiment show the achieved results using a combination of techniques required to create a more autonomous learning system for CGP-IP:

- To allow for autonomous generation of these unique identifiers a biologically inspired scene exploration approach is applied, then
- a feature based approach is deployed to create segmented images around a salient point, and finally
- this pre-segmentation provides the needed inputs to CGP-IP for learning robust filter using the same approach as in the previous experiments.

Scene Exploration for Autonomous Learning

A scene exploration approach is applied to find the majority of the objects in the workspace. To do so the robot looks around using a attention mechanism, i.e. the robot looks at objects, one after the other, based on the saliency of their features. Together with a stereo segmentation scheme this exploration technique avoids manual supervision for segmenting the objects in the scene required for training CGP-IP.

Bottom-up attention mechanism A neuro-morphic model developed by Itti et al. [1998] is employed. It aims to identify the elements of a visual scene that are likely to attract the attention of human observers. In this model the 'stimulus conspicuity' over the visual scene are topographically encoded in the form of a saliency map using a bottom-up control strategy. The different visual features that contribute to attentive selection of a stimulus (colour, intensity, orientation, motion etc) are combined into one single two dimensional map integrating the normalised information from the individual feature maps into a global measure.

Images from both the eyes are processed to obtain saliency maps which assign a salience value to each pixel in the image. These maps are used by the decision module to select the single most salient point in the scene (Figure 3.24a) and gaze upon it before segmentation can begin. As this technique is quite robust to variations, the same point will be selected at every run if the scene does not change. To avoid this and actually be able to detect different objects, a temporary inhibition of the saliency map around the winning location is applied. This is done by subtracting a Gaussian kernel at that location in the map, effectively creating a highly non-salient point. Recently visited location will be inhibited, in decreasing order, to avoid bouncing back between the two most salient points in the scene and find the next most salient object.

Top-down modification of attention In order to avoid constant switching between the two most salient locations, we also introduce a top-down inhibition of recently visited locations. To this end, the absolute 3D coordinates of the visited locations are saved in the memory and they are mapped onto the pixel coordinates on images from the cameras in their current positions to know the locations for inhibition on saliency map. In our experiments, a list of the 5 most recently visited locations is maintained and close-by points are inhibited for the next gaze shift (Figure 3.24b).

Feature-based Pre-Segmentation

The feature detection is performed by the FAST corner detection algorithm, as it provides both fast and high quality features [Rosten et al., 2010]. The local descriptors were then calculated using Gabor wavelets.

Feature Extraction Gabor wavelets for calculating the features which are convolution kernels having the shape of plane waves restricted by a Gaussian envelope function as suggested by Wiskott et al. [1997]. The choice is also motivated by the fact that these functions have similarity in shape to the the receptive field of simple cells found in the visual cortex of vertebrate animals [Pollen and Ronner, 1981; Jones and Palmer, 1987]. At each interest point a 40-dimensional feature vector is extracted, which is referred to as "Gabor-jet". This vector results

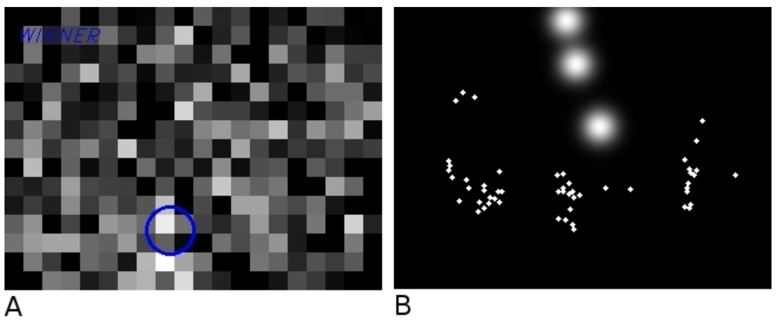


Figure 3.24. (a) Saliency map and the winning location (blue circle). (b) Locations of the objects on the table and already gazed upon locations.

from filtering the image with Gabor wavelets of 5 scales and 8 orientations. A Gabor-jet $\mathcal{J}$ obtained by convolving a small patch of an image at a given location $\mathbf{x}$ with Gabor wavelet with a specific orientation and scale can be mathematically described by the following equation:

$$\mathcal{J}_j(\mathbf{x}) = (\mathbf{I} * \psi_j)(\mathbf{x}) \tag{3.10}$$

The complex Gabor wavelets $\psi_j(\mathbf{x})$ are defined as

$$\psi_j(\mathbf{x}) = \frac{\mathbf{k}_j{}^2}{\sigma^2} exp\left(\frac{-\mathbf{k}_j{}^2\mathbf{x}^2}{2\sigma^2}\right)\left[exp\left(i\mathbf{k}_j\mathbf{x}\right) - exp\left(\frac{-\sigma^2}{2}\right)\right] \tag{3.11}$$

where $\mathbf{x}$ is image location and $\mathbf{k}_j$ is the wave vector. Hence, $\psi_j(\mathbf{x})$ that is located at $\mathbf{x}$ has the shape of plane waves with wave vector $\mathbf{k}_j$ restricted by a Gaussian envelope function (for details see Leitner et al. [2012a] or Wiskott et al. [1997]).

Feature Matching The extracted features were then matched between the two camera images. This is required as the stereo information allows to better segment the object from the background. Correspondence between the left and right eyes is calculated by comparing the local descriptors around each feature using normalised Euclidean inner product.

To segment a potential object (at the centre of the robot's gaze) the extra information by the second camera is incorporated. Correspondences are found between features detected in the left and right images by exhaustively comparing the Gabor-jets extracted at interest points. The normalised Euclidean inner product is applied to define the similarity between two jets $\mathcal{J}_1$ and $\mathcal{J}_2$:

$$S(\mathcal{J}_1, \mathcal{J}_2) = \frac{\mathcal{J}_1^T \mathcal{J}_2}{||\mathcal{J}_1||\;||\mathcal{J}_2||} \tag{3.12}$$

Each interest point in the left image is associated with the best matching interest point in the right image if the similarity S between the two jets is above a preset threshold (0.95 in these experiments). Figure 3.25 shows the result of such a matching during an experiment.

Stereo Segmentation The matched interest points are clustered into groups, using a greedy scheme based on their location in the image and disparity. Figure 3.26 shows two examples of this clustering, with cluster of dots of different colours belong to different objects. The object at the centre of the gaze is segmented (white dots) from other objects using spatial location and depth. This feature segmentation only provides a rough estimate for the correct binary segmentation.

To refine the segmentation the features are separated into on-object and outside-object categories (white vs. other colours). A minimum bounding rectangle enclosing on-object locations with an additional 20 pixel margin on every side is considered for the pre-segmentation (Figure 3.26b). These masks are then being used to build a training set for our CGP-IP based learning approach. The pre-segmented image can also be sent as input to the non parametric recursive watershed segmentation algorithm [Wegner et al., 1995] in OpenCV to provide the pixels belonging to the object (Figure 3.26c). The result of this algorithm is then comparable with the CGP-IP technique.

Training For Robust Image Segmentation

CGP-IP is a supervised learning technique. An obvious limitation is that the training images need to be provided. In our first object detection experiments with CGP-IP on the robot these images were hand-labelled in a rather tedious and time-consuming fashion. We noticed that the area specified by the labels did not need to be very precise as our technique seems to be robust to some errors in the provided bounding labels. The evolved solution was able to segment the object according to its outline rather than the label provided.

Extracting information from an image that allow for robust object detection is not an easy task. Figure 3.27 shows the same tea box and the FAST features extracted in two frames roughly 5 seconds apart. The features are quite different, making it hard to know whether this object is the same in both images.

To overcome this we use the features detected as rough segmentation and to

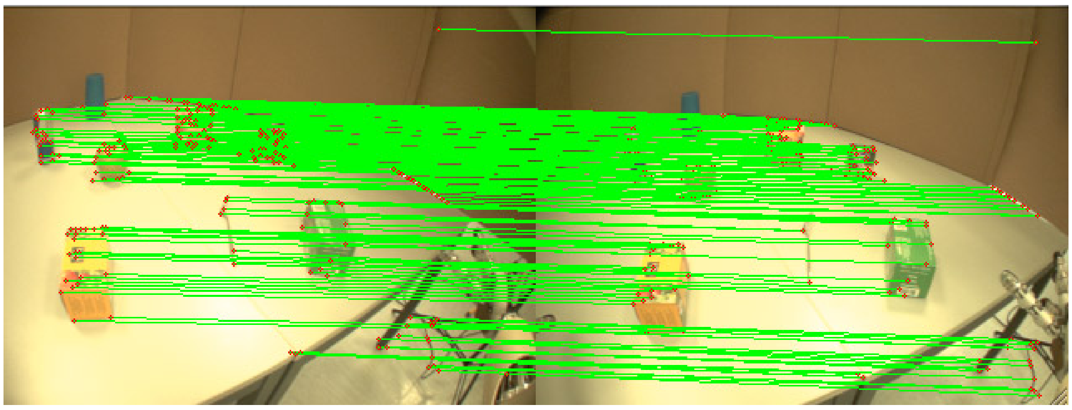


Figure 3.25. Feature matching between left and right camera images.

build a training set for our CGP-IP approach. A handful of examples, in which the object of interest has been correctly segmented, are used as a training set for CGP-IP. The selection of this training set is of importance for the capabilities of the found solution. If chosen correctly, the resulting programs are very robust to changing light conditions, as shown previously for segmenting a tea box (Figure 3.20). CGP-IP quickly learns, in about 1000 evaluations, how to segment the green tea box. The evolved program detects the object with the same filter in both images allowing for unique identification (Figure 3.27) . The resulting C++ code, shown in Listing 3.5, performs the detection on the hardware in real-time (below the frame update of our robot's cameras). As these evolved programs are largely linear in form, the memory requirements are low. Speed and simplicity of processing is important in many embedded systems, making this approach suitable for implementation in constrained computing environments.

Once the training is complete, the learned identifiers are tested by placing the objects in different locations on the table, including different poses and varying

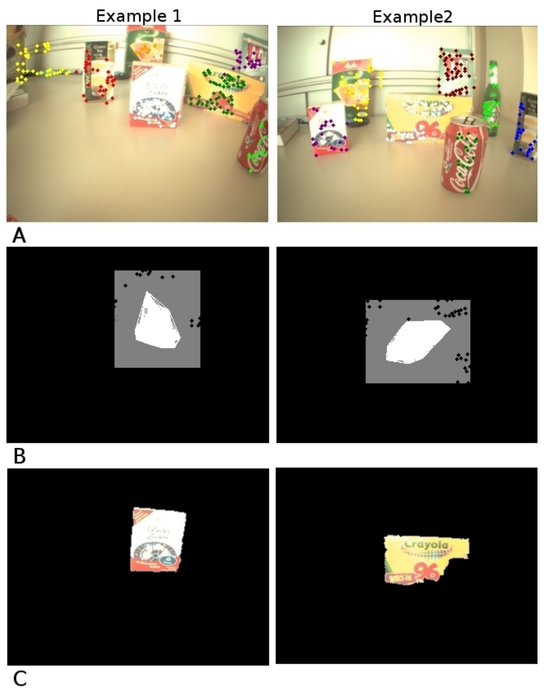


Figure 3.26. Two examples of creating pre-segmentation masks. (a) First the features, for which a matching exists, are clustered by location. (b) These clusters are the initialization for segmentation (based on the convex hull of the features detected). The grey area depicts the minimum bounding rectangle plus an extra 20 pixel margin. (c) The segmented object by using a watershed algorithm.

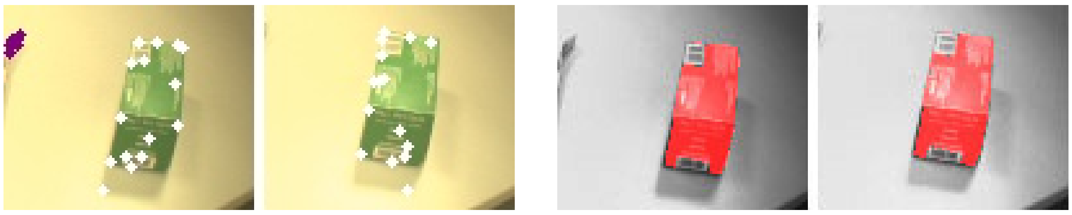


Figure 3.27. The FAST features (in white) found of an object is seen in the first two images. The tea box as detected with a learned filter using CGP is visible in the last two images. The binary output of the filter is used as a red overlay.

Listing 3.5. The generated code from CGP-IP for detecting the green tea box.

```
icImage GreenTeaBoxDetector::runFilter() {
    icImage node0 = InputImages[6];
    icImage node1 = InputImages[1];
    icImage node2 = node0.absdiff(node1);
    icImage node5 = node2.SmoothBilataral(11);
    icImage node12 = InputImages[0];
    icImage node16 = node12.Sqrt();
    icImage node33 = node16.erode(6);
    icImage node34 = node33.log();
    icImage node36 = node34.min(node5);
    icImage node49 = node36.Normalize();
    icImage out = node49.threshold(230.7218f);
    return out;
}
```

lighting conditions. The learned programs can also be used to build a model of the environment. This is described in more detail further down (Figure 5.1).

This section detailed the CGP-IP implementation, a technique for performing object detection by image segmentation. First the efficacy was shown on standard benchmark test (noise removal) before looking closer at object detection in robotic applications. The aim for a robust (e.g. to changing light conditions), readable (producing C++ code) and adaptive (relearn - improve with updated observations) system was shown. It is worth mentioning that object detection, identification, and tracking in CGP-IP is currently implemented to use only single frames. The performance can most likely be increased further by using information from multiple, sequential images, such as, optical flow, motion tracking, or just simply propagating information about the segmentation to the next frame.

To further improve the detection manipulation plays an important role. A prerequisite, the need to know where the object is in the world (with respect to the robot), is explained in the next section.

3.5 Spatial Perception and Object Localization

To localize objects in the environment the *iCub* has to rely solely on a visual system based on stereo vision, similarly to human perception. Its cameras are mounted in the head of the robot therefore the method for localization must be able to cope with motion to work *while* the robot is controlling its gaze and upper body for reaching. As part of my thesis work I investigated how spatial representation can be learned using visual perception only [Leitner et al., 2012d,f]. A machine learning based approach to enable spatial perception based on vision is proposed herein. It provides a humanoid robot with a method to estimate the position of objects relative to itself in 3D Cartesian space.

The most logical choice of coordinate system, from a planning and control standpoint, in our case, is a world reference frame originating from the robot's hip (*CSWorld*). To transform coordinates from the eyes to the world coordinates an accurate kinematic model of the robot is necessary.

While approaches based on projective geometry have been proven effective under carefully controlled experimental circumstances, they are not easily transferred to robotics applications. For the *iCub* platform several approaches have previously been developed. One of these methods is a biologically inspired approach mimicking the human retina [Traver and Bernardino, 2010]. Another, the 'Cartesian controller module', provides basic 3D position estimation functionality [Pattacini, 2011] and gaze control. This module works very well on the simulated *iCub*, however on the hardware platform multiple sources of noise and errors exist. It therefore generates absolute errors in the 2-4 cm range, in an area where the humanoid can reach and manipulate objects.[4]

Machine Learning Approach

Two biologically inspired machine learning approaches were investigated on how well they can 'learn' spatial perception on a humanoid – a feed-forward artificial neural network (ANN) and a genetic programming (GP) approach. Both these techniques use supervised learning, i.e. they require a dataset including both inputs and outputs (the 'ground truth'). More formally, the task is to estimate the position of an object $p \in \mathbb{R}^3$ in the robot's reference frame (*CSWorld*) given an input, also called feature vector, $v \in \mathbb{R}^n$. This vector contains the state of the robot as described by 9 joint encoder values (i.e. the nine controlled DOF) and the observed position of the object to be localized in both camera images. Two different sizes of v are used in our experiments, $v \in \mathbb{R}^{13}$, where the visual

[4]Even bigger errors are seen at the edge of the frame or further away from the robot.

information is represented by the u, v coordinates in each eye, and $v \in \mathbb{R}^{21}$, where additionally visual information (the bounding rectangle) is provided to the ANNs.

Artificial Neural Networks Approach An ANN is an information processing technique inspired by the way the human nervous systems processes information. Inspired by the neurons in human brains, which receive signals — electric impulses transmitted by chemical processes — from other neurons, modify the signals and forward them to their connections. An artificial neuron in comparison is a computational unit with many inputs and one output. The neuron 'fires' based on the input pattern received (Figure 3.28). The key difference of artificial neural networks is in the structure of the system. They are a network of interconnected firing neurons, with the important possibility to be trained. Neural networks, especially because of the power to derive meaning from complicated or imprecise data, have been used in a variety of applications.

A feed-forward ANN consisting of three layers – one input layer, a hidden layer, and an output layer – is used to predict the location (Figure 3.29). This structure of an ANN is sometimes also referred to as a multi-layer perceptron [Rosenblatt, 1961; Fausett, 1994]. Each input, in our case image coordinates and robot pose information, arrives at an input node in the input layer of our ANN. As our network is fully connected this input is then provided to every node at the hidden layer. Each connection has a weight attached to it. In addition to all inputs, each hidden node also has a connection to a bias node (i.e. a node with constant value of 1). Similarly the output value of every hidden node is then provided to the output node. These connections are again weighted. Each output node is also connected to a bias node. In mathematical terms each node

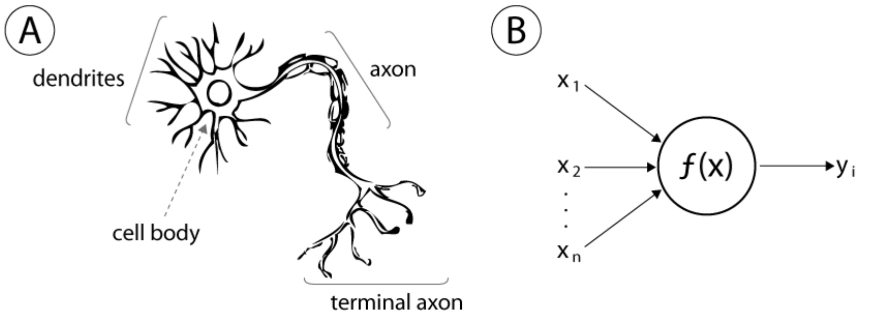


Figure 3.28. Model of a human brain cell (neuron, A) and an artificial neuron (B) used in Artificial Neural Networks (ANNs). (Image taken from Maltarollo et al. [2013])

in the hidden and output layer performs the following operation:

$$u = w_0 + \sum_{1}^{n} w_i x_i \tag{3.13}$$

where u is the output of the node, $n \in \mathbb{N}^+$ is the number of the neurons in the previous layer, w_i the weight related to the i-th connection and x_i the output value of the i-th node in the previous layer. w_0 is the weight related to the bias node. All the weights w and outputs x and u are in $\mathbb{R}$, most commonly within $[-1, 1]$.

The hidden and output layer neurons have a sigmoidal activation function of the following form to calculate their respective outputs:

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{3.14}$$

where u is the output calculated according to Eq. 3.13. We chose each ANN's output layer to be of only one single neuron. The output of this node is the

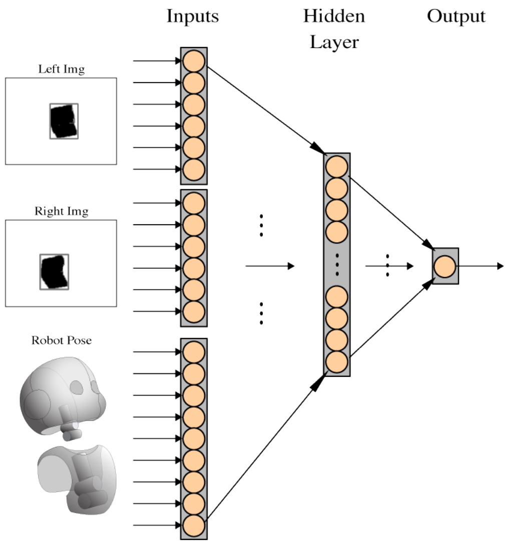


Figure 3.29. The model of the feed-forward ANN used for location prediction herein. The inputs are based on computer vision and the state of the robot, depicted on the left. These input neurons are connected to a hidden layer of neurons, which in turn are connected to the single output unit. This unit predicts the position of the object in one axis (of the 3D Cartesian space). (Note: Bias nodes and connections are omitted.)

Table 3.3. A typical entry from the dataset (in this case from reference point RP_i) and the limits used to scale the feature vector v for the neural network.

	ImageL					
	X	Y	X	Y	width	height
v	v_0	v_1	v_2	v_3	v_4	v_5
RP_i	479	411	429	401	99	25
max	640	480	640	480	640	480
min	0	0	0	0	0	0

	ImageR					
	X	Y	X	Y	width	height
v	v_6	v_7	v_8	v_9	v_{10}	v_{11}
RP_i	503	437	477	429	80	27
max	640	480	640	480	640	480
min	0	0	0	0	0	0

	Neck			**Eyes**			**Torso**		
	0	1	2	3	4	5	0	1	2
v	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}
RP_i	-10.0	0.0	0.0	-19.9	-19.9	0.0	-0.1	-9.9	10.1
max	25	25	10	20	15	5	20	20	50
min	-25	-25	-10	-20	-15	0	-20	-20	0

estimated position along one Cartesian space axis. Therefore to estimated the full 3D position three separate networks are required.

ANN Training The neural network approach requires a pre-processing step, in which the dataset (input vector) is scaled using the limits for each of the 21 possible inputs as given in Table 3.3 & 3.4 to calculate values in the range $[-1, +1]$. The limits are based on to the retrieved image size for the 6 values in each image, and the joint limits (i.e. range of motion) of the robot, for the encoder values. For training the network the (scaled) dataset was first randomly split into a training (80% of the data) and test set (20%). This is done to be able to check whether the results obtained via learning over-fitting on the training set or can generalize.

The ANNs were trained using the standard error back-propagation algorithm with the datasets collected. According to Russell and Norvig [2010] it is “the

Table 3.4. A typical entry from the dataset (again from reference point RP_i) showing the location vector p and its limits.

	Location		
	X	Y	Z
p	p_0	p_1	p_2
RP_i	0.42	0.27	-0.12
max	0.6	0.3	-0.2
min	0.1	-0.7	-1.2

most popular method for learning in multilayer networks [...] was first invented in 1969 by Bryson and Ho [Bryson and Ho, 1969], but was largely ignored until the mid-1980s.". Schmidhuber has surveyed the historical origins further since then and found that minimisation of error by gradient descent has been discussed for decades – e.g. continuous form of back-propagation was derived in the early 1960s [Kelley, 1960; Bryson, 1961; Dreyfus, 1962], Linnainmaa [1970] first described an explicit, efficient method for arbitrary, discrete networks – the community agrees that it was first applied to neural networks in the 1980s [Werbos, 1982]. This method is a generalization of the delta-rule and requires the activation function of the neurons to be differentiable and consists of two phases: (i) a forward propagation and (ii) a weight update phase. In phase one inputs from the training dataset are propagated through the network using the current weights for each connection. The output is then compared to the ground truth, the known result for the provided inputs. The error between the correct and predicted output is then used to calculate a gradient at each connection, which defines how the weights will be updated.

Genetic Programming Approach The second machine learning technique used to estimate the object's location is using an evolutionary search approach. GP is used to find expressions mapping the inputs to the outputs (3D coordinates).

'Eureqa' [Schmidt and Lipson, 2009], a freely available software for GP is used to find results. It produces compact, human readable expressions from datasets employing the above mentioned techniques. In previous work, this implementation has been shown to be particularly capable in using data generated from real world experiments and observations. For most parameters the default settings were used. These including a population of 64 individuals, a crossover rate of 0.5 and a mutation rate of 1.5% and the mean square error as fitness metric. The input values do not have to be scaled in this approach and can remain in the original form. As with the neural network regression, data was shuffled and then split into training and validation sets. The mathematical functions available to GP are listed in Table 3.5.

Collecting the Dataset

To learn about more than one state and hence get the ability to generalise, the dataset needs to contain points, where the robot is in various configurations looking at objects at various locations. A dataset of reference points (RPs) was collected on the real hardware to train the ANNs.

First we moved the *iCub* to a randomly selected pose, then the robot's state

Table 3.5. The mathematical functions available for the Genetic Programming (GP) method to select from.

add	subtract	multiply	devide
power	sqrt	exp	log
sin	sinh	cos	cosh
tan	tanh	asin	acos
atan2	min	max	abs

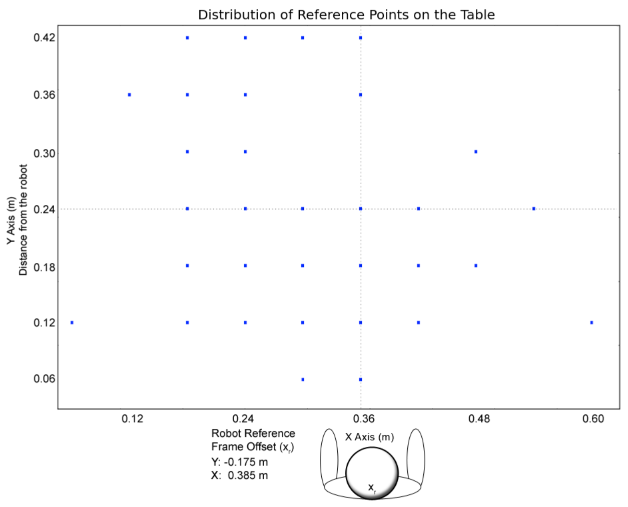


Figure 3.30. The distribution of the reference points which were collected on the table. The position the *iCub* with respect to the table is indicated and the robot's reference frame offsets to the table are specified.

and the location of the object in the camera frame, provided by the *icVision* framework (either from a hand-coded or learned filter), were recorded. The robot would then move about and another data point, with the object still at the same RP, but from a different view, was registered. Figure 3.30 shows the distribution of the RPs on the table. Some positions are missing data as the robot was unable to see the LED in both camera images in all the randomly selected poses for these points. At first these positions were hand measured and every few movements the object of interest was moved to another RP, while later on the process was automated by including another robot which would place the object at arbitrary 3D locations in the shared workspace.

3.6 Experiments and Results: Localization

Multiple experiments were run to showcase the proposed method for learning spatial perception on the *iCub*. Although the humanoid operates in a full 3D environment, in most of the scenarios we intend to work with, for now, objects will be placed on a table in front of the robot. Therefore, the problem, at first, is simplified, as we can assume a constant height for the estimation. This experiment was conducted, to show that the proposed learning approach is feasible. After sufficient performance was shown on the limited setup, a full 3D experiment was performed as well. In this case another robot provides the ground truth by placing the object at arbitrary locations in the shared workspace.

3.6.1 Learning to Localize Objects on a Table

The first experiments focusses on the feasibility of the approach, which does not require an explicit robot model nor a time-consuming stereo camera calibration procedure. A simplified setup, in which only 2D positions on the table need to be estimated, was chosen to start with. To learn the ability to generalise what it learns, the robot needs a dataset representing various configurations and object locations on the table. The first approach was to place a single object at different known positions on the table and collect data. To simplify the image processing, a bright, red LED was used. The LED was placed to mark the reference point, while the *iCub* moved into different poses. For each pose the joint angles and camera images were collected. After collecting data for a number of poses, the LED was moved to another position in a rather tedious and manual way. The detection and collection process was then repeated. The dataset contains 32 RPs on the table, with more than 30 robot poses per point. They lie in a region where the *iCub* is able to reach with its arms and were distributed in a grid with a spacing of 6 cm. One should not that although the system was trained with only one specific object, arbitrary objects can be localized by the deployed system.

To improve the data collection a chess board was used generating multiple RPs per robot pose of the *iCub*. The board was placed on the table (at a known location) in front the robot. Due to its size it was completely visible in both images only in a small subset of poses, and hence there were correspondence problems between the left and right images. So again this did not allow for a very large dataset to be collected (see statistics in Table 3.6).

As described above, two separate networks were trained to predict the coordinates on the X and Y axes independently. This approach was chosen as it allowed for faster learning (i.e. less generations needed to yield the results) and

Table 3.6. The statistics of the collected datasets for two different setups, one where the objects are localized on a table in front of the robot (2D), and the second one where the Katana robot is used to collect a full 3D dataset.

	Table (2D)	**Katana (3D)**
Data Points	965	1034
Reference Points (RPs)	32	45

the ability to run the learning in parallel. On average about 1700 epochs were needed per network for its prediction error to converge. After training the network produces estimates with an average accuracy of 0.8 cm, with lower separate errors on the axes (see Table 3.8).

The GP method, while converging faster than the neural network, performs at a lower average accuracy of 3.3 cm. Although this performance is worse than the ANN, it is still sufficiently accurate to allow for simple reaching and grasping tasks on the *iCub*. However, there are a number of advantages to be considered. The output is in a human-readable form, which can easily and quickly be transferred and tested on the robot (Table 3.7 shows the evolved equations). An interesting observation is that only one of the camera images is used (features v_0 and v_1). This allows to reduce the (complete) runtime of the estimation as only one images needs to be processed with object detection algorithms before the expression can be evaluated.

Table 3.8 compares the position prediction errors of the ANN and GP techniques. It shows that the neural network is performing better during learning, which can also be seen in Figure 3.31. It shows the difference between actual location and estimated position for each of the entries in the dataset. Both approaches perform similarly when generalising to unseen data (i.e. the test set). The ANN training necessitates a longer runtime, as the back-propagation algorithm repeats to update the neural network until the network performance is satisfactory. Both approaches were transferred to the *iCub* to perform real time

Table 3.7. The equations for the position estimation in the table case (2D) as generated by the GP learning approach.

$$\begin{aligned} x = \quad & 17.81 - 0.01906\ v_1 + 0.1527\ v_4 + 0.1378\ v_7 + 0.01108\ v_{10} \\ & -0.0296\ v_{11} - 0.1207\ v_{12} \\ \\ y = \quad & 1.1242240 + 0.1295921\ v_{10} + 0.1156011\ v_8 + 0.01695235\ v_0 \end{aligned}$$

distance estimation of objects and to allow for further verification of the off-line trained results. The object position in the images and joint encoder values were provided to both the trained neural network and the GP evolved formulae, to allow easy comparison of the position predictions.

The validation results were obtained using locations on the table and poses that were not in the original training nor test set. It was found that the GP out-performed (average error of 2.7 cm) the ANN (average error of 3.5 cm) on localization. Both techniques performed slightly worse than a fully calibrated *iCub*'s 'stereoVision' module (1.8 cm accuracy). The performance on the relative error, i.e. where the target object was moved by small increments away from a central point, was high for both implementation with the ANN yielding slightly

Table 3.8. Estimation accuracy on the table dataset for both machine learning techniques, ANN and GP.

	ANN	GP
Average Error 2D (cm)	**0.846**	**3.325**
Standard Deviation 2D (cm)	0.504	2.210
Average Error X (cm)	0.540	2.028
Standard Deviation X (cm)	0.445	1.760
Average Error Y (cm)	0.5433	2.210
Standard Deviation Y (cm)	0.4304	1.716

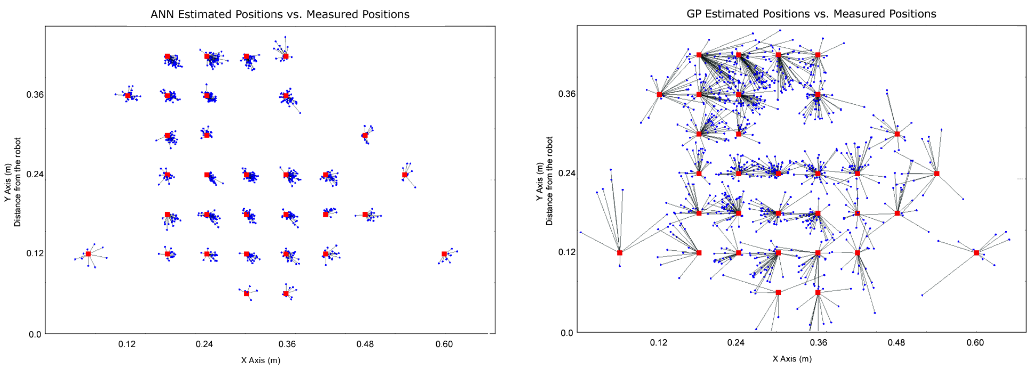


Figure 3.31. The estimated object position on the table (blue dots) vs. the measured object position (red blocks) for the machine learning approaches: on the left the result obtained from artificial neural networks (ANN), on the right the results using genetic programming (GP). To clarify which ground truth these estimates belong to a line connects them.

Table 3.9. The relative estimation errors (in cm) for both approaches, when estimating the position using fixed poses of the robot and object locations not in the training nor test set.

		ANN		GP		current *iCub*	
dX	dY	estX	estY	estX	estY	estX	estY
0	**+2**	0.10	**1.93**	0.51	**2.28**	0.0	**2.17**
0	**+1**	0.10	**0.78**	0.30	**0.91**	0.05	**1.0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**-1**	0.03	**1.14**	0.31	**1.35**	0.03	**1.07**
0	**-2**	0.11	**2.08**	0.61	**2.40**	0.03	**2.07**
+2	0	**1.70**	0.01	**1.93**	0.57	**2.01**	0.17
+1	0	**0.71**	0.10	**0.81**	0.34	**0.92**	0.11
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
-1	0	**0.99**	0.21	**1.12**	0.11	**1.17**	0.06
-2	0	**1.98**	0.30	**2.24**	0.34	**2.33**	0.1

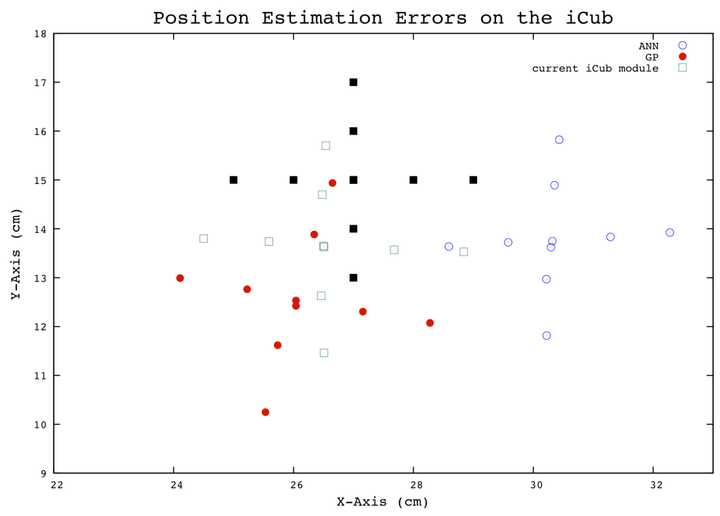


Figure 3.32. The relative localization errors on the real hardware. The ground truth is shown in black, the circles represent the learning approaches, ANN (empty circle) and GP (filled). Results from the *iCub* 'stereoVision' module is plotted in green.

better results (see Table 3.9 and Figure 3.32). The results of the current *iCub* localization module are added for comparison.

To test these approaches under moving conditions, we scripted the robot to move a given trajectory and recorded the position estimates for an object at a fixed location. The errors were tested for using only head/neck joints, for only using torso and for a combination of both all ranging 2-4cm. Higher errors were observed when moving faster, leading to the believe that mainly it is an issue with getting the images from both cameras synchronised. Test with a moving test object were also performed, the error though is harder to measure when both objects are moving (without the use of an external camera system, such as VICON), therefore only visual verification was possible.[5]

3.6.2 The Right Size of the Hidden Layer

To find the appropriate number of neurons for the hidden layer experiments with various hidden neurons were performed. The number of hidden neurons achieving the lowest estimation errors was then selected. Figure 3.33 shows that at around 10 neurons in the hidden layer a lower average prediction error can be achieved. This experiments operated on a reduced dataset with only one axis. The networks were trained on the dataset and the errors reported are on the 20% validation set. The errors were averaged over ten runs, i.e. with ten trained networks all having the same number of hidden neurons. For prediction along one axis the lowest average error was found with ten hidden neurons and is about *0.8* cm.

[5]A video showing object localization while both the *iCub* and the object are moving is available at `https://www.youtube.com/watch?v=xsz0Cj4A1eA`.

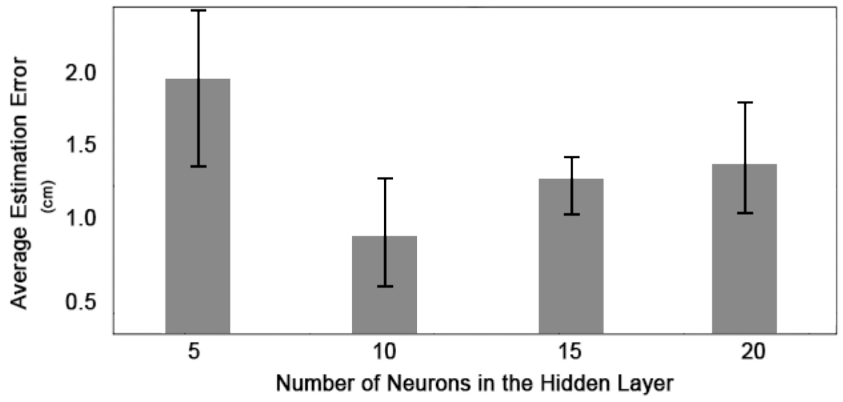


Figure 3.33. The estimation errors when training ANNs with varying number of hidden neurons. A reduced dataset with only one axis is used. The error bars show the minimal and maximal errors found over 10 trials.

3.6.3 Learning to Estimate 3D Object Positions

To automate and speed up the data collection and extend it beyond the 2D table, a high precision robotic arm was used. The 5 DOF Katana arm by Neuronics AG [2008], places an object at an arbitrary 3D position in the workspace it shares with the *iCub*. While the *iCub* moves about, it observes the object, and the robot 'learns' how to relate its pose and visual inputs to the object location in Cartesian coordinates (as before). To provide the humanoid with the information to learn from, the ground truth is provided by the precise inverse kinematics of the industrial robotic arm. That means, the output vector of our dataset p is taken from the Katana arm, which provides its end effector position in *mm* accuracy. To collect the dataset both robots move to randomly selected poses allowing for a random sampling of the configuration space. More than 1000 points were recorded in a fraction of the time it took collect a similar dataset by hand (see Table 3.6). As now two robots are sharing the same workspace, while acting independently, special precautions need to be taken to avoid collisions. To do this, *MoBeE*, an open-source software system, which provides real-time collision avoidance based on geometric calculations is employed (see Figure 3.36; a more detailed description can be found in Chapter 4.2). Note that for learning spatial perception, a very precise kinematic model of the *iCub* is not required. For example, one could imagine to replace the model with a simplification, that just uses the 9DOF of the iCub. In fact, even a box or single sphere, covering the whole robot is sufficient.

The feature vector $v \in \mathbb{R}^{21}$ is defined to contain the state of the robot as described by 9 joint encoder values and the observed position of an object in both camera images. The position is described by the centre of the detected object in the image plane (X and Y coordinate) and additionally by the bounding box of the detected object (upper-left corner, width and height). In the experiments though no clear evidence was found that this extra information helps to better predict the position. If more information about the object would be known, like e.g. its size, then this information will clearly have value for estimating the distance. A typical segmentation is shown in Figure 3.34, with the bottom showing the object and its bounding box, defining the values to be added for both the left and right camera image.

Three separate ANNs were trained to predict the position along single axes, each containing 10 hidden neurons. As mentioned earlier the neural network approach requires a pre-processing step, in which the dataset (input vector v) is scaled to values in the range $[-1, +1]$. The output of the neural network is in the same limited range and needs to be un-scaled. For training the network the (scaled) dataset was first randomly split into a training (80% of the data) and test

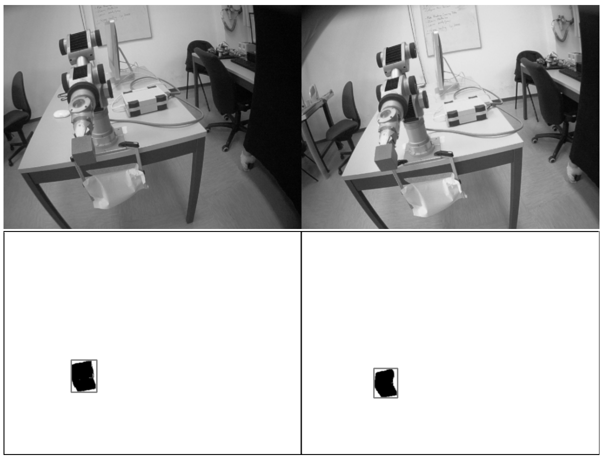


Figure 3.34. The upper row shows both camera images (left and right) perceived at the *iCub* (Note: while the camera images are shown in greyscale here the technique presented uses the raw RGB images provided by the robot). The lower one shows the results after processing. The object is segmented out and a bounding box is calculated (shown in bright grey).

set (20%). The independent training for each axis allows for parallel evaluation and reduces the complexity of each ANN. On average 1800 epochs were needed for the prediction error of the ANNs to converge, which is, as expected, very similar to the time needed to train for the table (2D) case. The back-propagation technique was again using a learning rate of 0.35 and a momentum of 0.1.

Figure 3.35 shows on the left three plots, one per axis, visualising the location and prediction error per sample in the dataset. The dataset was sorted by location in this figure, to highlight that multiple samples per location were collected. Overall the ANNs seem to quite nicely predict the position of the object. The average error on the dataset is 15.9 mm for the X-axis, 43.1 mm for the Y-axis and 37.3 mm for the Z-axis. This is also in the error range of current localization methods available for the *iCub*. The errors reported are the average of 10 learning runs. A few prediction outliers can be seen which might result from errors during the collection of data points (e.g. when a data point was collected while a collision was prevented and the robots were reset to a safe pose).

On the right, to better visualise where the prediction errors are largest, the difference between actual and predicted locations are again plotted for the XY and XZ 2D plane respectively (compare also with Figure 3.31). It can be seen that the dataset is not evenly distribute over the 3D space. This is due to the independent and random movements of the Katana and the *iCub* robot. A dataset

Table 3.10. Comparing the errors measured with the ANN machine learning method herein to the previous 2D case.

	ANN-2D	**ANN-3D**
Average Error X (mm)	5.40	15.9
Average Error Y (mm)	5.43	43.1
Average Error Z (mm)	-	37.3

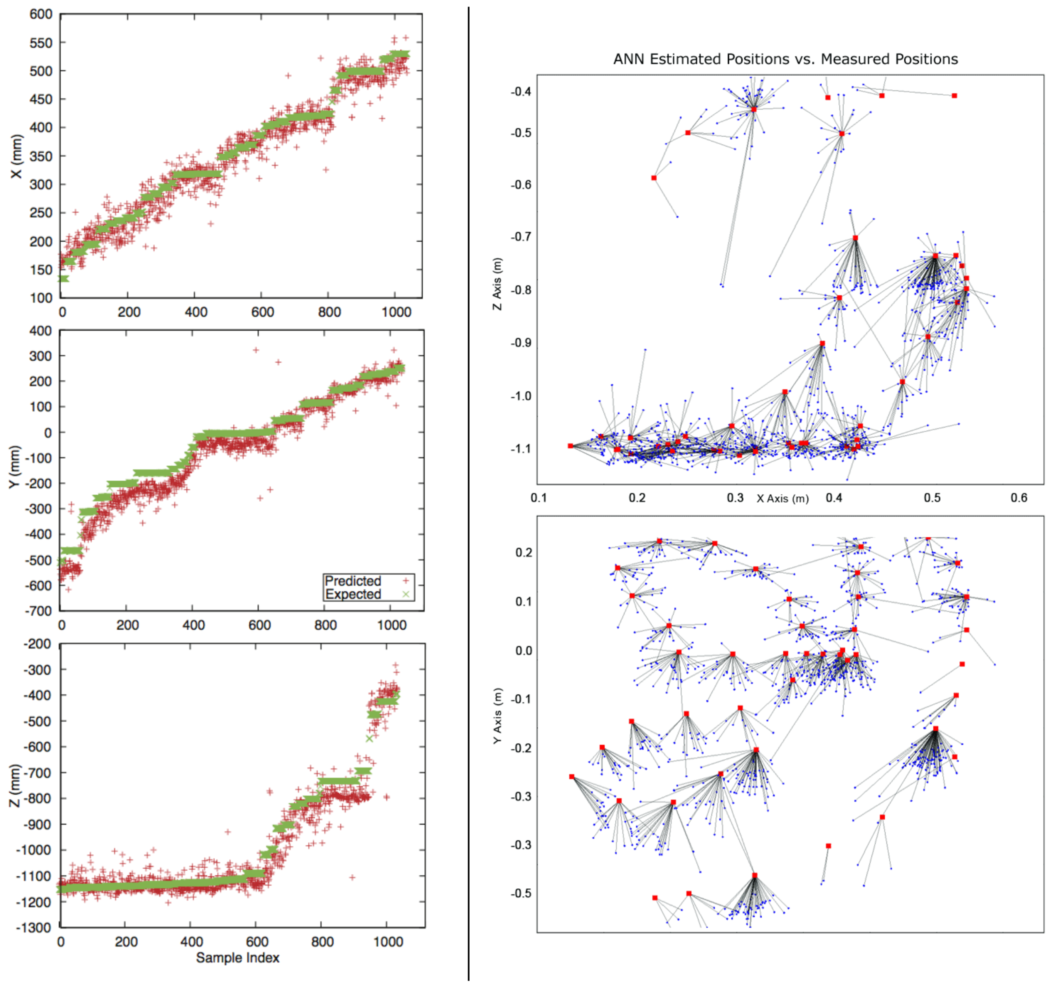


Figure 3.35. The predictions for the three separate ANNs (red) compared with the ground truth (green) in the full 3D perception case. On the left the estimated object positions (blue dots) are plotted against the ground truth (red blocks) *Note: The values are in the coordinate frame of the Katana robot.*

containing more points and distributed better throughout the workspace is expected to allow for improved prediction. In addition, Table 3.10 shows the average errors per axis and compares the results with the previous experiment. The ANNs were then again transferred onto the *iCub* to perform real time distance estimation of objects and to allow for further verification. The learned localization was tested by reaching for the red block held up by the Katana manipulator (seen in Figure 3.36). The estimated position from the ANNs was directly fed into the existing operational space controller [Pattacini, 2011] the provided operational space controller, enabling the *iCub* to reach for objects (Figure 3.37).

Differences in accuracy between the off-line training and real world validation were observed (as seen in Figure 3.32). There seems to be a systematic error of about 3-4 cm, when trying to estimate the position of objects in areas where training cases exist. In addition the pixel errors for the detection increase when objects are not in the centre of the frame, this is most likely due to the training set only containing pictures where the robot gazes upon the red block. In future work we intend to investigate this further. The results of the experiments presented in this section for learning object localization, based on this first 3D

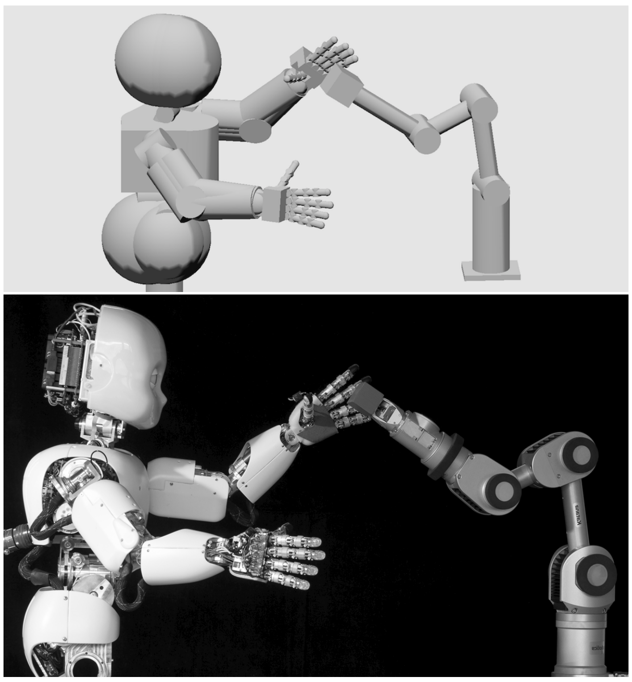
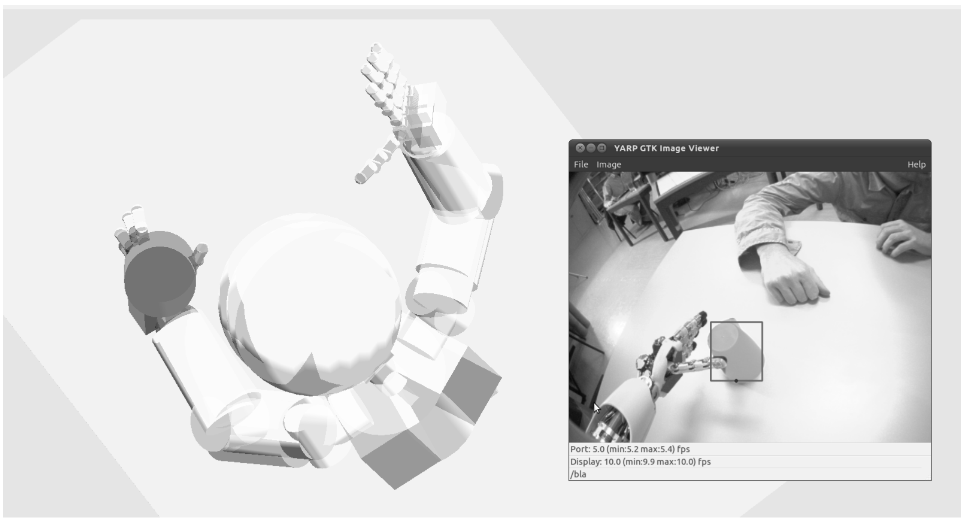


Figure 3.36. iCub and Katana arm models loaded into the *MoBeE* (top) to perform collision detection for both robots, while working in a shared workspace. The bottom picture shows the real scene.

dataset, show that the method can be scaled to perform full 3D estimation.

The results show that the *iCub* can learn simpler ways to perceive the location of objects than the human engineered methods. Both approaches provide simple and fast methods that can be used in real time on the robot. A future research topic will be the investigation of how this framework can be used to on-line learn the spatial perception, for example in ego-sphere and arm reach. This would enable the autonomous calibration of the full hand-eye system, an important issue in many areas [Ambrose et al., 2012; Kragic and Vincze, 2009]. As the learnt models are 'light weight' they could easily be incorporated into embedded systems and other robotic platforms.



```
Port: 5.0 (min:5.2 max:5.4) fps
Display: 10.0 (min:9.9 max:10.0) fps
/bla
```

Figure 3.37. The object's position is estimated and the world model of the *iCub* is updated. The evolved formula is used to calculate the position on the table based on the input images and the current encoder positions. *Note: The cup is placed directly under the arm, due to the parameters of the camera and the different perspective this is hard to see.*

Recap

To recap, in this chapter *icVision* and *CGP-IP*, two frameworks and a novel approach to provide our robot with flexible object detection capabilities is described. It furthermore details a technique to learn spatial perception without the need of exhaustive calibration or depth sensing.

The main contributions are:

- *icVision*: I built a framework to enable a wide variety of computer vision tasks on the *iCub*. It includes a variety of abstract modules, which can easily be swapped, even on the fly. It enables to use a variety of machine learning approaches directly on the hardware.

- *CGP-IP*: based on previous work on Cartesian Genetic Programming (CGP) this framework was created by myself to allow for the learning of object detectors from very small datasets, on the real hardware and based on domain knowledge provided by OpenCV. Results presented show that using between 5 and 20 images CGP-IP can learn visual object detectors. Due to the small training set, it is quite easy to learn the filters and deploy them on a robotic platform, in connection with *icVision*.

- *Robots training each other*: by using a precise robot to teach the *iCub* about spatial perception the task of creating training sets for experiments using standard off-the-shelf machine learning techniques was automated. To my knowledge this is the first time that something like this has been done. While the proof-of-concept for learning spatial perception on the humanoid requires another robot, the approach is flexible enough to allow for online, life-long adaptation and learning on the *iCub*.

Chapter 4

Motion Generation and Learning

Even with great advances in sensing technologies, the robot still needs to control its motion to perform manipulation tasks. Directly programming robots by writing code can be tedious, error prone, and inaccessible to non-experts. While this explicit model-based control is still the primary approach – which works very well if the world's state is known and predictable – it has limitations when it comes to uncertainties. Robots, used in a human environment, cannot expect to estimate the state of the surroundings with certainty. It seems almost inevitable that learning will play an important role in robot manipulation. Through learning, robots may be able to reduce the burden of programming and in addition continue to adapt even after being deployed, creating some level of autonomy. As our platform is a highly complex system, learning instead of direct programming are deemed to be the most suitable. When attempting to create behaviours on a complex robot like the *iCub*, state-of-the-art machine learning and control theories can be tested and shortcomings can be discovered and addressed. For example, Hart et al. [2006] showed that a developmental approach can be used for a robot to learn to reach and grasp.

The issue of uncertainty in real-world applications has been addressed with robust closed-loop controllers using sensory feedback. For example, Piatt et al. [2006] and the Robonaut group at NASA/JSC and Hart et al. [2006] at UMass Amherst have explored ways to learn and compose real-time, closed-loop controllers in order to flexibly perform a variety of autonomous manipulation tasks in a robust manner. Edsinger and Kemp [2006] at MIT often use hand-coded behaviour-based controllers that specify tasks in terms of visual or other sensory feedback driven control. Another important issue with learning in real robotic systems though is the need to provide some safeguard in case of a 'bad' command is sent during a trial-and-error phase.

4.1 Background: Robot Motion Creation & Control

Only after the robot knows *which* objects are in the environment and *where* they are located can it start to interact with them. The Path Planning Problem is an important problem in robot motion research. It denotes the issue to find motions that pursue a defined goal, e.g. reaching, while deliberately avoiding obstacles and other constraints. Solving this problem is critical to deploying complex robots in unstructured environments. An overview of interesting approaches to motion planning can be found in the textbook "Planning Algorithms" [LaValle, 2006]. To solve the path planning problem is generally expensive, therefore robots controlled this way are typically very deliberate and slow. This is often perceived during robotics demos (e.g. at recent DARPA Robotics Challenge Trials), where one can see them 'think' first, often for quite a bit, before a motion is executed, as e.g. seen for manipulators with many DOF, even with fast replanning [Baginski, 1999].

A more reactive approach, instead of the 'think first, act later' paradigm, emerged in the 1980s. A variety of approaches have been applied to quickly generate control commands, without searching the robot's configuration space [Khatib, 1986; Brooks, 1991; Schoner and Dose, 1992]. All these use local information from the workspace, and transform it into motor commands according to some heuristics. It is therefore not surprising that these approaches excel at fast, reactive obstacle avoidance while they have trouble with global planning tasks. Because of this such approaches have become popular in the context of safety and human-robot interaction [De Santis et al., 2007; Dietrich et al., 2011].

Generally the approach is to start with low complexity control, such as, controlling the robot's gaze and slowly include more DOF until the full arms are controlled for reaching. This has been extensively investigated previously, also on the *iCub*, e.g. using a combination of open and closed loop control [Natale et al., 2007; Nori et al., 2007] and motor babbling [Caligiore et al., 2008]. To advance towards the goal of general object manipulation, where a robot can autonomously manipulate any given object within its workspace, it would ideally encode and reuse knowledge in terms of task features that are invariant to the object. Confronted with a novel instance of a specific task the robot needs to establish appropriate correspondences between objects and actions in its repertoire and their counterparts in the current task.

Once a possible action or motion is found to reach for a certain object, the actual manipulation is yet another problem to solve. This is not a trivial thing, even for humans. For example, research shows that even a prototypical precision grasp task is not matured until the age of 8-10 years [Forssberg et al., 1991].

Moreover, complexity, as seen by the number of neurons comprising the control of the arm and hand, is staggeringly high. Recent results though highlight the progress and advanced state of research in robotic grasping [Carbone, 2013].

4.2 *MoBeE* and Collision Avoidance

Over the last years "“Modular Behavioural Environment” or *MoBeE* was developed at IDSIA under the lead of fellow PhD student Mikhail Frank [2014]. It is a novel framework for robots, from simple robotics arms to complex humanoids, which integrates elements from planning and control, facilitating the synthesis of autonomous, adaptive behaviours.

The main contribution of MoBeE for the *iCub* is to prevent collisions, both with itself and with objects in its surroundings. As the robot does not contain a tactile skin, which would provide feedback of (unwanted) world interactions, a software module was developed that provides a similar ‘sensory’ system. At the core of MoBeE is “a parsimonious, egocentric, kinematic model (figure 4.1), which does collision detection while driven by the state of the actual hardware” [Frank, 2014]. *MoBeE* is intended to enable machine learning research on real robotic systems. The system allows to send random signals to the robot without the robot getting ‘hurt’. This is especially useful for techniques that try to learn to interact with the world and are based on a trial-and-error paradigm, such as, for example, reinforcement learning.

MoBeE has a similar design philosophy to YARP and also aims for transparency and modularity in its subsystems. This allowed to extend the behaviour to the multi-robot setup used for the experiment in the Section 3.6.3. It has an approximate complexity of the computation of $O(n^2m)$, where n is the number of objects in the robot model and m the number of objects in the environment. For this detection the Software Library for Interference Detection (SOLID) [van den Bergen, 2004] is used. It provides highly optimized code for geometric computations (supporting primitives, Minkowski sums, and polyhedra).

MoBeE aggregates contributions from three separately running parts: the Sensor, Agent, and Controller. The *Sensor* provides information about the state of the world to *MoBeE* – see the previous chapter for a detailed description of how *icVision* provides object information. The *Agent* communicates planned motions to *MoBeE* periodically. The *Controller* plays ‘man-in-the-middle’ between the *Agent* and the real hardware, suppressing the Agent’s commands when necessary, i.e. when an impeding (unwanted) collision is detected. At first developed as a simple switching control this system over the last years evolved into a more complicated but also more adaptive approach of using virtual force fields (as suggested first by Borenstein and Koren [1989]) to avoid colliding. *MoBeE*'s creation started out of three primary ideas and requirements for our research. These in turn laid the foundation for its first components [Frank et al., 2012], namely:

1. A kinematic model of the robot and workspace (World Model).

2. A port filter that allows it to act as a proxy between an arbitrary control module and the robot.

3. A collision avoidance/response behaviour.

The system works as follows: A controller, connects to the proxy created at runtime (instead of the direct YARP interfaces of the robot) and can then start controlling the robot. *MoBeE* uses the state messages arriving from the real hardware to update the kinematic model and performs real-time collision detection computations based on these. When an impeding collision is detected the robot is stopped and the proxy is closed. Then the defined reflexive collision behaviour is triggered and the controller module is notified. Once the the system is recovered from the dangerous configuration the reflex stops and the controller can continue to use the robot, this way successfully preventing impeding collisions.

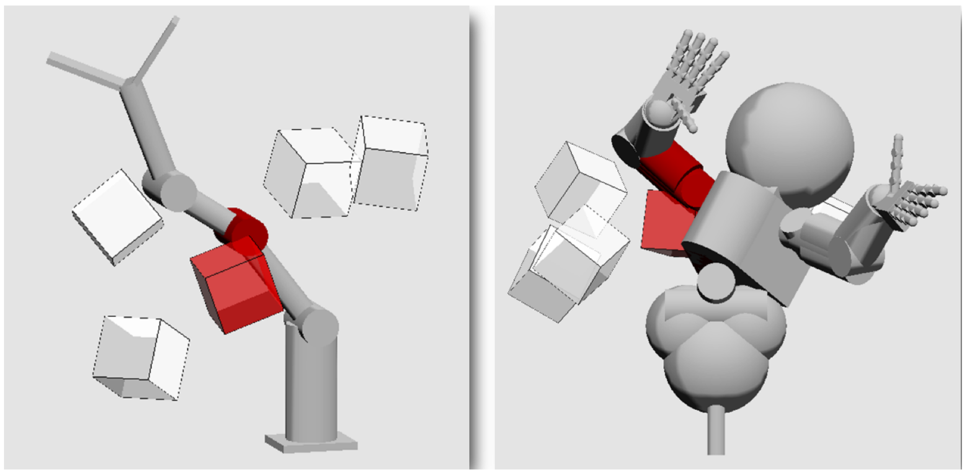


Figure 4.1. MoBeE has at its core a kinematic model of the robot. On the left the model for the Katana is shown, on the right the one for the *iCub*. It is driven by the encoder information from the real hardware. The geometries are constantly checked for intersection, in which case the robot's current motion is stopped because of impeding (self)collision.

An improved approach is currently used where each object in the world model is having a virtual force field surrounding it. These forces are accumulated and used to 'push' the robot away from potentially dangerous interactions. For this the forward kinematics together with the Jacobian are used to calculate the control signals for each individual joint. The details can again be found in Frank [2014]. In addition this approach allows us to send fictional forces to selected makers of the robot in the world model – such as the left or right end effector – to create some more human-like motions with just a single force control of the robot.

At the IM-CLeVeR demo a low-level reactive controller was coupled with a RL framework to find 'interesting' states through interaction with the environment. We showed that the robot can learn models to represent large regions of the *iCub*'s configuration space [Frank et al., 2014].

4.2.1 Movement in a Shared Workspace

Multiple robots sharing the same workspace, may it be in cooperation or competition, have been investigated previously. The research is usually focused on mobile robotics for exploration scenarios [Burgard et al., 2005], as area coverage is seen as one of the canonical problems. Cooperating mobile robots have been researched for diverse applications, such as, outdoor and indoor surveillance operations [Moors et al., 2005], cleaning [Forlizzi and DiSalvo, 2006], and space exploration [Leitner, 2009].

Surprisingly little work has been done on multi-robot setups with humanoids. The main focus seems to be on shared workspaces with humans. This field of human robot interaction (HRI) is growing, e.g. Dautenhahn and Saunders [2011] nicely describes the current research trends. In the last years the work on humanoids, while controlling both arms to, e.g. perform bimanual grasping, has become more prominent. Gharbi et al. [2009] presented a roadmap approach for path planning using both arms of the DLR Justin robot [Ott et al., 2006]. It allows to plan object manipulation motions, based on decomposing the system into kinematically independent parts, without the two arms colliding or interfering with each other. Only recently cooperation has also become of interest for humanoid robots and robotic helpers geared towards the use at home. A dual-robot setup was shown by TUM using their PR2 *James* and humanoid robot *Rosie* (based on Justin) making pancakes [Beetz et al., 2011]. In their setup the workspace of the two robots is overlapping only for a very little part of the demonstration space and therefore collision avoidance between robots is mainly ignored.

We at IDSIA are interested in robotic applications with drastically changing, unpredictable, unstructured environments. To do this we want to have multiple agents, either robots or humans, sharing a common workspace, in which collision avoidance becomes critical. The aim is for the *iCub* to learn to interact in such an environment, act out pre-defined tasks, and adapt to changes in the environment, introduced by the Katana manipulator arm. One such setup, in which two robots are facing and teaching each other, has been briefly mentioned in the previous chapter (Figure 3.36).

An obvious challenge in the multi-robot scenario is to prevent the robots from colliding with each other, or indeed, with themselves and other parts of the surroundings. Collisions are likely to lead to damage to either the robot or the environment, leading to time-consuming maintenance. One approach to tackle this problem of multiple robots interfering and colliding while in the same workspace is to plan ahead of time. Algorithms that take this approach are generally called 'Path Planning' or 'Motion Planning' algorithms, as they plan and validate feasible motions, which can later be passed to the robot as reference trajectories. For multi-robot settings a good and thorough introduction to collision avoidance and detection problems has been published by Gill and Zomaya [1998]. Akella and Hutchinson [2002] investigated collision-free trajectory coordination, in industrial applications, where the trajectories of the (homogenous) robots were predefined and known and coordinating these was the main issue. Multiple robot arms and generation of non-colliding paths while, e.g. passing it from one arm to another were explored by Koga and Latombe [1994].

Alternatively the robot can react to impending collisions as they are predicted (as in the case of some of the above listed works). *MoBeE* allows to build such a framework also for a multi-robot configuration. As a YARP module, it can easily be used with any robot, as long as YARP drivers have been implemented. The *iCub* drivers are included in the standard version but for the Katana, they needed to be added. The basic driver was developed previously [Kaufmann, 2010] and was adapted to work with the current YARP version. Due to the open-source design this could be done rather modular, building on various wrappers for the Katana API [Neuronics AG, 2008]. For the spatial learning setup a few parts needed to be created and/or adapted, namely:

- add a kinematic model of the Katana arm
- allow to load two models side by side and
- add a collision response for the second robot

The robot model can be specified via an XML configuration file, using the "Zero Position Displacement Notation" [Gupta, 1986], which is significantly less complex and more intuitive than the popular convention by Denavit and Hartenberg [1955]. Due to the lower complexity of the Katana arm the XML file is short and easy to read. The reflex behaviour was a bit tricker to modify, mainly because the parameters used for the two robots had to be tuned to be synchronous.

Allowing our *iCub* to interact with other robots opens up a range of potential research avenues. In the previous chapter it was shown how the *iCub* can learn spatial perception from another teaching robot. Emphasise was put on learning positions in the (rather limited) area the robot is able to reach. There the location is precise enough to allow the *iCub* to grasp and manipulate objects. *MoBeE* was a key component in the success of this work. Without it, the *iCub* would only be able to learn about objects that were far away, or in a constrained region – such as on the surface of a table [Leitner et al., 2012e].

4.3 Action Repertoire: Task-relevant Roadmaps

The kinematic model in MoBeE was extended into pose/path planning application under the lead of Marijn Stollenga. The system is combining inverse kinematics, i.e. find a robot pose $q_{goal} \in C$, where C describes the robot's configuration space, that satisfies some operational space constraints, with planning, i.e. find a feasible configuration-space trajectory, $Q \subset C$, which is the curve from the current pose, $q_{initial}$ and the target pose q_{goal}.

At IDSIA we pursue this with Natural Gradient Inverse Kinematics (NGIK). An approach developed that benefits from a recent and powerful black-box optimization algorithm, called Natural Evolution Strategies (NES) [Glasmachers et al., 2010]. In line with ideas from current planning literature [Berenson et al., 2009, 2011], we too define *task spaces*, which are covered iteratively by applying NGIK. The resulting sample set, comprising a family of task-related poses, is interpolated to yield a Task-relevant Road Map (TRM) [Stollenga et al., 2013]. To build TRMs, NGIK iteratively optimizes postures covering task-spaces expressed by arbitrary task-functions, subject to constraints expressed by arbitrary cost-functions, transparently dealing with both hard and soft constraints. TRMs are grown to maximally cover the task-space while minimising costs. Unlike Jacobian methods, our algorithm does not rely on calculation of gradients, making application of the algorithm much simpler. In our experiments NGIK outperforms recent related sampling algorithms, winning also a 'Shakey'.[1]

[1]The demo won the AAAI Best Student Video Award: `http://youtu.be/N6x2e1Zf_yg`

During the recent IM-CleVeR final demo we showed that *MoBeE* also enables the robot to ‘learn’ to plan motions using reinforcement learning (RL) and ‘artificial curiosity’. While these fields usually are employed in toy-scenarios, e.g. navigation in a simple maze, in our case we embody it into the real high-DOF robotic hardware. But searching the configuration space of a complex, high DOF robot, such as a humanoid is a computationally expensive procedure. Our framework tackles complex IK and planning at the same time by combining NGIK with an iterative roadmap construction strategy. It finds a family of postures that are optimized under constraints defined by arbitrary cost-functions, and at the same time maximally covers a user-defined task-space. Connecting these postures creates a rather dense, traversable graph, which we call roadmaps. In other words, the task-relevant constraints are built directly into the TRM, and motion planning is reduced to graph search. This allows us to build TRMs that can perform useful tasks in the 41-dimensional configuration space of the upper body of the *iCub* humanoid. Additionally these maps can be stored to create an action repertoire that can be recalled when a certain tasks needs to be executed. Details about NGIK and TRM can be found in Frank [2014] and Stollenga et al. [2013]. Figures 4.3-4.5 show time-lapse snapshots of motions, planned within TRMs. They provide an idea of what kind of motions can be generated.

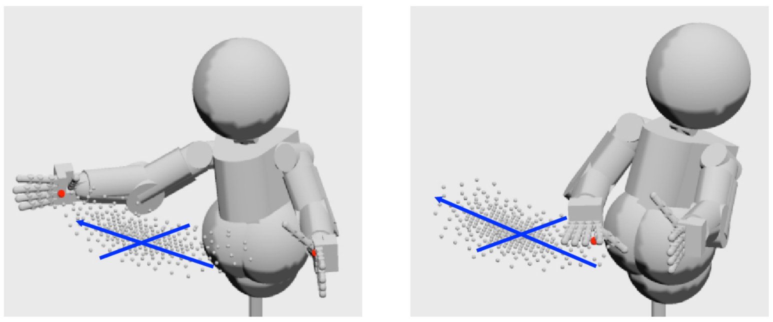


Figure 4.2. A Task-relevant Road Map for the hand is visualized in the robot’s workspace. Each dot indicates the position of the right hand in one of the found poses. The position is varied while the hand orientation is constrained: on the left the palm faces side-ways, on the right the palm is restricted to be facing downwards.

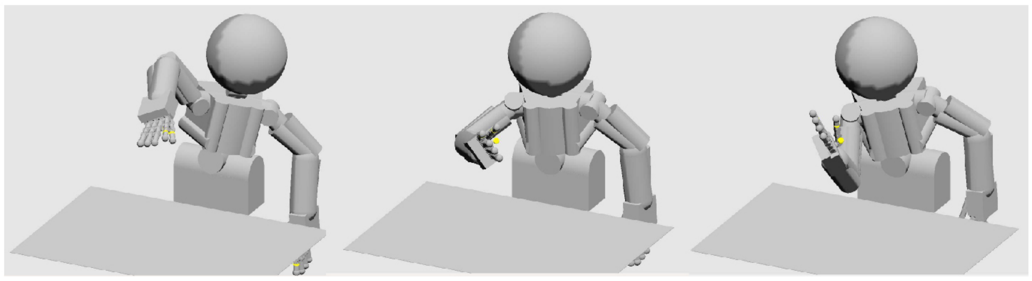


Figure 4.3. 'Curiously inspect something small': the 3D position of the hand is constrained, and the task space is its angle with respect to the gaze direction. The resulting map rotates the hand (and any grasped object) in front of the eyes.

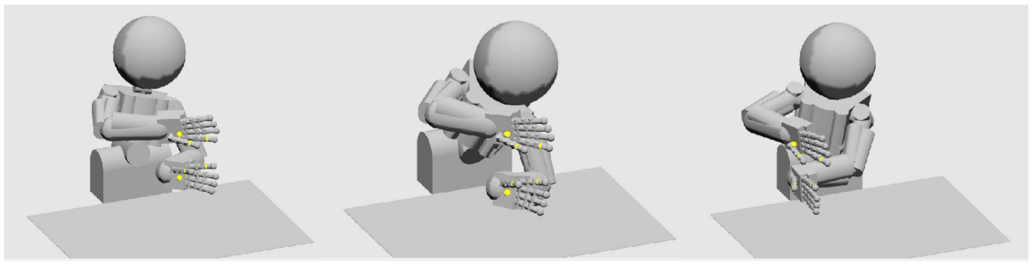


Figure 4.4. 'Push over': the height and relative position/orientation of the hands is constrained, and the 1D task space is to move them left/right. The resulting map allows the robot to push a large object to the side.

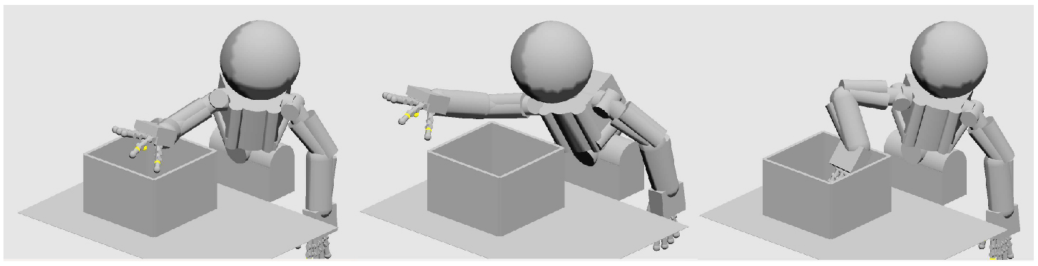


Figure 4.5. Reach into: the robot is constrained not to collide with the box, and the task space is the 3D workspace for the hand. The resulting map allows the robot to pick/place from/into the box.

4.4 Action Repertoire: Grasping

Robotic grasping is an active and large research area in robotics. One of the problems with grasping is that robot grasping is generally very sensitive to how accurate is the pose estimation of the object to manipulate. Even a small error in the estimated pose may cause the planned grasp to fail. Several methods for robust grasp planning exploit the object geometry or tactile sensor feedback. However, object pose range estimation introduces specific uncertainties that can also be exploited to choose more robust grasps.

Grasp planning methods that explicitly considers the uncertainties on the visually-estimated object pose for known shapes have recently been investigated. Very recently a grasp planner has been proposed using a particle filter to estimate the object probability distribution to select robust grasps observed as a–possibly sparse–point cloud. The points of the cloud are usually not uniformly distributed over the surface of the object, and are a function of the viewpoint. Additionally textures over the object surface can lead to irregularities and more uncertainties, when using stereo-vision algorithms based on robust feature-point matching. Consequently the pose estimation may be more accurate in some directions and contain unavoidable ambiguities. Such an approach was recently tested on the *iCub* and its stereo cameras [Saut et al., 2014].

LEOGrasper is our light-weight, easy-to-use, one-shot grasping system for the *iCub*. It has been used extensively at IDSIA, not just during experiments but also especially for the various video shots (Figure 4.6).[2] While it is not a planner for grasping, it can be configured to perform a variety of grasps. Each grasp requires to close the fingers in a coordinated fashion to be successful. These different

[2]Source code available at: `https://github.com/Juxi/iCub/`

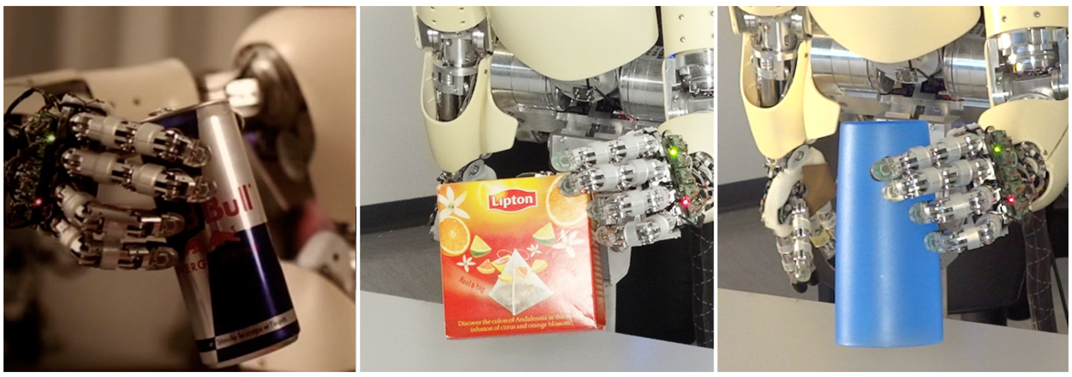


Figure 4.6. Grasping a variety of objects, such as, tin cans, tea boxes and cups.

coordinated motions of all the fingers in unison are pre-programmed, for actions such as a power grasp, pinch grasp or even just a pointing gesture. The *iCub* incorporates touch sensors on the fingertips. Due to the high noise of these, they are not used to provide information about the contact with an object. Instead the errors reported by the PID controllers, for the motors of each digit, are employed to indicate that the motors should stop.

4.5 Learning Motion from Humans and Bio-Signals

Further input about motion generation was provided by two EU projects, *WAY*[3] and *STIFF*.[4] Both take a closer look at how human motion is generated and how we control our actions. The goal is to transfer this knowledge into robotic systems, e.g. to allow better control and/or tele-operation.

The goal of the *STIFF* project was to equip a highly biomimetic robot hand-arm system with the agility, robustness and versatility that are the hallmarks of the human motor system, by understanding and mimicking the variable stiffness paradigms that are so effectively employed by the human central nervous system. A key component of this study will be the anatomically accurate musculoskeletal modelling of the human arm and hand. IDSIA's role was to investigate ways of extracting the 'policy' a human is following, from information collected about the motion (e.g. trajectory, velocity, ...).

The *WAY* project addresses the scientific problem of recovery of hand function after amputation or neurological disabilities like spinal cord injury,brachial plexus injury, and stroke. It introduces several conceptual novelties which explicitly take into account and overcome the limited band-width in actual Brain-Neural Communication Interfaces (BNCI). IDSIA's contribution was to investigate how bio-signals, such as EMG or EEG, can be classified and further be used to control robotic systems.

Extracting Human Cost Functions from Observations

The main aim of the STIFF project was to answer the question: *How can we transfer principles of human arm movement to a robot?*

The first step was to look at methods for extraction of these principles from measured (human) arm trajectories. Our partners in this project were aiming to model the full 139 human arm muscles (TU Delft) on one side and building a 52

[3]Project: *Wearable interfaces for hAnd function recoverY* (WAY) FP7 contract #288551
[4]Project: *Enhancing biomorphic agility through variable stiffness* (STIFF) #231576

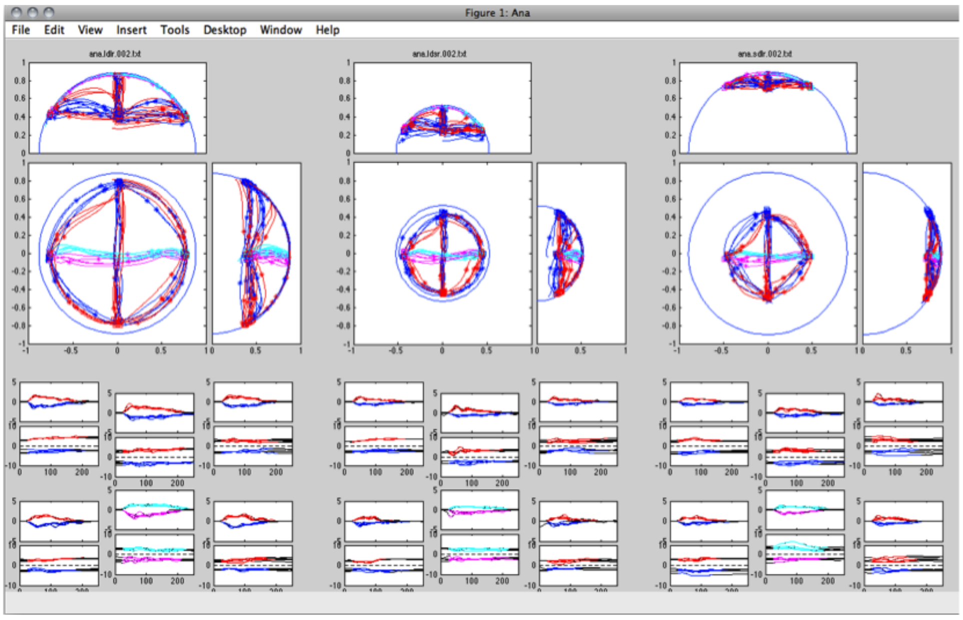


Figure 4.7. This visualization tool provided by UPD is showing the trials performed by one individual during the test session. The figure can roughly be split into three columns separating the runs performed in three different environments. The upper half shows the trajectories (XY, YZ, and XZ), whereas the lower half shows the velocities and forces vs. time (the active parts are coloured).)

DOF robot arm using antagonistic muscles on the other side (German Aerospace Agency, DLR). Instead of directly mapping the muscles to motors, a meaningful high-level principle from the human motion data is extracted, which in turn can be implemented on the DLR hand-arm system (HASy).[5]

The experiments were carried out at Joe McIntyre's lab at UPD in Paris. In this task a human subject is moving a robotic arm, which records the forces and movements, and is able to introduce systematic perturbation forces in catch trials. The subjects were tasked to steer the end effector from one point to another along a curved surface. In addition they were told to put a constant force onto the surface while moving to the end point.

The data received from UPD contains the test runs from 11 subjects with roughly 50 runs per person performed in three different scenarios: large disc - large radius, large disc - small radius, small disc - large radius. In a visual analysis of the data the team from UPD found some interesting trajectories: throughout

[5]http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-5486/8995_read-16713/

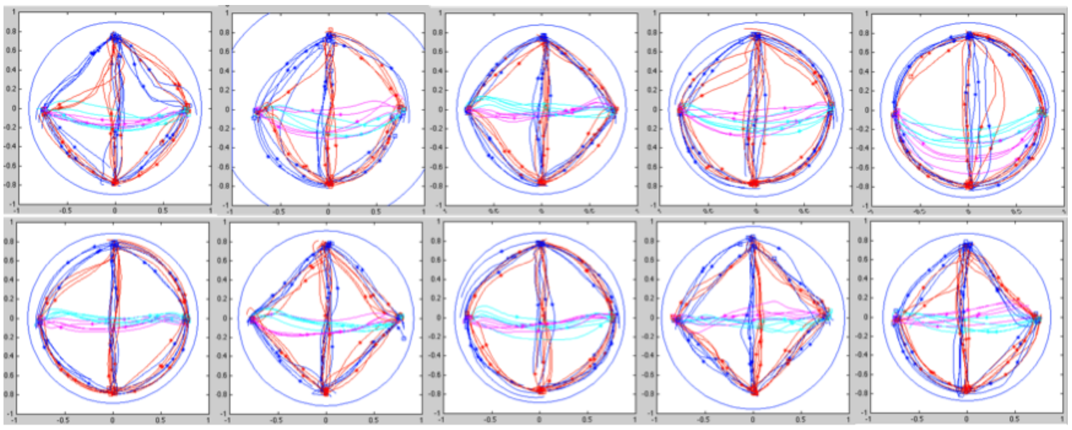


Figure 4.8. Showing the trials performed by all individuals in one environment. The interesting behaviour can be seen in the upper left corner of every plot.

the 11 test subjects (and all three setups) the trajectory for the upward and downward motion differ in the upper left section of the sphere. Figure 4.8 shows all trajectories for 10 subjects on the same setup (large disc - small radius). It can also be seen that the subjects are using individual strategies to reach the goal position, which can be categorized roughly into two groups: some subjects follow an arc (in the frontal plane) to reach the points on the circle, whereas some others are trying to reach the point via a line. But note that even within the trials of one subject switching between arcs and lines could be observed (for example, the last plot on the bottom row).

The aim was to generate various controllers for a simulated robot arm that would be able to generate similar behaviour to that observed in the human trials. Also to compare the human performance to an 'optimal' model, in which we find controllers that perform the same task, but while attempting to the time to completion and the effort required. Using evolutionary algorithms, controllers that generate signals to move the end-point in addition to scaling factors for the stiffness are created. For these scaling factors the underlying assumption that there exists a constant equilibrium point position is used. The controllers themselves are simply a function of time.

Here, where the task is to operate in 3D, two stiffness ellipses are used, each with a different scaling factor to represent an anisotropic 3D ellipsoid. The two stiffness ellipses can be considered as ellipses in the XY and XZ directions. With the ellipses rotated to point towards the shoulder (which is at the origin of the reference frame).

Search Technique Similar search algorithms to the ones used for image processing earlier on were applied to find the force and stiffness profiles (namely GP and CGP). The goal was to find a controller $C(t)$, for which, in the simulations here, five different control signals are needed. Three signals are used to modify the forces applied to the endpoint F_X, F_Y and F_Z and two to scale the stiffness ellipses λ_h, λ_v. The outputs from the controller are a function of time, leading to a trajectory performed by the endpoint of the robotic arm.

The genotype of each individual contains lists of parameters for Gaussian functions, where one list is needed per dimension of control. Each list contains a variable number of these parameters, which are then evolved throughout the run. The five dimensions of the controller $C(t)$ are each defined by the sum of Gaussian functions,

$$u(t) = \sum_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-\mu_i)^2/2\sigma_i^2} \tag{4.1}$$

The number of functions per list, represented by n in the previous equation, can vary over time as regulated by the evolution. The controller's complexity increases with the number of functions (Gaussians) that are summed together, hence the more functions used the more complicated the controller can become. To prevent the controllers becoming unnecessarily complex, mutation removes Gaussians with a higher probability than it adds them.

More formally the evolutionary search is trying to find an individual consisting of 5 lists of genes, $I = \{L_X, L_Y, L_Z, L_h, Lv\}$, which represent the parameters for the controller, where $L_i = \{(g_{0,\sigma}, g_{0,\mu}), ..., (g_{n,\sigma}, g_{n,\mu})\}$.

A conventional evolutionary strategy was used as algorithm. In this approach, the best individual from a small population P is used as a parent to generate four offspring. Each of these candidate individuals is tested, and the best one replaces the parent. If the best offspring has the same fitness as the parent, then it becomes the parent.

Fitness function A fitness function, sometimes also referred to as cost function, is used to evaluate a candidate solution's performance. Here we use the following definition:

$$
\begin{aligned}
f = & \sum_{t=0}^{T} |F_n(t) - F_h(t)| * w_1 + \\
& \sum_{t=0}^{T} |v(t) - v_0| * w_2 + \\
& \sum_{t=0}^{T} |p(t) - p_h(t)| * w_3 + \\
& \sum_{t=0}^{T} |p(t) - R| * w_4
\end{aligned}
$$

Where the terms are defined the following:

- The normal force error (first term) is the sum, over each simulation time step, of the difference between the average of the human applied force, normal to the surface, and the force control outputs of the evolved controller, also normal to the surface.

- The equilibrium position error (second term) is the sum of the distance between the virtual position, assuming a fixed depth, and the calculated virtual position using the scaled stiffness ellipse from the controller.

- The position error (third term) is the sum of the distance between the position recorded in the human trials and the controller's position.

- The surface error (last term) is the sum of the distance of the controller's position to the spherical surface (radius). (This not-touching-the-surface error is penalized comparatively large to the other errors.)

The changing of the weights $w_{1...4}$ adapts how much emphasis the fitness function places on those error terms. As with the previous deliverables in the *STIFF* project, the controllers are time dependent. Some pre-processing was needed to get the human trial data into a form suitable to work with for the fitness function. The trial runs were trimmed to only include data from when the human test-subject started and stopped moving. The data per trajectory, per individual were averaged together by firstly re-sampling the data into a consistent number of data points, and then averaging over the data of multiple trials.

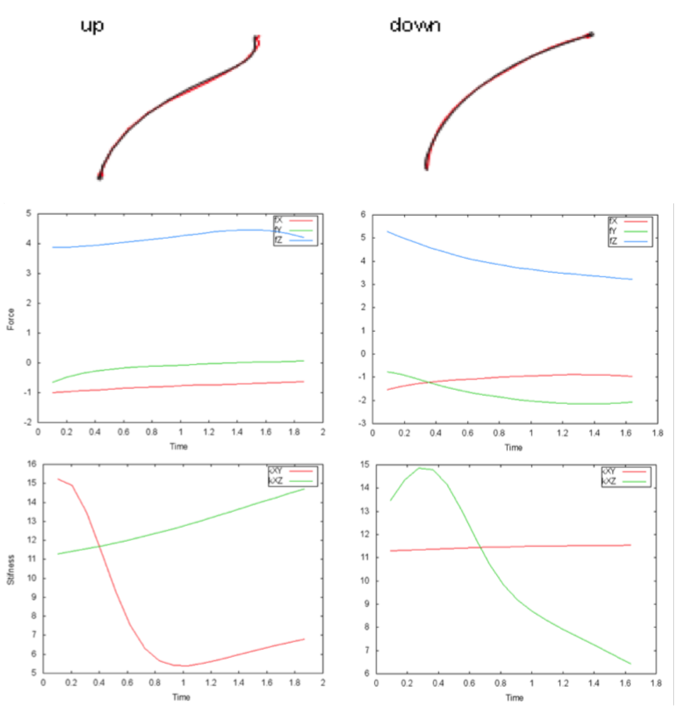


Figure 4.9. The results of two of the EA runs generating controllers that would follow almost exactly the human trajectory. On the bottom it shows for those two cases (upward and downward) the forces applied and the stiffness scaling factors evolved.

Results From Matching the Human Data in One Quadrant To start with, controllers that would be able to control the robot arm to use a trajectory and (normal) force profile as close as possible as the human measured data were evolved. These runs aim to generate a more and more optimized model of the stiffness adaption to come closer to the human performance. The control output $u(t)$ and the scaling factors are functions of time here. The evolution was done for specific trials generating specialized feed-forward controllers for specific tasks. The first controllers to be evolved were designed to allow for a very precise reconstruction of the human trajectory and normal force applied (Figure 4.9). This was achieved rather quickly and showed that the chosen approach is feasible.

Results in One Quadrant Without Matching Human Data ('Optimal Control') The next try involved a controller, which has just the same high-level information as the humans have: its aim is to go from point A to point B while following the surface of the sphere. The controller in this case does not have to follow the same

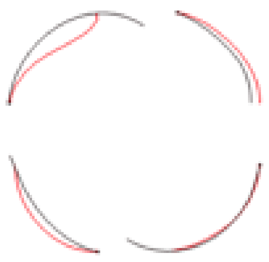


Figure 4.10. Plot of an evolved controller performing a full circle trajectory. It tries to optimize the control, i.e. minimize the normal force.

trajectory as the human subjects, but it tries to minimize the time and forces. Stiffness scaling is not used for these controllers. For this the fitness function was simplified to:

$$
\begin{aligned}
f = &\sum_{t=0}^{T} F_n(t) * w_1 + && \text{(sum of forces)}\\
&\sum_{t=0}^{T} |p(t) - p_t| * w_2 + && \text{(sum of distance to target)}\\
&\sum_{t=0}^{T} |p(t) - R| * w_3 && \text{(surface error)}
\end{aligned}
$$

Results for Full Circle Controller Without Matching Human Data ('Optimal Control') This is an intermediate step towards the overall goal of generating a controller that would be the single control for all trajectories by the human. Trying to find if one control policy can be found that describes the whole range of human movements is though a harder problem than the one tackled above. Here we are trying to see if we can find a controller based on the optimized model that would generate a trajectory to follow (in a clockwise fashion) the targets (see Figure 4.10). Only one direction was used and a speedup was gained because the evolution does not need to compare each controller with the human trajectory. When evolving the optimal controller to operate in multiple quadrants, only one direction of motion (clockwise) was used. During evolution, only the behaviour in three quadrants contributed to the fitness with the remaining quadrant (top-left) being used to show how the behaviour generalised.

Towards A Single Controller For Multiple Human Trajectories Another step forward is to test a single controller on multiple trajectories (e.g., up and down in a specific quadrant). Based again on the assumption that humans would not change their behaviour for the different task. The interesting question to further study is 'how does the one unique policy look'? And can we find it with our controllers? Finally the goal for the future is to look for a controller that is able to steer the endpoint through all four quadrants, matching the human trajectory and forces.

The first try to find such a single control for multiple paths was to evolve the controller over two trajectories (up and down) in a single quadrant. One attempt is shown in Figure 4.11, where it can be seen that the forces (over time) are the same for both trajectories, however this leads to quite different trajectories because of the surface restrictions and the different starting points. The results are of very poor quality (including not reaching or ending at the target) and this appears to be because the basic control as a function of time $u(t)$ does not provide enough information about the current task. Clearly the strategies that humans employ to control their motion (and how to apply force) are more complex than just a simple time-dependent control. A possible solution to find the a solution that might approximate the underlying principles is to provide the controller with more information about the task to be solved. Another possibility is that the model we use does not adequately capture the dynamics of the human arm, and so is unable to replicate the results. Using an alternative robot arm model may therefore be necessary.

For a second try one could evolve controllers, where the emitted forces are a function of distance in $[x, y, z]$ from the target position $u(\delta X)$, whereas the stiffness scaling factors are a function of the current position. This approach required extending the Gaussian kernel representation to accept higher-dimensional input parameters. This more flexible approach may be interesting to investigate in future work. This significantly alters the search algorithm used, and is currently a work in progress.

Discussion During the STIFF project a focus was put on on generating controllers that in simulation of specific tasks achieve comparable results to the recorded human motion. The human performance measurements at UPD defined a task aimed to help describing arm stiffness behaviours. We tested the approach chosen on various parts of the dataset received from UPD. This was based on a simple controller and was used previously to describe some aspects of human stiffness. We demonstrate here that our evolutionary algorithm approach is capable of generating a controller for the simulated robot arm, following the human trajectory and matching the human applied forces in 3D. The controller outputs both force control signals and values that alter the cost function's interpretation of how stiffness is being utilized.

These simple controllers were not capable enough though to generate a single control policy for multiple trajectories. They are not able to describe what strategy humans are following, yet it is believed that this research might allow to describe certain aspects of why the trajectories differ in one quadrant. This will need further investigation and possible more experimentation with human test subjects.

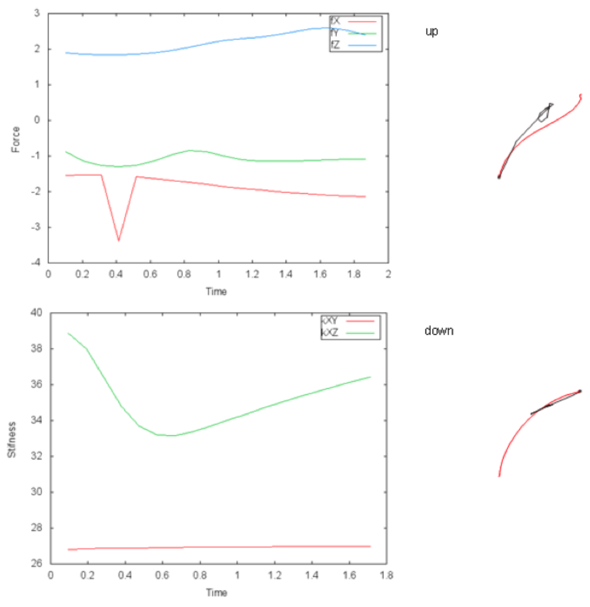


Figure 4.11. The left side shows the force and stiffness scaling profile used for the controller in both test runs. The trajectories generated for the upward and downward motion can be seen on the left side in black (with the human trajectory in red).

Bio-signal Processing

The *WAY* project investigates new ways to link the brain with upper limb aids. It hopes to restore a physiological bidirectional link between artificial aids and patients, to improve the ability of users to perform activities of daily living (ADL) and thus to attain enhanced autonomy and quality of life. The aim is to employ already available sensorized hand assistive devices – i.e., a dexterous prosthesis and an exoskeleton built at the Scuola Superiore Sant'Anna in Pisa – and by developing non-invasive wearable interfaces designed for bidirectional data flow of sensory information and motor commands (Figure 4.12). *WAY* hopes to bridge several currently disjointed scientific fields and is therefore critically dependent on the collaboration of engineers, neuroscientists and clinicians. The *WAY* project investigated ways for improving classification accuracy of electromyography (EMG) pattern recognition in different limb postures.

The term biomedical signal or bio-signal generally refers to any electrical impulse acquired from an organ or part of the human body that represents a physical variable of interest [Merletti and Parker, 2004].. Like other signals, it is considered a function of time and described in terms of its amplitude, frequency and phase. The EMG signal represents the stimulus sent to achieve a certain neuromuscular activity. An EMG sensor measures electrical currents generated in a muscle during its activation. The signal itself stems from the control of the muscles by the nervous systems and is dependent on the anatomical and physiological properties of muscles. Furthermore the signal collected contains noise acquired while traversing through varying tissues. In addition the collection of the signal by the electrode most likely contains the signal from multiple motor

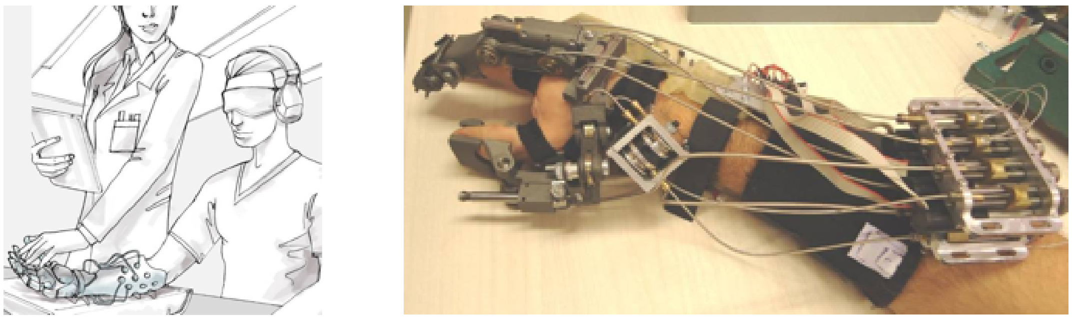


Figure 4.12. On the left the vision at the start of the project, a prosthetic stump, connected to an artificial hand prosthetic hand with embedded touch sensors, on the right the new prototype of the exoskeleton developed within the *WAY* project, to be used in human trials.

units, generating interference and signal interactions.

The first documented research on electricity and its relation to muscles is the experiments by Francesco Redi [1686]. In the mid-1600 he discovered that highly specialised muscles in the electric ray fish are generating electricity. In the 18th century the famous experiments by Luigi Galvani showed that electricity could also be used to stimulate muscle contraction [Galvani, 1791]. The first recording of electrical activity during voluntary muscle contraction was performed by Étienne-Jules Marey [1890], who also introduced the term electromyography. In the 20th century ever more sophisticated methods of recording the signal were invented, with the first surface EMG (or sEMG) electrodes being used by Hardyck et al. [1966].

There are two methods to collect EMG signals, directly in the motor units with invasive electrodes or with non-invasive, surface attached electrodes. Both methods have their own advantages and disadvantages. We are focussing on sEMG herein and during the *WAY* project. The comparatively easy setup has the drawback of increased noise, motion artefacts and possible readout failures when losing the contact to the skin. EMG has previously been used in medicine to measure the rehabilitation in case of motor disabilities, but only recently has it been identified as a possibly useful input in applications, such as, prosthetic hand control, grasp recognition and human computer interaction (HCI) [Cipriani et al., 2011].

Machine Learning for EMG

Machine learning techniques have already been used to classify EMG signals from one subject or one instrument previously. Of interest nowadays is to compare how well these techniques perform with multiple subjects, as well as, collected with multiple instruments. Furthermore we investigate how classification can still be achieved with limited available training data. This is of interest, not just to allow flexibility in who is controlling the robot but also in other areas of application, such as rehabilitation or prothesis control. It is important to not exhaust (or annoy) the subjects, amputees or people with other impairments, with too much training. Reaz et al. [2006] published an overview of techniques applied to the problem of detection, classification and decoding EMG signals.

The NASA JPL developed BioSleeve [Wolf et al., 2013] contains an array of electrodes and an inertia-measurement unit that can easily be put onto a subjects forearm. The detection is using a learning approach (multi-class support-vector-machines) to decode static and dynamic arm gestures.

EMG Measurement

The collection of EMG signals on the operator allows to control the robot's grasp by performing the grasp itself. To necessary signals are collected using surface electrodes mounted on the operator's lower arm (Figure 4.14). In this figure you can also see the accelerometer placed at the back of the hand (or the wrist) of the subject. To collect the data a system called *BITalino* was employed (also visible in the picture). It is developed and produced by the Instituto de Telecomunicações in Portugal and their spin-off PLUX.[6] It is a small stand-alone embedded system with its own battery, allowing to collect up to 8 data channels without cables running from the subject to the computer. The channels, which can be either EMG or electrocardiography signals, accelerometer data or luminosity information are transferred from the system via Bluetooth to a computer in real-time. With this signal the intent to grasp an object can be detected. This results in a command to the robot to perform the equivalent action (using the subsystem mentioned above).

This human motion detection using the bio-signals was tested together with the integration of the visual perception from Chapter 3 and our safety/world model from Section 4.2.

Experiments: Remote Control of the *iCub* With Signals Collected From the Operator

The tele-operation of complex robots remotely is of interest for a variety of application areas, from space exploration to handling radioactive materials, from bomb disposal to disaster response scenarios (as selected for the DARPA Robotics Challenge). In every tele-robot system time lag and the connection bandwidth will limit a full immersion of the operator. Therefore to ensure a safe operation various modules need to take over certain aspects on the robot side. These modules, mainly, the low-level control, obstacle avoidance and obstacle detection are integrated into a full system.

Two separate setups were tested to show that the humanoid can be controlled safely by means of (rather noise) bio-signals from a remote operator. In both cases self-collisions and collisions with the environment were completely avoided.

[6]Webpage: `http://bitalino.com/`

Remote Control using Accelerations

The setup was tested at first with a standard 6-axis gamepad to show the efficacy of our approach and control system (Figure 4.13 shows this in action). The two sticks on the pad are used to control each of the two arms respectively. Up/down, left/right motions of the sticks would apply fictional (operational space) forces (relative to the stick displacement) to the robot's end effector. To force the arms forwards (away from the body) or back, two more buttons (triggers on the back-side of the pad) were incorporated. The 'forcing' of the dynamical system to control the robot's arm in operational space is using the the same method as the virtual force fields for obstacle avoidance. Here an additional virtual force is added based on the operator's signal.

Various test subjects were allowed to control the robot, without receiving too much information on how the system or controller works. After their initial fear that they could break the robot and the realization that the robot would prevent all impeding collisions, they were enjoying to control the both arms of the robot simultaneously. To test how an operator could perform ADL remotely, functionality to trigger a grasp was added. This was simply done by pressing one of the gamepad buttons, which would then send the appropriate command to the *LEOGrasper*.

Moreover to create a better, more intuitive user interface a LEAP sensor was employed instead of the gamepad. This sensor developed by Leap Motion Inc. is a small USB peripheral device designed to be used as a novel input method for computers. Similar to the structured light sensors described earlier, it contains two monochromatic IR cameras and three infrared LEDs that project a 3D pattern

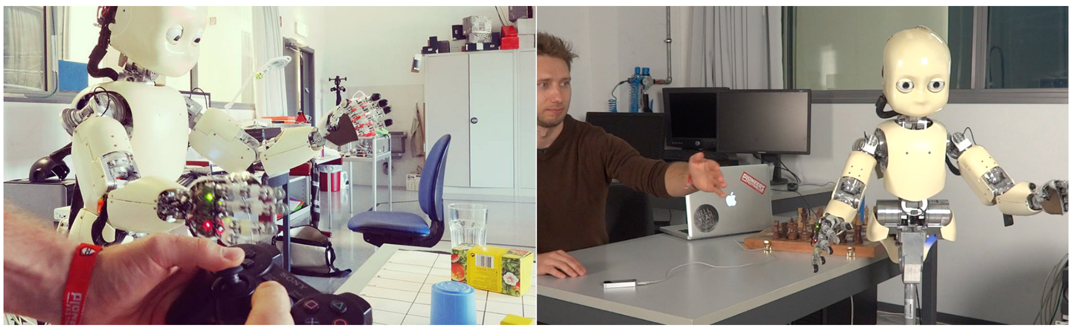


Figure 4.13. The *iCub* being controlled by a standard 6-axis gamepad (left) and a Leap Motion sensor (right) (video available at: `http://www.youtube.com/v/5i6tAOb922g`).

into the workspace.[7] The output is the 3D position of hands and some basic information about the gestures executed.

The test subjects were able to place the robot's hand close to targets specified by us. They also realised quickly that the robot is able to react to changing environments. This lead to them being less scared of breaking the robot and trying more complex actions. The next chapter goes a bit more into the details of how this was then extended to function in cluttered and non-static environments (e.g. Figure 5.2 shows *MoBeE* and its information about impeding collisions, and Figure 5.3 shows a sequence revealing the robot's motion). For this the objects and obstacles need to be detected and tracked prior to be avoided. This is done using the *icVision* subsystem. The created interfaces between *MoBeE* and *icVision* allow for a continues visual based localization of the detected objects to be propagated into the world model. This very basic 'eye-hand coordination' allows for an adaptation to changing circumstances, while executing the behaviours specified by the operator, improving the usability of the whole system.

EMG and Accelerations

This next experiment employs the recorded bio-signals of the operator (the arm's accelerations and the EMG signals) to control the *iCub* humanoid. Instead of the gamepad used previously a 3-axis accelerometer was connected to the *BITalino* system. It was placed on the operator's hand (back) or wrist for some individuals. The BITalino was furthermore facilitated to collect the EMG signal measured at the *extensor digitorum communis* (see the electrodes in Figure 4.14). The differential between the two electrodes is measured and sent back via Bluetooth, a third electrode is used as reference/neutral and is attached around the elbow.

The low-pass filtered accelerations of the arm are used directly to 'force' the end-effector, similar like with the gamepad in the previous experiment. In addition the EMG readouts are used to trigger a closing or opening of the *iCub*'s hand. A simple mean-average-value (MAV) thresholding was used, which seemed to reasonable for most subjects after a quick manual tweaking of the MAV threshold. The opening and closing though required to be more forced than a regular use of the hand to provide a better signal-to-noise ratio.

This was tested with a handful of subjects at IDSIA. While the setup is less comfortable than the gamepad, a certain sense of excitement was clearly visible in the subjects. The motion of the arm though produced a rather noisy signal, yet because of the mechanisms in place (mainly *MoBeE*) this is not dangerous

[7]This video shows the pattern as recorded with a webcam: `http://www.youtube.com/v/UI5EBzU_QqM`

for our robot. The reasons for the noise are most likely the motion of electrodes on the skin (or even detachment from the skin), some interference in the cables while they are being moved around, and last but not least the change of the EMG signal due to the change in the dynamics of the system (e.g. counteracting gravity effects the muscle contraction therefore the signal). While the *WAY* project has concluded, it would still be interesting to use machine learning techniques to classify more robustly both the acceleration data and the EMG signal (probably of more channels). Especially of interest would be the separation of different grasp types.

Herein a system enabling an operator to control the end-effector of a complex humanoid using either a gamepad or an accelerometer attached to his/her arm. The robot then performs the equivalent operation, while still avoiding obstacles and collisions. A simple EMG measurement can be used to trigger grasps. This setup worked very well for when the user was placed behind the actual robot, when the operator was facing the robot, a few test subjects had issues because from their point-of-view the right stick would control the left arm. It was seen that the control of the arm alone yields only a very limited reach space for our humanoid robot. An intuitive way to control also the torso would clearly improve this.

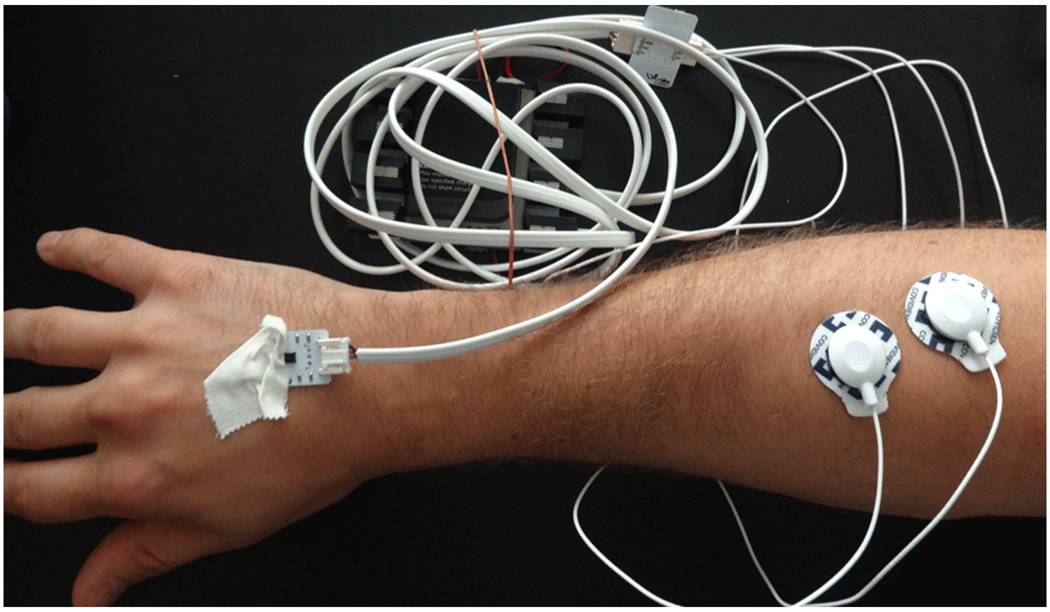


Figure 4.14. The setup for collection of one of the bio-signal (in this case, EMG signals and accelerations of the hand) at IDSIA. The data is collected by the *BITalino* sensor, which sends the data via Bluetooth to the operator's computer.

Recap

In this chapter the action side required for eye-hand coordination is described. The main contributions apart from the interfaces developed for other tools created at IDSIA, namely *MoBeE* and *TRM* (these are pre-requisites for the coordination detailed in the next chapter), are:

- *LEOGrasper*: the development of a simple grasping subsystem, based on previous experiments in Matlab by Leo Pape. This flexible implementation in C++ was created, consisting of simple interfaces to *MoBeE* and the robot. It has recently been stress-tested during the USI Ten-Year-Informatics event.

- *Human Motion Decoding*: the investigation of collected data from human motions. The data was collected mainly at the partner institutions of the STIFF and WAY projects. Various machine learning techniques were used to decode the recorded sequences and use it as a control input for our complex humanoid. A pure time-dependent decoding seems to not yield the required results, other options will have to be pursued in the future.

Chapter 5

Integration and Sensorimotor Coordination

Although there exists a rich body of literature in computer vision, path planning, and feedback control, wherein many critical subproblems are addressed individually, most demonstrable behaviours for humanoid robots do not effectively integrate elements from all three disciplines. Consequently, tasks that seem trivial to us humans, such as picking up a specific object in a cluttered environment, remain beyond the state-of-the-art in experimental robotics.

It is becoming increasingly clear that robotic systems will need to exhibit sophisticated capabilities in the future for the tasks they are proposed to perform. This will require advances along the complete processing 'pipeline', from sensing through to learning and interaction.

Sensory feedback is of critical importance to decide and act in unstructured environments. Robots require the ability to perceive their surroundings, as it is generally not feasible or possible to provide all the information needed a priori (the most striking example being robotic space exploration). Creating a functional perception system is a hard, but important, problem. It is a prerequisite for robots to act in a purposeful, 'intelligent' way. In Chapter 3 a technique that allows for visual perception based on a machine learning approach was presented. It also enables the continues adaptation and update of the object detection filters. Clearly vision is an important thing for a humanoid robot, but it needs to be closely integrated into the control and motion sub-system (as described in Chapter 4). To facilitate more autonomous and more adaptive behaviours a close integration of computer vision and control modules is developed and implemented to 'close the loop'.

5.1 Background: Coordination and Adaptation

The main aim of my doctoral research is a closer integration of the sensory and motor sides and creating an closely tied action-perception loop. Interfaces between MoBeE and *icVision* allow for a preliminary visual based localization of the detected objects and placement of them into the world model. Currently very simple operational space control is employed to move the arm to the target object. This, rather open-loop control, is the first step towards visual guided reaching and grasping of objects with the *iCub*. It enables the robot to detect an object in the visual frame then localize it in Cartesian coordinates and eventually executing a reach, with either the existing operational space controller or our TRM approach. While this approach is already a step forward of current systems, it requires a very accurate calibration of both camera and mechanical links to be successful. However some mechanical non linearities still cannot be taken into account.

Sensorimotor Development is aiming to learn a basic eye-hand coordination. This type of coordination for robotics has previously been investigated using extra sensors like LASERs or other helpers mounted on the end-effector, such as, LEDs, bright coloured symbols or markers, etc. Langdon and Nordin [2001] have shown this on a simple humanoid robot using GP techniques. Hulse et al. [2009] had shown that machine learning to grasp a ball with a robot arm is possible. Yet only the ball, not the arm was visually detected. Another approach to detect hand gesture directly, using a coloured glove and machine learning was presented by Nagi et al. [2011]. Various methods for learning this sensorimotor mapping have been investigated [Hoffmann et al., 2005; Hülse et al., 2010]. These lead to biologically inspired mappings, yet applying these directly to control the robot is still an issue. *Adaptation* is needed for precise object manipulation, as highly accurate models of the world and also of the robotic system can not be assumed in most interesting scenarios. An estimation of the robot kinematics might help in generating more precise motions. So far no module exists to estimate the kinematics of the *iCub*, this is partly due to the openly available CAD models and thorough calibration procedures that need to/should be applied regularly. For other robotic platforms machine learning has been used to estimate the kinematic model, e.g. Gloye et al. [2005] used visual feedback to learn the model of a holonomic wheeled robot and Bongard et al. [2006] used low-dimensional sensory feedback to learn the model of a legged robot.

Another often used option to track and detect the position of the robots end-effector are external motion capturing and imaging systems (e.g. [Oikonomidis et al., 2011; Ehara et al., 1995]). Recently, the German Aerospace Agency

(DLR) has investigated methods for better position estimation for their humanoid robot. Using RGB-D (Kinect) and a stereo camera approach both combined with a model-based technique, their system was able to qualitatively decrease the error from the pure kinematic solution [Porges et al., 2012]. An earlier approach describes a simple 5DOF arm being controlled by a trained neural network and visual feedback. The robot is using a calibrated, fixed stereo camera pair to move the end effector to a goal position [Kuperstein and Rubenstein, 1989]. A drawback of this early implementation was not able to avoid fixed or moving obstacles.

Interesting research towards sensorimotor coordination on a humanoid arm were described in Tuci et al. [2010]. Using continuous-time recurrent non-linear networks are used in to control and classify object shapes. While the approach was so far only tested in simulation where there are no constraints and collisions are not a problem, it seems that this approach together with MoBeE might lead to interesting research avenues in the future.

The idea to combine the planning of motions with vision is not novel, in fact it's a intuitively quite clear that this is a very useful skill for any robot to posses. In the past such sensorimotor coordination has been investigated from various aspects. For example, obstacle avoidance for humanoid (soccer) robot was developed based on the detection of very simple objects (fixed colour, fixed form) and a behaviour-based action stack [McGill et al., 2012]. The obstacle avoidance though is solely in the plane and leads only to simple navigational actions, not complex human-like motions.

Already earlier Dickinson et al. [1993] were using a layered object detection approach to coordinate between actions and vision. This work, like many others, has tried to re-plan the robot's action when a new object is detected, and use only reactive vision. This leads to a turn-based motion of the robot, where each new observation usually leads to a re-planning phase. As mentioned already, visual feedback control of complex robots in complex environments is still a challenge in robotics [Kemp et al., 2007].

The approach described herein tries to overcome the strict constraints on objects and robots, especially on the use of discrete, behaviour-based architectures with re-planning steps.

5.1.1 Integration

Even when sufficient manipulation skills are available these need to be constantly adapted by an perception-action loop to yield desired results. "Robotics, Vision and Control" by Corke [2011] describes common pitfalls and issues when trying to build such systems with high levels of sensorimotor integration. To allow for a variety of objects to be picked up from various positions the robot needs to see, act and react within an integrated control system. For example, methods enabling a 5 DOF robotic arm to pick up objects using a point-cloud generated model of the world and objects are available to calculate reach and grasp behaviours [Saxena et al., 2008]. A technique for robots to pick up non-rigid objects, such as, towels was presented [Maitin-Shepard et al., 2010]. It allows to reliably and robustly pick up a towel from a table by going through a sequence of vision-based re-grasps and manipulations-partially in the air, partially on the table.

In the DARPA ARM project, which aims to create highly autonomous manipulators capable of serving multiple purposes across a wide variety of applications, the winning team showed an end-to-end system that allows the robot to grasp and pick-up diverse objects (e.g. a power drill, keys, screwdrivers, ...) from a table by combining touch and LASER sensing [Hudson et al., 2012].

Visual Servoing [Hutchinson et al., 1996; Chaumette and Hutchinson, 2006] is a commonly used approach to closed-loop vision based control of robotic systems – i.e. some level of hand-eye coordination. It has been shown to work as a functional strategy to control robots without any prior hand-eye calibration [Vahrenkamp et al., 2008].

5.2 Closing the Action-Perception Loop

The sensory and motor sides establish quite a few capabilities by themselves, yet to grasp objects successfully while avoiding obstacles they need to work closely together. The continues tracking of obstacles and the target object is required to create a reactive reaching behaviour which adapts in real-time to the changes of the environment.

Interfaces between *MoBeE* and *icVision* were created and allow for a continues visual based localization of the detected objects to be propagated into the world model. The evolved filters for each object and the hand can be used to update the positions. This basic approach to eye-hand coordination allows for an adaptation while executing the reaching behaviour to changing circumstances, improving the autonomy of the humanoid robot.

Already in Chapter 2 an overview of the on-going research towards a functional eye-hand coordination system on the *iCub* at the IDSIA Robotics Lab was show (Figure 2.1). The various modules (grey boxes) described in the previous chapters were developed during the last years and represent the subsystems:

perception (green), action (yellow) and memory (blue). To recap the various modules are:

- *Object Models, Detection and Identification:* as mentioned earlier, the detection and identification of objects is a hard problem. To perform these tasks CGP-IP is providing an approach to learn visual object detection. The resulting programs perform the (binary) segmentation of the camera images for specific objects.

- *Object Localization:* by using the image coordinates of the detected object from the two cameras together with the current robot's pose, the position of the object can be estimated in Cartesian space wrt. the robot's reference frame. Instead of a calibration for each single camera, the stereo system and the kinematic chain, a module that learns from a training set is incorporated, as shown in Section 3.5. After the location of an object is estimated, the world model is updated, i.e. the a priori known geometry of the object is placed correctly.

- *World Model, Collision Avoidance and Motion Generation:* the world model keeps track of the robot's pose in space and the objects it has visually detected. Figure 5.1 shows this model including the robot's pose the static table, and the objects localized from vision. *MoBeE* is used to safeguard the robot from unwanted interactions with these geometries in the world model. It furthermore allows to generate motion by 'forcing' the end-effector in operational space, e.g. towards an object to be pickup up.

- *Action Repertoire:* herein we only use a light-weight, easy-to-use, one-shot grasping system described in Section 4.4. More actions and tasks, as generated by our TRM approach, should be included in future works. One can even envision them to be created on the fly during the robot's operation.

The aim is to generate a pick-and-place operation for the *iCub*. For this, functional motion and vision subsystems are integrated to create a closed action-perception loop. The vision-side detects and localizes the object continuously, while the motor-side tries to reach for target objects, while avoiding obstacles. A grasp is triggered when the hand is near the object.

The integration requires to specify clear interfaces between those modules, while ensuring that the dependencies do not hinder a real-world, real-time application of the whole system. A typical workflow can be imagined to go like this: A new set of images is received from the cameras at the *icVision* framwork.

The images are sent to all active filters. These filters perform the object segmentation and provide as outputs, a binary image (single-channel), as well as, some numerical information, such as the size and location of blobs, their estimated 3D position based on the *icVision* localization abstraction, and more minor details. The estimated position can be from the learned system presented earlier or the IIT provided library. Each filter sends this information (together with an object ID) to *MoBeE*, which in turn updates the world model with a pre-defined geometry. Any on-going control of the robot, through the *MoBeE* proxies will now 'feel' a force from the detected objects to avoid collisions.

5.3 Experiments & Results

The first experiment shows that the herein presented system is able to reactively move the arm out of harms way when the environment changes. We then show how we can use this system to reactively reach and grasp these objects when we change their type from 'obstacle' to 'target', therefore changing the fictional forces calculated. Lastly an experiment is described which combines the detection, localization and actions to pick up an object and further improve the learned CGP-IP classifier by doing so.

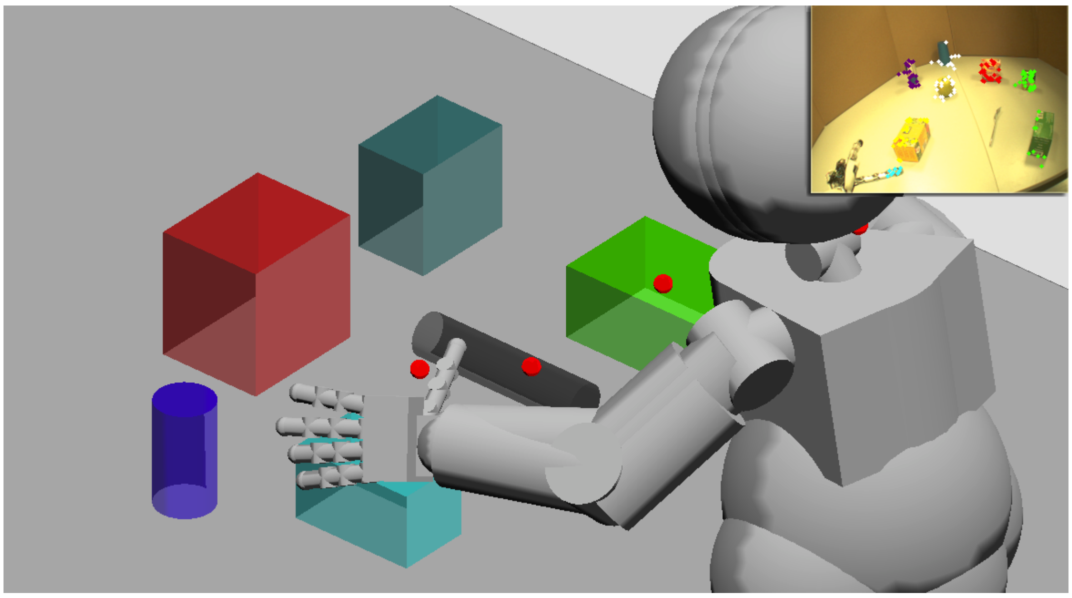


Figure 5.1. World model generated in the MoBeE framework using the output from learned filters. The inset shows the camera image.

5.3.1 Avoiding a Moving Obstacle

Static objects in the environment can be added directly into *MoBeE*'s world model. Once, e.g. the table, is in the model, actions and behaviours are adapted due to computed constraint forces. These forces, $f_i(t)$ in (1), which repel the robot from collisions with the table, governs the actual movement of the robot. This way we are able to send arbitrary motions to our system, while ensuring the safety of our robot (this has recently been shown to provide a good reinforcement signal for learning robot reaching behaviours [Pathak et al., 2013; Frank et al., 2014]). The presented system has the same functionality also for arbitrary, non-static objects. After detection in both cameras the object's location is estimated (*icVision*) and propagated to *MoBeE*. The fictional forces are calculated to avoid impeding collisions. Figure 5.2 shows how the localized object is in the way of the arm and the hand. To ensure the safety of the rather fragile fingers, a sphere around the end-effector can be seen. It is red, indicating a possible collision, because the sphere intersects with the object. The same is valid for the lower arm. The forces, calculated at each body part using Jacobians, push the intersecting geometries away from each other, leading to a forcing of the hand (and arm) away from the obstacle. Figure 5.3 shows how the the robot's arm is avoiding a non-stationary obstacle. The arm is 'pushed' aside at the beginning, when the cup is moved close

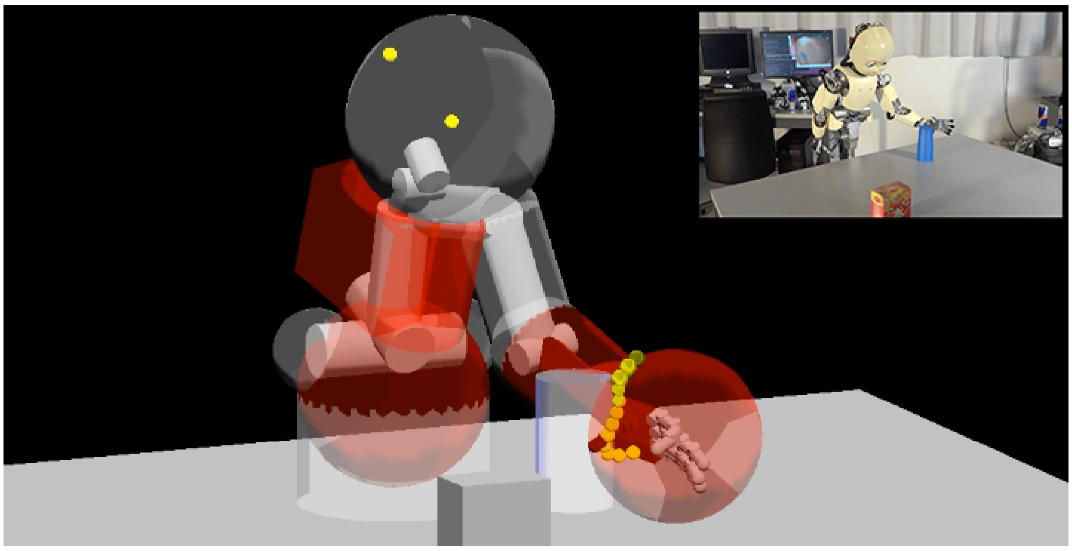


Figure 5.2. Showing the visual output of the *MoBeE* world model during an experiment. Parts in red indicate (an impeding) collision with the environment (or itself). The inset shows the actual scene. A video of this can be found at: `http://www.youtube.com/watch?v=w_qDH5tSe7g`

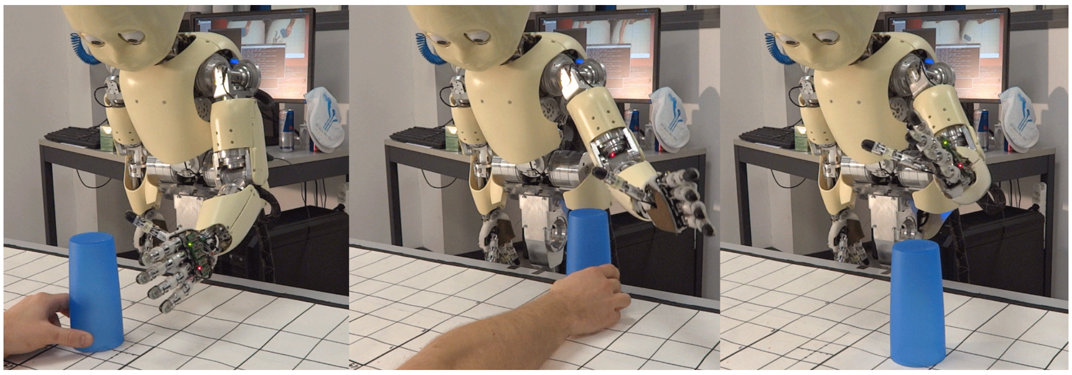


Figure 5.3. The reactive control of the left arm, allowing the *iCub* to stay clear of the static and non-static obstacles.

to the arm. It does so until the arm reaches its limit, then the forces cumulate and the end-effector is 'forced' upwards to continue avoiding the obstacle. Without an obstacle the arm starts to settle back into its resting pose q^*.

5.3.2 Reaching and Grasping Objects

This next experiment is on a simple reactive pick-and-place routine for the *iCub*. Similarly to the above experiment we are using *MoBeE* to adapt the reaching behaviour while the object is moved. To do this we change the type of the object within the world model from 'obstacle' into 'target'. Due to this change there is no repelling force calculated between the object and the robot parts. In fact we can now use the vector from the end-effector to the target object as a force that drives the hand towards a good grasping position.

MoBeE also allows to trigger certain responses when collisions occur. In the case, when the robot is supposed to pick-up the object, a grasp subsystem is triggered whenever the hand is in close vicinity of the object. Currently a prototypical power grasp action is applied, which has been successful in a lot of cases especially during our various demos and videos.[1] Figure 4.6 shows the *iCub* successfully picking up (by adding an extra upwards force) various objects using our grasping subsystem, executing the same action. As shown in Figure 5.2 our robot is able to track multiple objects at the same time. By simply changing the type of the object within *MoBeE* the robot reaches for a certain object while avoiding the other.

[1]See Appendix C for a list of videos or check the webpages of the IDSIA Robotics Lab `http://robotics.idsia.ch/media/` or the author `http://Juxi.net/media/`

5.3.3 Improving Robot Vision Through Interaction

One of the advantages of a (humanoid) robot is that it can actually manipulate the world around it. As shown above the *iCub* is now capable of simple pick-and-place actions. One of the most interesting applications is to facilitate improved object detection by interacting with the world around the robot.

Herein an experiment is proposed, combining CGP-IP simple manipulation actions to generate better detectors (filters). The experiment aims for learning specific object representations that can be applied (and reused) in visual detection and identification tasks. This section describes how visual object representations can be learned and improved by performing object manipulation actions, such as, poke, push and pick-up, by the *iCub*. The improvement can be measured and allows for the robot to select and perform the 'right' action, i.e. the action with the best possible improvement of the detector.

To build more accurate and robust representations, allowing the detection and identification in a wide range of settings, interaction is of critical value. CGP-IP is used to create these models based on a series of images collected during the robot's interaction with the world. It is shown that the improvement can be measured using this method described herein. Therefore the robot is able to select the 'right' action, i.e. the action providing the best possible improvement of the detector.

Researchers started addressing how in robotic settings a more autonomous fashion to object detection can be devised [Kim et al., 2006], as well as, how the need for a human teacher can be minimised [Gatsoulis et al., 2011], compare also Section 3.4.5. Yet all these approaches neglect the possibility of the robot taking actions to improve its ability to detect objects – i.e. the scene the robot is looking at is considered static (at least during the learning phase). Robotic vision has the ability to interact with these objects and learn more and better models about the world.

In the primate brain visual stimuli and motor commands are closely intertwined [Rizzolatti and Craighero, 2004], and it has been suggested that this enables more adaptive, more autonomous behaviours. For robots to act in such a fashion a higher level of integration and coordination between perception and control is necessary. In the last few years robot vision research has been extended to investigate how the embodiment of the robotic system can be used to create better visual perception skills.

How to do motion planning for a high DOF is another issue that needs to be tackled to perform useful actions. There has been a extensive research on this in the past (see e.g. LaValle [2006]; Li and Shie [2007]; Peters and Schaal

[2008]. A commonly used approach is based on Rapidly Exploring Random Trees (RRT) [Kuffner Jr and LaValle, 2000]. For example, Vahrenkamp et al. [2012], used a variant of RRT for both planning the reach and the grasp on a high DOF humanoid robot. Few have been investigated on how to plan for a good action to allow better visual recognition.

One of the first to investigate robot actions for robot vision were Metta and Fitzpatrick [2003]. They showed that performing actions can lead to a simple object segmentation based on using optical flow information. Their work extracts visual information through autonomous exploration of the environment. When the robot hits an object placed on a plane in front of it, a binary segmentation is performed based on the object's motion. The authors highlight that following the causal chain from the robots action allows to develop visual competence. While the theoretical impact is large it does not go into the details of how to generate suitable object representations for online object search and recognition.

With the rise of cheaper stereo vision systems and depth sensors most of the object recognition seems to focus on 3D geometry, or shape-based, detection. Welke et al. [2010] showed an implementation that allows one to build visual detection based on a depth information and a 3D view sphere concept. Using a static camera setup they are able to generate a motion of the arm to rotate and cover as much of a 3D sphere as possible, allowing to build a model of the object from these multiple views. A similar approach by Gonzalez-Aguirre et al. [2011] tried to overcome some of the issues when trying to detect general objects based on their 3D shapes purely from vision. By fusing multiple views and vantage points a more robust detection was generated. In contrast our work does not require or even try to build 3D models of the objects.

A complex, humanoid robot, with its extra DOF, allows for more and different types of 'interaction' with the environment, also with respect to robot vision applications. Yet little work has investigated these extended possibility, such as in our `LEAN` action. Stasse et al. [2008] suggest that using the hip can extend the view area and therefore the possibility for detecting known objects. Their work, based again on a next-best geometric view point approach, can derive motions of the hip to change view points. It is not clear though how their technique can handle redundant DOF.

The focus here was on developing visual detection models for four distinct objects (Figure 5.4), without the use of any 3D or shape models or other priors. Each model shall uniquely detect one object, also solving the object identification problem at the same time. The aim is to create models that can be run, in parallel, on the real robot and allow for robust detection and identification of these objects in changing environments even when other objects are present.

To learn robust models of the objects our humanoid robot needs to perceive the objects in various poses and from various angles. For example, the tea boxes (in Figure 5.4), differ visually between the front and back side. To learn better models for the detection of these objects, a pre-selected action set is available to the humanoid robot. These actions allow to see the object from various angles.

The available, scripted actions are the following:

- **Action** `LEAN`**:** this simple action using 1 DOF, changes the position of the torso, by performing a hip motion. The robot will lean about 30° to its left and then 30° to the right, while keeping its gaze fixed at the position of the object on the table (Figure 5.5).
- **Action** `POKE`**:** the extended index finger of the end-effector is used to poke the object from the right side. A (more or less) linear movement of about 10cm in operational space is performed, while the robot gazes at a fixed location (Figure 5.6).
- **Action** `PUSH`**:** the palm of the right hand is used by the robot to push the object from right to left. This is again a linear movement, but with higher velocity and longer path. The gaze is controlled to continuously look at the end-effector (Figure 5.7).
- **Action** `CURIOUS`**:** is the most complex action available and is modelled after

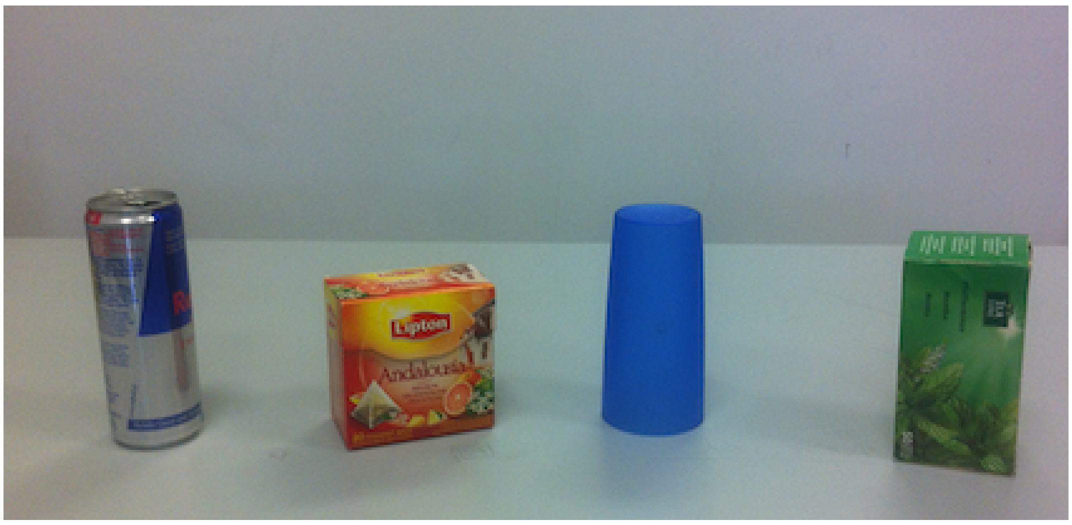


Figure 5.4. The objects to be detected and distinguished in this scenario are: a soda can, a yellow-red tea box, a blue plastic cup and a green tea box.

a child curiously exploring an object. The object is grasped, picked-up (from a fixed location using a predefined grasp primitive) and then brought closer to the face while being rotated in various ways. These rotations involve almost all DOF of the wrist, elbow and shoulder. The gaze during this whole motion is continuously adjusted to look at the hand and the object it is holding (Figure 5.8).

Camera pictures are recorded in a fixed interval , while the robot is performing these actions. These are then used to improve the visual models. To allow for a fair comparison, as the actions have different runtimes, only 5 images are taken from each action of the experiment. This is done to be able to compare the results. During these experiments a predefined area of interest in the camera

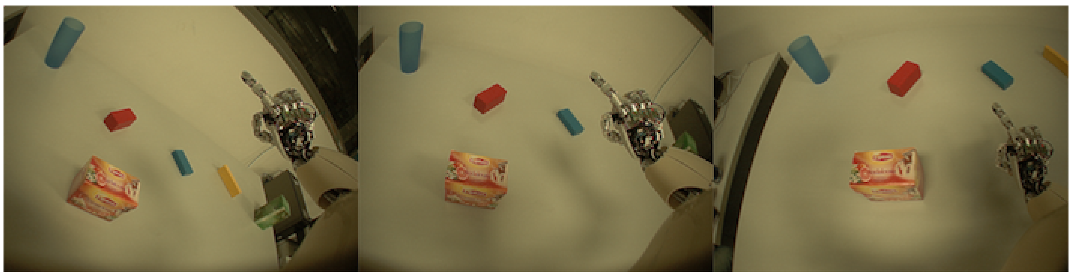


Figure 5.5. An example of a series of images collected during a `LEAN` action.

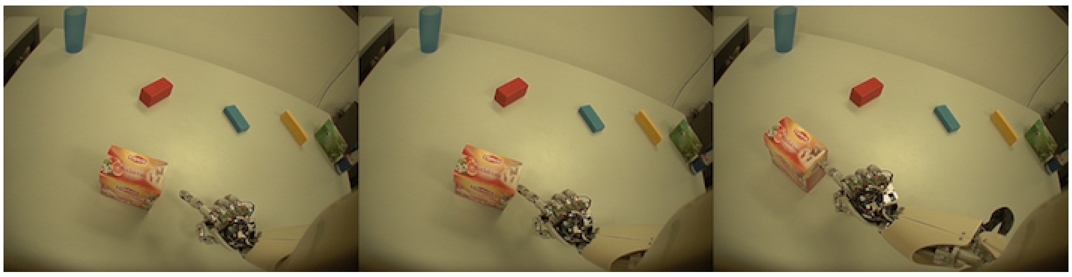


Figure 5.6. An example of a series of images collected during a `POKE` action.

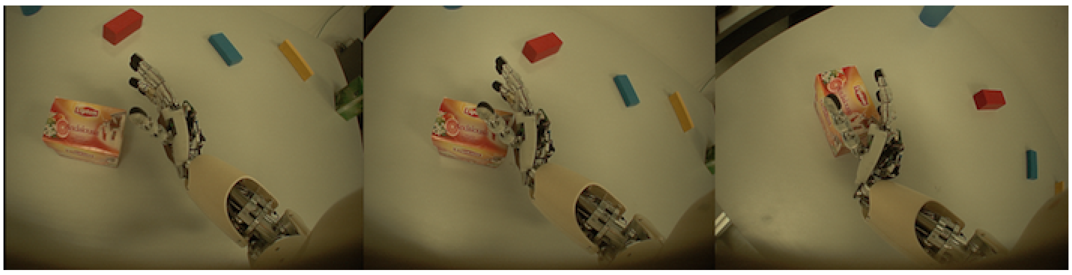


Figure 5.7. An example of a series of images collected during a `PUSH` action.

image is selected. The objects are placed at fixed positions to be within these established masks.

Learning a Simple Model Using CGP-IP

Analogue to the experiments in Section 3.4 the first detector is created from a single starting image.

During these experiments an area of interest in the camera image is defined and the objects are planed at fixed positions to be within this mask. This is used as training set to build the visual object representation. The mask does not precisely segment the image from the background, but highlights the area in the image the object is visible. The exact segmentation of the object is part of the first learning phase. Herein we use a fixed mask and manually place the object to be within. In the future this preliminary mask could very easily be generated from stereo vision information (e.g. [Leitner et al., 2008]). For the creation of the first detector the robot is static. Yet to not simply detect objects on very simple visual cues, e.g. colour in the set-up blue cup vs. red tea box, a variety of objects (kids toy blocks) with different colours are scattered on the table and visible in the scene (in the background in Figure 5.9). We use just one input image with a fixed mask for all objects as training set to start with.

Though only one training image is used, and the mask is not very accurate,

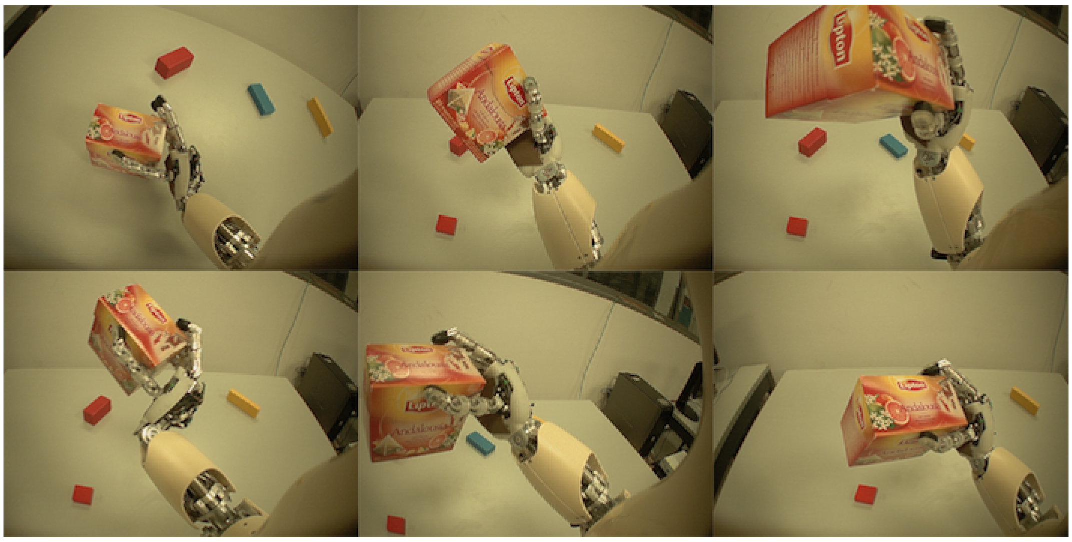


Figure 5.8. An example of a series of images collected during a `CURIOUS` action, which enables the *iCub* to closer inspect the object.

our CGP-IP approach learns to segment and detect the object quite accurately. The simple mask used for all objects does not specify the detailed outline of the item, nevertheless the detectors manage to come close to a precise segmentation. A specific detector is trained for each object individually therefore allowing identification as well.

The learned detection model for the visually rather simple blue cup object (see Figure 5.4) is used as a representative example here. The object representation, converted into executable code is shown in Listing 5.1. The representation also allows for a skilled engineer to understand and manually improve the code. The solution was found after only 1214 individuals were evaluated, taking a few seconds on a standard desktop computer. The detection is shown in the middle of Figure 5.9, where the binary segmentation is used as a red overlay in the input image. The execution of the code takes 140*ms* for a 320×240 pixel image.

The resulting fitness values from this first experiments are not particularly high. There are multiple reasons for this, first the mask or predefined segmentation is not very accurate, due to fact that it is the same mask for every object. Secondly the CGP approach is limited in its training time (to around 20k evaluations). This limit is chosen as to avoid over-fitting to the single input image available.

To show this we ran another experiment for the red tea box for over 2.6m individuals. The found solution was very fit with $f = 0.06$. To achieve this the detector tried to artificially increase the found area to match the input mask as precisely as possible. This is in comparison to another solution evaluating only about 22.7k individuals to find a detector with $f = 0.27$ (see Table 5.1 for more details). The runtime is the *ms* needed for the execution of the detector on a 320×240 colour image. Accuracy refers to the correct detection of the object on an unseen test set of 15 images.

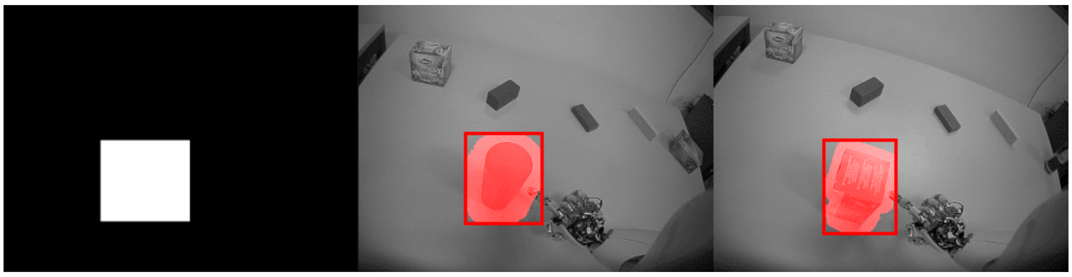


Figure 5.9. The mask (left) used as input to the supervised training, next to two frames showing learned detection (as red overlay) for two distinct objects, the blue cup and the green tea box respectively.

Table 5.1. Comparing the various visual detection models derived from manipulating the objects using a set of actions.

Detector	Fitness f	Individuals tested	Runtime	Accuracy
BlueCup Start	0.28	1214	140.69	100%
BlueCup `LEAN`	0.31	1835	234.03	100%
BlueCup `POKE`	0.18	1214	5.04	100%
BlueCup `PUSH`	0.39	12110	162.75	93%
BlueCup `CURIOUS`	0.42	2445	156.34	100%
GreenTbox Start	0.18	10122	73.08	93%
GreenTbox `LEAN`	0.29	3524	64.00	100%
GreenTbox `POKE`	0.28	1432	73.11	100%
GreenTbox `PUSH`	0.26	1374	119.29	100%
GreenTbox `CUR.`	0.33	3678	71.12	100%
RedTeabox Start	0.27	22697	121.05	100%
RedTeabox `LEAN`	0.45	1426	141.40	100%
RedTeabox `POKE`	0.35	2090	70.11	87%
RedTeabox `PUSH`	0.45	2011	53.96	100%
RedTeabox `CUR.`	0.36	738	114.45	100%
SodaCan Start	0.46	1882	107.79	60%
SodaCan `LEAN`	0.38	6581	140.62	67%
SodaCan `POKE`	0.68	3673	3.68	87%
SodaCan `PUSH`	0.38	1049	223.28	80%
SodaCan `CUR.`	0.31	16078	213.50	87%

Listing 5.1. The generated C++ code from the first learned object representation for the blue cup. This detector is rather simple. Although it detects the blue cup in all test images it has also a few false positives, due to its simplicity. `icImage` is again the wrapper class for the OpenCV functionality.

```
icImage BlueCupFilter::RunFilter() {
    icImage *node43 = InputImages[4];
    icImage *node49 = node43->LocalAvg(15);
    icImage *out = node49->threshold(81.53244f);
    return out;
}
```

Learning Better Models Trough Interaction

To improve the visual models for the object detection the robot can choose to perform an action. The images collected during these interactions are used for learning a better detector. The learning experiment is started by placing the object to be learned in a specific position, as in the first experiment. Similar to the first experiment a single input image is collected to generate a preliminary detector. After this the robot selects one out of four possible actions described above.

The robot observes the object while performing the action. As described above, images in fixed intervals are collected. From these new observations – masks are provided by hand for the supervised learning step – together with the image from the start, a new object representation is learned. Depending on the action and its duration the number of images collected varies. The `LEAN` action, being the shortest, allows only for the collection of 3 new, different images, whereas the `CURIOUS` action can be used to collect 12 images.

Trials are performed for each combination of one object and an action and separate detectors are trained. Table 5.1 shows the learning and performance details of the various detectors. Learning in CGP-IP, like in other evolutionary methods, is non-deterministic, therefore the results shown are based on the best out of five runs. For each of these detectors the fitness during training and the number of individuals evaluated are reported.

The runtime reported is the average of three runs over the 15 images. These images have been collected separately and build a validation set, as they have not been seen during training. The accuracy reported in the last column refers to this training set. It specifies the number of times the object was detected in those images. From this performance we can conclude the 'right' action to chose. For all of the four objects in this experiment the `CURIOUS` action is the best choice for improving the detector. This is not too surprising, as this action allows to perceive the object from various viewing angles.

Even for objects like the blue cup, where detection should be easy, the action allows for an improvement in the detector. There is no increase in the number of times the object is detected, but there is a signification increase on how much of the object is detected. An increase is also visible in the precision of the segmentation, i.e. the improved detector finds matches that are very close to the contours of the objects. The improvement after the `CURIOUS` action is visible in Figure 5.10. The figure also shows the vanishing of the false positive of the red tea box (a red block in the camera image, visible on the table in the background). The changes of the object representation, after observing the execution

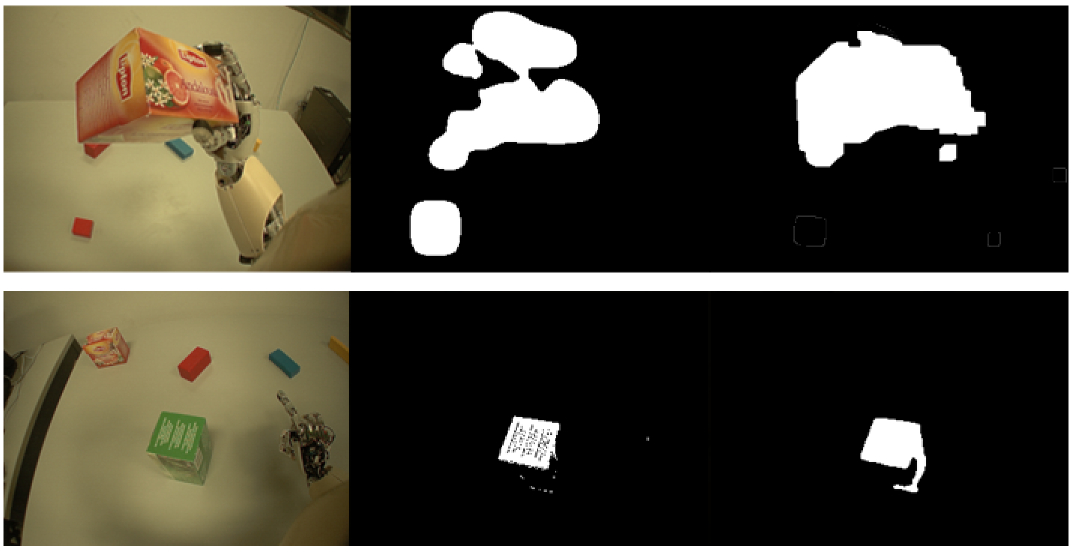


Figure 5.10. Comparing the generated segmentation of an input image (left), using the detector at the start — using one image (middle), with the detector trained after the `CURIOUS` action (right). The first row shows an example from the red tea box the second from the green.

Listing 5.2. The generated C++ code for detecting the blue cup after performing the `CURIOUS` action. Compared to the detector in Listing 5.1, this more complicated model reduces the number of false positives to only 1 in the 15 images of the test set.

```
icImage BlueCupFilter::RunFilter() {
    icImage node0 = InputImages[4].Exp();
    icImage node5 = InputImages[0];
    icImage node16 = node0.Gabor(-8, 14, 1, 13);
    icImage node17 = InputImages[4].LocalAvg(6);
    icImage node18 = node16.Laplace(5);
    icImage node19 = node5.Sobel(13,9);
    icImage node24 = node17.Erode(5);
    icImage node28 = node19.Min(node18);
    icImage node29 = node28.Min(node24);
    icImage node41 = node29.LocalAvg(7);
    icImage node49 = node41.LocalMax(7);
    icImage out = node49.Threshold(68.03109f);
    return out;
}
```

of a CURIOUS action, can be seen in the listing shown in Listing 5.2 (compared to the code above in Listing 5.1).

The detection of the soda can, made of quite reflective aluminium, is not as good as for the other objects. One issue here is the visual similarity of the material with the fingers and other parts of the humanoid's body. This issue is reinforced by the use of simple masks. The fingers are quite often within the masked area, especially during the CURIOUS action. One possibility we are investigating for future research is the use of a disparity map, optical flow or similar approaches to help generating the first masks. Another idea is to use the model learned in one action as the mask for another data collection.

Once the models are learned they can be used on the *iCub* to detect objects in the environment. The system can be fully integrated in the currently available frameworks available in the *iCub* community. Running the detector for both 'eyes' the object's location can be determined [Pattacini, 2011; Leitner et al., 2012f] to update the robot's world model [Leitner et al., 2012a]. By combining all these the *iCub* is able to learn object representations of unseen items, then localize and plan around them.

The *iCub* humanoid robot created object representations using a machine learning approach to computer vision. By interacting with the objects, the robot was able to further improve its object detection and identification skills. It did so by collecting observations during action execution. These new observations allowed to learn a better object detection model. An advantage of our model is that it can be directly mapped into human read-able source code and instantly be compiled to run on the real hardware.

Furthermore, the system was demonstrated to learn how to select the right action, i.e. the action leading to the largest improvement in detection. The experiments show that our CURIOUS action, which contains a pick-up and a variety of rotations to inspect the object, allows for the best improvement. During this action the object can be viewed from almost every angle allowing to build a robust model. This has been observed to be especially useful for visually complex objects, e.g. a tea box.[2]

Although we have a limited number of actions, the results show that performing these actions in connection with learning system for vision allows for more information gained from the environment. We believe that in the future a better sensorimotor coordination can be achieved using this approach. We are especially interested in evaluating possible machine learning techniques, e.g. re-

[2]A video of the experiments is available at: http://Juxi.net/projects/iCub/#wcci2014 or http://www.youtube.com/watch?v=nKlk4mXci_c

inforcement learning, to learn the best possible action directly on the robot. In the future it would be nice to extend the number of actions and their granularity and their robustness, e.g. in non-static environments.

Recap

Based on the description of the perception and actions sides, Chapter 5 details the integration of those subsystems. The advantage of the integration of *icVision* with *MoBeE* is due to abstracting the objects and the robots, allowing minor changes in the environment to be reactively controlling the *iCub* without the need for re-planning. The reactive controller allows a more reactive reaching and grasping. Previously this was done on a variety of simpler robots [DeSouza and Kak, 2002] but not on highly-complex, high-DOF robots, such as the iCub.

Experiments show how the visual detection can be used to improve the actions, especially to create a reactive reaching capability, enabling the robot to avoid dynamic obstacles while reaching for objects. The second part of the integration experiments show how through interaction, based on a set of pre-defined actions, the robot can learn a better visual representations. The ability to change the environment and create novel observations in turn leads to better classification of the objects in the scene.

The integration of the visual frameworks with the action side leads to novel capabilities on the *iCub*; an integrated system now exists allowing for:

- *Eye-Hand Coordination*: my system allows for the detection of arbitrary objects in the workspace and the safe execution of arbitrary motions around these objects.
- *Improving Vision Through Interaction*: building on *CGP-IP* and *icVision* the system is flexible enough to allow the improved detection based on new observations. These new observations can be coming from pre-scripted or learned (with TRM) motions, that are reasonable safe to execute with *MoBeE* providing collision avoidance.

Chapter 6

Conclusions

In this dissertation I present my research towards more autonomous, more adaptive humanoid robots. As mentioned in the introduction creating robots that can 'see' the world is of importance to the robotics community. A major novel contribution is the development of a framework for visual perception that can easily be connected with the control and action side of the robot, therefore creating a tightly integrated system on the *iCub* that did not exist before. This system, consisting of the *icVision*, *CGP-IP* and *MoBeE* frameworks, allows for the detection of arbitrary objects in the workspace (i.e. open-world assumption) and the safe execution of arbitrary motions around these objects. It enables the robot to perform pick-and-place tasks, which in return can improve its own visual perception models through this interaction. The integrated system I created enables the robot to adapt to changes in the environment and safeguards the *iCub* from unwanted interactions, i.e. collisions with the environment or with itself. By integrating the visual system with the motor side and the implementation of an attractor dynamic/virtual force field technique based on the world model, a level of eye-hand coordination not previously seen on the *iCub* is now possible.

In Chapter 3, a learning technique for visual object detection is proposed. Cartesian Genetic Programming for Image Processing (CGP-IP) yields nice results in several different domains with a focus on robotic applications – computer vision and robotic vision have the tendency to look at these problems from slightly different angles. The results themselves indicate that *CGP-IP* is working very well and seem to be competitive with other state-of-the-art learning methods for robotic vision applications. *CGP-IP* is based on well-known image processing operations and generate human readable programs and requires only very limited training sets (all the experiments herein use between 5 and 20 images only). In combination with *icVision* it is not restricted to a closed world assumption used in other dataset-driven approaches and not limiting yourself to a set number of classes is very important for robotic applications. Available computer vision datasets tend to focus on classification of general object types, not identify specific objects. They also focus more single classifications, whereas in robotics more cluttered environments are of interest.[1] Using OpenCV as the basis for the implementation also enables *CGP-IP* to evolve high-speed programs. This, together with the rather small training sets required compared to other approaches, such as neural networks, make it a good candidate to be used in industrial scenarios.

Although it was not known at the beginning — nor was it an aim to build such a system – the ability of *CGP-IP* to work with poorly labelled data is obviously beneficial. In the steel defect dataset (Section 3.4.2) in particular the poor quality of the labelling did not prevent the system from successfully evolving solutions. Unfortunately this means it is quite hard to quantify the quality of the results, as there is no pixel-perfect ground truth (and in fact in most cases, there will be no or subjective differences in any available "ground truth").

Furthermore a learning technique for spatial perception is also presented in Chapter 3. It is shown that satisfactory results can be obtained for localization even in scenarios where the kinematic model is imprecise or not available, and is possible without the need of any lengthy camera calibration procedure. Our humanoid robot learns to provide estimates of object positions placed on a table and in 3D, even while the robot is moving its torso, head and eyes. The outputs, provided by trained artificial neural networks (ANN) and a genetic programming (GP) method, are based solely on the inputs from the two cameras and the joint encoder positions. It was found that ANN and GP are both able to localize objects robustly regardless of the robot's pose and without an explicit kinematic model or camera calibration. These approaches yield an accuracy comparable to current techniques used on the *iCub*. A method is presented, which allows for the automatic collection of the dataset using another robot. This novel way of having two robots acting in the same workspace, while being independently controlled, can be accomplished safely by using *MoBeE*.

While learning spatial perception in Cartesian coordinates is not necessarily the best option – humans are also not good at precisely determining distances of perceived objects – it enables the use of currently existing operational space controllers on the *iCub*. Again while satisfactory results are achieved with this proof-of-concept, a more thorough experimental testing on the *iCub* will need to be conducted in the future.

[1]The 2014 version of the ImageNet dataset now contains more images similar to scenes perceived in robotic settings.

Chapter 4 describes in some detail how motions can be generated for the *iCub* and how we make sure these motions can be executed safely and without collisions on our complex humanoid. It presents *MoBeE* developed over the last years under the lead of Mikhail Frank and how it is a prerequisite to perform tele-operated control with noise bio-signals, or indeed, use machine learning techniques – especially reinforcement learning, which requires repetition of trials without breaking the robot – on the real hardware. The chapter also contains some of the work performed in the frame of two other EU projects at IDSIA. Both of them aim at creating a better understanding of how humans control their motions and how this knowledge can be transferred to robotics (and control). Demonstrations and experiments for controlling the *iCub* remotely with a gamepad, EMG signals and a Leap Motion sensor are described.

Chapter 5 revisits the integration issue and our approach as sketched at the start. It highlights how the separate developments at IDSIA over the last years fit together into a single processing 'pipeline' enabling a eye-hand coordination on the *iCub*, not previously seen. Reactive reaching is possible, wherein the robot is able to avoid all detected objects. It was shown that in a dynamic environment objects can 'push' the robot's arm away from colliding poses. The final experiment shows that by having this sensorimotor coordination, the robot can now exploit this to improve its visual perception skills. It does so by using a set of actions to interact with the world around it. By doing so the scene changes and different training images can be collected and used to learn better classifiers using *CGP-IP*.

Future Work

The research proposed herein and the experiments performed yielded some interesting result. Based on these various future research avenues are possible. From a high-level perspective it would be interesting to further integrate machine learning to extend the object manipulation capabilities of robotic systems. In particular improving the prediction of what is happening in the environment will lead to a more adaptive, versatile selection of actions for the robot to execute. The following should be seen as a starting point for discussion on how one could continue this research.

Object Detection

Although CGP-IP works well on the problems presented here, there is still room for improvement. One major omission here is multi-class classification. Combining CGP-IP with features from MT-CGP [Harding et al., 2012], should solve this. A more thorough analysis of the CGP-IP performance, especially when compared with neural networks on classification problems, will be performed in the future. CGP-IP should also benefit from a GPU based implementation.

One could also imagine to continue the work presented in Sections 3.4.5 and 5.3.3, and aim to build systems that are able to improve their perception skills over their whole lifetime. This life-long learning, especially for deployed robotic systems, is increasingly interesting to researchers in both robotics and machine learning. This could also be facilitated by the use of committees of evolved filters, that are each specializing on detecting the object in specific settings. The filters could even be used in a hierarchical order to closer represent what we belief the brain does. This grouping of filters, leading to some sort of committee voting, might allow for novel approaches to robust detection.

It would also be interesting, as suggested by members of the audiences at multiple conferences, to see if one could evolve such detectors with more than just visual information. Of interest might be to have also tactile representations (from interaction), or geometric models that are being evolved, maybe in parallel, to help with a more precise classification of objects. For the applied settings it would be interesting to see what can be done to allow a more flexible, simpler transfer of the learned detectors between different setups, e.g. would it be possible to transfer the learned mars detectors to the camera setup of the second Mars Exploration Rover (MER) or even to Mars Science Laboratory (MSL/Curiostiy)?

Generally of interest would be to extend the frameworks further to deal with human-robot interaction scenarios. For this a better detection of humans and also improved safety requirements might have to be taken into account. Additionally it might make sense to allow the fusion of other available sensors, especially the force/torque sensors in the upper arm of the *iCub*.

Spatial Perception

A better dataset and collection thereof might lead to better results when learning to predict the location of an object. Especially the various stark outliers seen in the 3D results of Section 3.6.3 make a case for this. One of the challenges here will be in the process of collecting the data. Different methods for simplifying the data collection will be required. One possibility is to determine how few data points are actually needed for the various learning methods to estimate object positions precise enough for manipulation. Fewer, but better, data points will allow to reduce the time needed to collect the dataset on a different robot or when parameters are changing, for example, replacing of parts, such as the cameras.

Alternatively, it may be possible to turn the process into an incremental learning problem. In this case the robot would be learning to improve its predictions of the positions while performing actions, this would be closer to how human sensorimotor skills are believed to develop. Extending this thought further a robot knowing its perceptual precision in various parts of the environment could be lead by artificial curiosity [Frank et al., 2014; Schmidhuber, 1991] to ask for more data points in those areas with, not just the highest localization error, but the greatest improvement of localization error and aim to improve its capabilities specifically in those regions.

In addition the method described can learn whether an object visible in both cameras is within the reachable workspace of the robot. Assuming that the maximum reachable space is limited by the robot's arm, it seems possible to train an ANN to predict whether an object in the scene is reachable given the current robot body pose. A simple modification in the training set, to contain, instead of the 3D coordinates in p, only a binary output, whether the point is reachable or not, e.g. every point further away than 60 cm, could be used for a simple proof-of-concept. In this simplified problem one does not predict though if there is a feasible joint configuration for this point. The ANN for this task, which would have the same model as each of the ANNs before, can again be trained using error back-propagation. Another goal is to extend this work to detection of the object's orientation. In fact we already collect 6 inputs from vision per image (centre of blob, as well as, the location and size of the bounding box), which could enable the estimation of the objects orientation, if the detected object's geometry is known.

A different approach that might also be interesting to pursue is Structure-from-Motion (SfM). It has been shown to work quite nicely in flying robots and movable camera setups. In connection with *MoBeE* and the moving *iCub* a rough world model of the surroundings could be generated, reducing the dependency on loaded geometries for objects in the scene.

Integration

Furthermore it might be of interest to investigate an even tighter sensorimotor coupling, e.g. avoiding translation into operational space by working in vision/-configuration space. One could also imagine to have an iterative approach to learn more and more about world. First a simple means of detection would be used (e.g., as in Section 3.4.5, or a simple disparity map). Then simple actions, prepared for example with TRM, could be performed to learn more and better classifiers, which would allow to perform more complicated interactions with the object.

Visual Servoing (VS) is a commonly used approach to closed-loop vision based control. It refers to the use of computer vision data to control the motion of a robot [Chaumette and Hutchinson, 2006, 2007]. It relies on techniques from image processing, computer vision, and control theory. The visual information is usually coming from a camera mounted on a robot manipulator or a mobile robot. Various configurations of the robot and camera(s) are possible, the most common in literature being the eye-in-hand case. With human-style robots, VS has only been investigated recently. The issue here is that it is not a eye-in-hand nor a eye-to-hand system, as the camera is on the robot, but only some of the reaching movements will imply a motion also with the head (this only happens when the hip is moved). Generally there are two separate approaches: a position-based VS (PBVS) and an image-based VS (IBVS). There is no definitive answer which ones is better, PBVS tends to be used more in setups with a 3D sensor (e.g. LIDAR, Kinect, etc.) whereas IBVS seems to be preferential when using cameras, like in our case.

It would be interesting to extend the system we currently have, which is similar to a PBVS, to an image-based VS setup. The foundations, object and hand detection in the images, are there, so is a simple forcing mechanism. Due to time constraints a full integration and more importantly experiment and test was not been performed so far. With our current setup we should be able to measure the error between the hand and the object we want to pick up, similar to previous approaches on less complex robots or with the help of external sensors [Hager et al., 1995].

Tele-operation of Complex Robots A proof-of-concept for tele-operating a complex humanoid safely was shown in Section 4.5. Future research could investigate closer how more sophisticated bio-signal processing can be included and applied to control the robot. Furthermore a significant amount of user testing will need to be done to further improve the operator interface and create really useful systems. Especially the EMG and machine learning side can still be improved to include multiple grasp types and reduce the amount of learning required for each operator. We are confident that this can also be transferred to other scenarios, such as, prosthetics control by amputees. Different collecting methods, different number of channels, and instruments have so already been used to collect new datasets for a variety of grasp types.

The most interesting problem to solve will be the robust detection of the signal while the operator is moving her/his arm. It might contain artefacts and high levels of noise stemming from other muscle activations (e.g. to counteract gravity and the dynamics of the motion).

In the light of current news coverage about the advancement of robotics and automation, in various application areas, from autonomous cars to military use, one should be careful not to overstate what robots can currently do and what they will be able to do in the future. The “jobpocalypse”, i.e. the loss of jobs due to the increasing number of robots and the progression of automation[2], also is covered vividly in the news. While one can rightfully be sceptic, it might though be time to start thinking about the implications of robotic efforts on society, or at least think about how robotics is perceived by the public and in the media.

[2]"It would be irresponsible to predict that “jobpocalypse” is around the corner. The T-800 is unlikely to replace dentists, rabbis and gym instructors: a recent Oxford University study [Frey and Osborne, 2013] found that ‘only’ 47 per cent of US employment is at risk of computerization within the next decade or two. This will relieve those who already have trouble sleeping in an age of proliferating doom scenarios. " taken from: `http://www.newstatesman.com/economics/2014/03/learning-live-machines`

Appendix A

CGP-IP Function Set

Table A.1. The CGP-IP function set, part I.

Name	Arity	Description
NOP	1	No Operation, returns first input into the node
INP	0	Returns the current `ImageInput`, move pointer forward
INPP	0	Moves pointer backwards, returns input
SKIP	0	Moves pointer by (int)`Parameter0`, returns input
add	2	Adds two images together
sub	2	Subtracts one image from another
const	0	Makes a new image, all pixels set to Parameter0
mul	2	Multiplies two images together
addc	1	Adds Parameter0 to each pixel in the input image
subc	1	Subtracts Parameter0 from each pixel in the input image
mulc	1	Multiplies each pixel in the input image by Parameter0
log	1	Performs log each pixel in the input image
exp	1	Performs exp on each pixel in the input image
sqrt	1	Performs $\sqrt{}$ on each pixel in the input image
absDiff	2	Finds the absolute difference between two images
avg	2	Finds the per pixel average of two images
max	2	Returns the maximum of each pixel pair in the two input images
min	2	Returns the minimum of each pixel pair in the two input images

Table A.2. The CGP-IP function set, part II. Functions that operate on a neighbourhood, can be expected to use a size of `Parameter1` by `Parameter2`. Note: all functions in this table have an arity of 1.

Name	Description
threshold	Binary threshold the image at Parameter0
thresholdbw	Binary threshold the image at 64
normalize	Normalize an image in the range 0 to 255
laplace	Calculates Laplacian of an image
gauss	Performs Gaussian blur
gauss2	Performs Gaussian blur, with a kernel sized Parameter1 by Parameter2
sobelx	Performs Sobel edge detection in the X direction
sobely	Performs Sobel edge detection in the Y direction
sobel	Performs Sobel using Parameter1 and 2 to provide the X and Y orders
smoothMedian	Performs the smooth median filter
smoothBilateral	Performs the smooth bilateral filter
smoothBlur	Performs the smooth blur filter
unsharpen	Performs the unsharpen operation
shift (0,-1)	Circular shifts the image down
shift (0,1)	Circular shifts the image up
shift (-1,0)	Circular shifts the image to the left
shift (-1,0)	Circular shifts the image to the right
shift (Parameter1, 0)	Circular shifts the image `Parameter1` pixels horizontally
shift (0, Parameter2)	Circular shifts the image `Parameter2` pixels vertically
shift (Parameter1, Parameter2)	Circular shifts the image pixel values in a given direction
gabor (Orientation, Frequency)	Performs the Gabor operation
gabor (Orientation, Frequency, Parameter1, Parameter2)	Performs the Gabor operation
gabor (Orientation, Frequency, Parameter0)	Down-samples the image, and then performs the Gabor operation
minValue	Creates an image where every pixel is the minimum pixel value in the input image
maxValue	Creates an image where every pixel is the maximum pixel value in the input image
avgValue	Creates an image where every pixel is the average pixel value in the input image

Table A.3. The CGP-IP function set, part III. Functions that operate on a neighbourhood, can be expected to use a size of `Parameter1` by `Parameter2`. Note: all functions in this table have again an arity of 1.

Name	Description
reScale (Parameter0)	Down-samples the input image by a factor of `Parameter0`
min (Parameter0)	Compares each pixel to `Parameter0`
max (Parameter0)	Compares each pixel to `Parameter0`
localNormalize	Performs a local normalization of the input image
localMin	Computes the local minimum for each pixel in the input
localMax	Computes the local maximum for each pixel in the input
localAvg	Computes the local average for each pixel in the input
erode	Performs the erosion operator
dilate	Performs the dilation operator
canny	Performs Canny edge detection

Appendix B

CGP-IP Additional Results: Medical Imaging

B.1 Medical Imaging: Cell Mitosis

Detecting, and then counting, cancer cells undergoing mitosis is useful diagnostic measurement in breast cancer screening. However, the cells are small and have a large variety of shapes. This challenge has led to a competition at the 2012 International Conference on Pattern Recognition (ICPR)[1], where entrants are invited to submit methods for solving this problem.

To test the problem with CGP-IP, the training set was sliced in a number of patches (i.e. small sections of the image). Half of the patches contain one or more mitoses to detect, the other half contains randomly selected empty patches. In total 420 images patches were used, with 356 used for training and the remaining 64 reserved for validation. As this is a binary classification problem, the MCC based fitness function was used (i.e. Type B).

Due to time constraints, CGP-IP was run only 6 times. In future work we will perform a more thorough investigation in the performance of CGP-IP on this problem. Statistical results for these runs are shown in Table B.1. Figure B.1 shows the input image, expected and predicted classes for validation images for the best performing individual. These results are based on per-pixel analysis, and suggest that CGP-IP provides excellent segmentation. With additional analysis of the segments, the classification rate of the mitoses can also be determined. Out of 42 mitoses, CGP-IP correctly identified 36 of them i.e. 86% of the mitoses were correctly identified. There were 12 false-positives, and 6 false-negatives.

[1]The competition website: `http://ipal.i2r.a-star.edu.sg/event/icpr-2012`

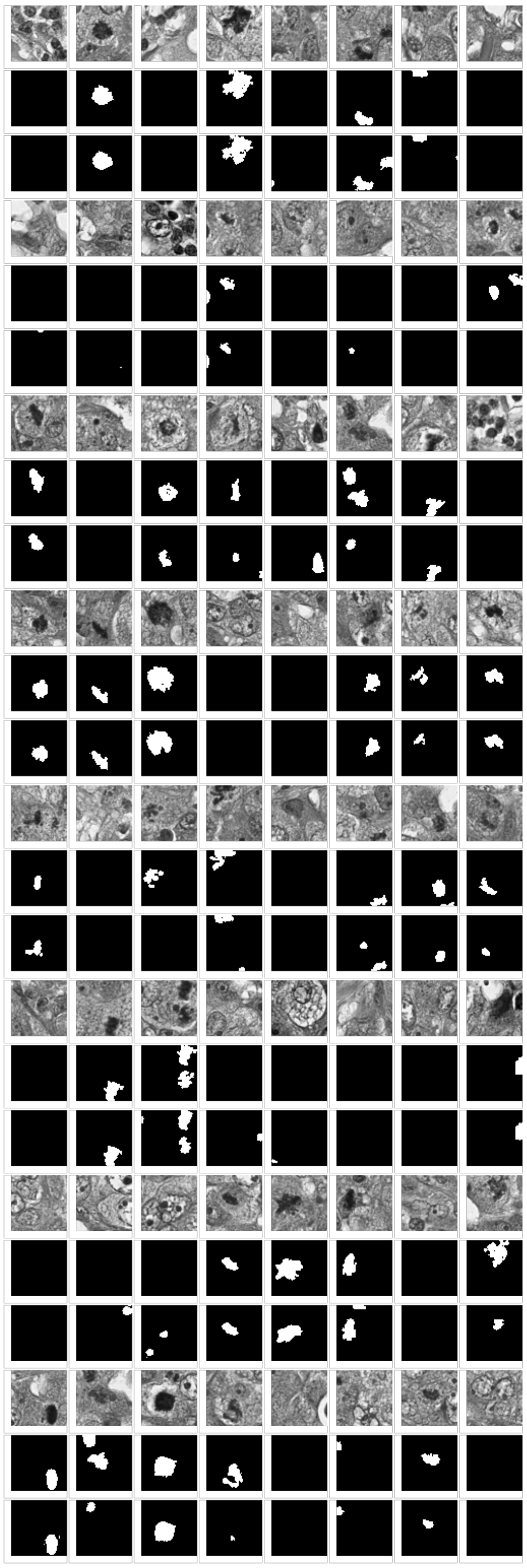


Figure B.1. Validation set for the ICPR Mitosis dataset. The columns with the images show the input images, the next column to the right shows the expected output and the following column shows the predicted classifications.

It will be interesting to compare CGP-IP to other published methods once the competition results are available.

Table B.1. Results for the ICPR dataset. The classification accuracy is per-pixel. The fitness function was uses the MCC score (Type B).

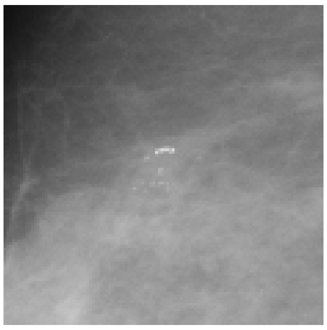
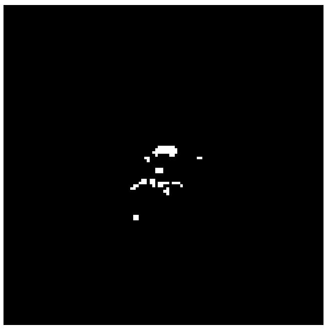
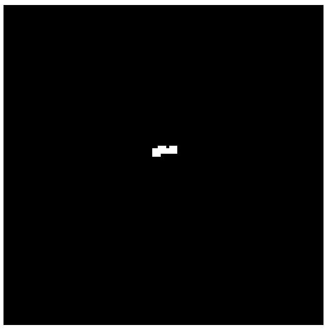
	% Classification accuracy	MCC (Fitness)
Average	98	0.36
Minimum	97	0.28
Maximum	98	0.46
Std. Dev.	0.3	0.08

B.2 Medical Imaging: MIAS Mammographic Database

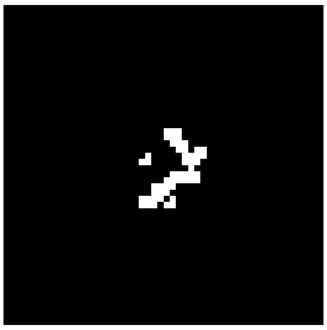
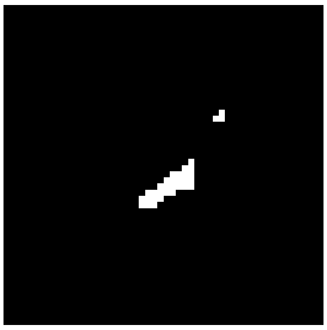
The 'mini-MIAS' database [Suckling et al., 1994] contains 322 labelled X-ray mammogram images. For each image, a number of different *Regions Of Interest* (ROIs) are defined, relating to features such as calcification or spiculated masses. In the original dataset, the ROIs are described by a circle at a given position and with a given radius. However, the ROI is extremely crude, and may only contain a small proportion of the labelled class. Using CGP-IP on these ROIs, we were unable to obtain satisfactory results. As the calcifications are relatively easy to identify (given the provided ROI), a new version of the dataset was produced where the calcifications were labelled on per-pixel basis by hand. It should be noted that this was not performed by a medical expert, so errors in the labelling are expected. Using this set however, it was possible to get CGP-IP to identify the calcified areas.

To build the training set, the images containing calcified areas were identified, and the calcifications relabelled. The original images were then cleaned up to remove artefacts such as text and registration marks. Finally, the images were cropped tightly to the ROI. For training 14 images were used, and 8 images used for validation. Again, as this is a binary classification problem, the MCC based fitness function was used (i.e. Type B).

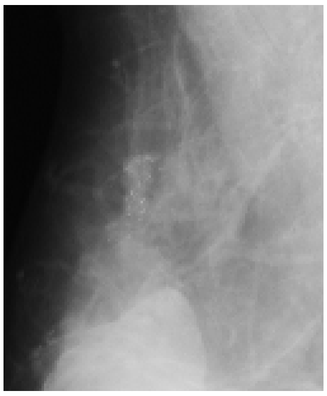
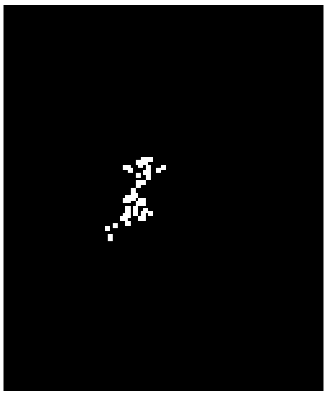
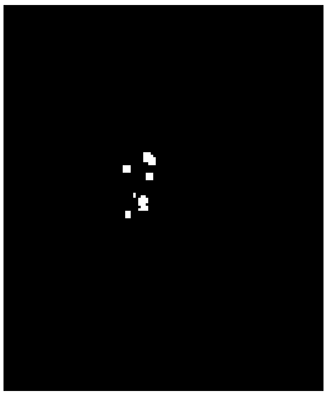
Table B.2 shows the classification rates of the evolved filters. The high classification rate is due to the large imbalance in the classes. The MCC score (used for the fitness), indicates that the filters are not particularly good. However, this is most likely due to poor labelling in the validation data. Figure B.2 show the evolved filters working on patches from the validation set of images.

(a) Input image. *(b)* Expected output. *(c)* Predicted output.

(d) Input image. *(e)* Expected output. *(f)* Predicted output.

(g) Input image. *(h)* Expected output. *(i)* Predicted output.

Figure B.2. Examples of evolved filters from the MIAS dataset.

Table B.2. Results for the MIAS dataset. Classification accuracy is per-pixel. Ten runs were performed. The average number of evaluations to convergence was found to be 21,104 (std. dev. 12,210).

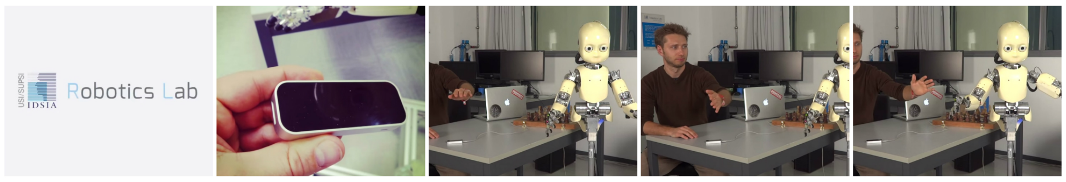
	% Classification Accuracy	MCC (Fitness)
Average	99.3	0.40
Minimum	99.2	0.30
Maximum	99.3	0.46
Std. Dev.	0.06	0.05

Appendix C

IDSIA Robotics Research Videos

During the course of my PhD project work at IDSIA a variety of research videos were created. These show the progress and implementation on the *iCub*. The following is a list of these with links to view them online.

Controlling the iCub with a LEAP sensor: “ The Real 'Real Steel' ” (2014)
`http://www.youtube.com/watch?v=5i6tA0b922g`



Reactive Reaching and Grasping on a Humanoid (2014)
`http://www.youtube.com/watch?v=w_qDH5tSe7g`

the detected object is localised and placed in the world
Grasp
another view of the grasping

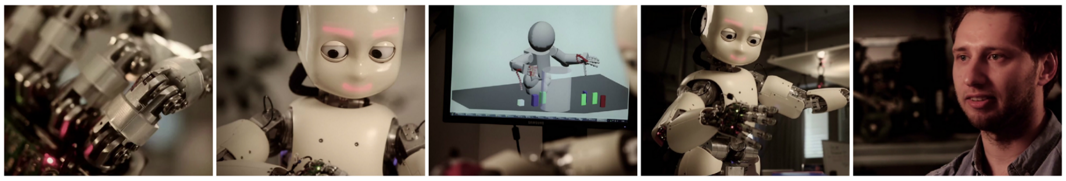
Improving Robot Vision Models for Object Detection (2014)
`http://www.youtube.com/watch?v=nKlk4mXci_c`

autoCGP-IP: Autonomous Learning of Robust Visual Object Detection and Identification (2013)
`http://www.youtube.com/watch?v=gXaa6dwE5Bs`

IDSIA
Robotics Lab
The most salient point is then used to
DisplaySaliency_left
DisplaySaliency_right
by again using the saliency map, v

Task Relevant Road Maps (AAAI Shakey Winner - Best Student Video 2013)
`http://www.youtube.com/watch?v=N6x2e1Zf_yg`

IM-CLeVeR Video: "Toward Intelligent Humanoids" (2012)
`http://vimeo.com/51011081`



IM-CLeVeR Video: icVision (2012)
`http://www.youtube.com/watch?v=xsz0Cj4A1eA`

In addition our robots and research was featured in other videos a full list can be found at: `http://Juxi.net/media`

Appendix D

Acronyms

AAAI	Association for the Advancement of Artificial Intelligence
ADL	Activities of Daily Living
AI	Artificial Intelligence
ANN	Artificial Neural Network
CGP	Cartesian Genetic Programming
CGP-IP	Cartesian Genetic Programming for Image Processing
CV	Computer Vision
DARPA	Defense Advanced Research Projects Agency
DOF	Degrees of Freedom
EMG	Electromyography
GP	Genetic Programming
GPGPU	General-purpose Computing on Graphics Processing Units
GPGPGPU	Genetic Programming On General-purpose Graphics Processing Units
GPL	GNU General Public License
GPU	Graphics Processing Unit
IDSIA	Istituto Dalle Molle di Studi sull'Intelligenza Artificiale

IK Inverse Kinematics

MCC Matthews Correlation Coefficient

MoBeE Modular Behavioural Environment

ML Machine Learning

ROIs Regions Of Interest

TRM Task-relevant Road Map

YARP Yet Another Robotics Platform

Bibliography

Agarwal, S. and Roth, D. [2002]. Learning a sparse representation for object detection, *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 113–130.

Akella, S. and Hutchinson, S. [2002]. Coordinating the motions of multiple robots with specified trajectories, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Vol. 1, pp. 624–631.

Aloimonos, J., Weiss, I. and Bandyopadhyay, A. [1988]. Active vision, *International Journal of Computer Vision* **1**(4): 333–356.

Ambrose, R., Wilcox, B., Reed, B., Matthies, L., Lavery, D. and Korsmeyer, D. [2012]. NASA's Space Technology Roadmaps (STRs): Robotics, tele-robotics, and autonomous systems roadmap, *Technical report*, National Aeronautics and Space Administration (NASA).
URL: `http://www.nasa.gov/pdf/501622main_TA04-ID_rev6b_NRC_wTASR.pdf`

Ampatzis, C., Izzo, D., Ruciński, M. and Biscani, F. [2011]. ALife in the Galapagos: Migration Effects on Neuro-Controller Design, *in* G. Kampis, I. Karsai and E. Szathmáry (eds), *Advances in Artificial Life. Darwin Meets von Neumann*, Vol. 5777 of *Lecture Notes in Computer Science*, Springer, pp. 197–204.

Amzajerdian, F., Pierrottet, D., Petway, L., Hines, G. and Roback, V. [2011]. Lidar systems for precision navigation and safe landing on planetary bodies, *Proceedings of SPIE 8192, International Symposium on Photoelectronic Detection and Imaging*, International Society for Optics and Photonics, p. 819202.

Argall, B. D., Sauser, E. L. and Billard, A. G. [2010]. Tactile guidance for policy refinement and reuse, *Proceedings of the International Conference on Development and Learning (ICDL)*, pp. 7–12.

Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M. and Yoshida, C. [2009]. Cognitive developmental robotics: A survey, *IEEE Transactions Autonomous Mental Development* **1**: 12–34.

Asada, M., MacDorman, K., Ishiguro, H. and Kuniyoshi, Y. [2001]. Cognitive developmental robotics as a new paradigm for the design of humanoid robots, *Robotics and Autonomous Systems* **37**(2): 185–193.

Asimov, I. [1942]. Runaround, *Astounding Science Fiction* **29**(1): 94–103.

Baginski, B. [1999]. *Motion planning for manipulators with many degrees of freedom: the BB-method*, Infix.

Bajracharya, M., Maimone, M. and Helmick, D. [2008]. Autonomy for Mars rovers: Past, present, and future, *Computer* **41**(12): 44–50.

Balch, T. and Arkin, R. C. [1998]. Behavior-based formation control for multi-robot teams, *IEEE Transactions on Robotics and Automation* **14**(6): 926–939.

Baldassare, G., Mirolli, M., Mannella, F., Caligiore, D., Visalberghi, E., Natale, F., Truppa, V., Sabbatini, G., Guglielmelli, E., Keller, F., Campolo, D., Redgrave, P., Gurney, K., Stafford, T., Triesch, J., Weber, C., Rothkopf, C., Nehmzow, U., Condell, J., Siddique, M., Lee, M., Huelse, M., Schmidhuber, J., Gomez, F., Förster, A., Togelius, J. and Barto, A. [2009]. The IM-CLeVeR project: Intrinsically motivated cumulative learning versatile robots, *Proceedings of the International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*.

Bay, H., Tuytelaars, T. and Van Gool, L. [2006]. SURF: Speeded up robust features, *in* A. Leonardis, H. Bischof and A. Pinz (eds), *Computer Vision – ECCV 2006*, Vol. 3951 of *Lecture Notes in Computer Science*, Springer, pp. 404–417.

Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mösenlechner, L., Pangercic, D., Rühr, T. and Tenorth, M. [2011]. Robotic roommates making pancakes, *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia.

Berenson, D., Srinivasa, S., Ferguson, D. and Kuffner, J. [2009]. Manipulation planning on constraint manifolds, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 625–632.

Berenson, D., Srinivasa, S. and Kuffner, J. [2011]. Task space regions a framework for pose-constrained manipulation planning, *The International Journal of Robotics Research* **30**(12): 1435–1460.

Berthier, N., Clifton, R., Gullapalli, V., McCall, D. and Robin, D. [1996]. Visual information and object size in the control of reaching, *Journal of Motor Behavior* **28**(3): 187–197.

Billard, A., Bonfiglio, A., Cannata, G., Cosseddu, P., Dahl, T., Dautenhahn, K., Mastrogiovanni, F., Metta, G., Natale, L., Robins, B., Seminara, L. and Valle, M. [2013]. The ROBOSKIN Project: Challenges and Results, *in* V. Padois, P. Bidaud and O. Khatib (eds), *Romansy 19 – Robot Design, Dynamics and Control*, Vol. 544 of *CISM International Centre for Mechanical Sciences*, Springer Vienna, pp. 351–358.

Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E., Goza, M., Rehnmark, F., Lovchik, C. and Magruder, D. [2003]. Robonaut: A robot designed to work with humans in space, *Autonomous Robots* **14**(2): 179–197.

Bongard, J., Zykov, V. and Lipson, H. [2006]. Resilient machines through continuous self-modeling, *Science* **314**(5802): 1118–1121.

Borenstein, J. and Koren, Y. [1989]. Real-time obstacle avoidance for fast mobile robots, *IEEE Transactions on Systems, Man and Cybernetics* **19**(5): 1179–1187.

Bouguet, J.-Y. [2014]. Camera Calibration Toolbox for Matlab, `http://www.vision.caltech.edu/bouguetj/calib_doc/`. [Online; accessed 24-July-2014].

Bradski, G. [2000]. The OpenCV Library, *Dr. Dobb's Journal of Software Tools* .

Brooks, R. [1991]. Intelligence without representation, *Artificial intelligence* **47**(1): 139–159.

Brooks, R. [1999]. *Cambrian intelligence: the early history of the new AI*, The MIT Press.

Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M., Lipson, H. and Jaeger, H. [2010]. Universal robotic gripper based on the jamming of granular material, *Proceedings of the National Academy of Sciences (PNAS)* **107**(44): 18809–18814.

Bryson, A. E. [1961]. A gradient method for optimizing multi-stage allocation processes, *Proc. Harvard Univ. Symposium on digital computers and their applications*.

Bryson, A. E. and Ho, Y. C. [1969]. *Applied optimal control: optimization, estimation and control*, CRC Press.

Burgard, W., Moors, M., Stachniss, C. and Schneider, F. [2005]. Coordinated multi-robot exploration, *IEEE Transactions on Robotics* **21**(3): 376–386.

Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M. and Baldassarre, G. [2008]. Using motor babbling and Hebb rules for modeling the development of reaching with obstacles and grasping, *Proceedings of the International Conference on Cognitive Systems (CogSys)*.

Cangelosi, A., Belpaeme, T., Sandini, G., Metta, G., Fadiga, L., Sagerer, G., Rohlfing, K., Wrede, B., Nolfi, S., Parisi, D., Nehaniv, C., Dautenhahn, K., Saunders, J., Fischer, K., Tani, J. and Roy, D. [2008]. The ITALK project: Integration and transfer of action and language knowledge in robots, *Proceedings of the International Conference on Human Robot Interaction (HRI)*.

Carbone, G. [2013]. *Grasping in robotics*, Vol. 10 of *Mechanisms and Machine Science*, Springer.

Castano, A., Fukunaga, A., Biesiadecki, J., Neakrase, L., Whelley, P., Greeley, R., Lemmon, M., Castano, R. and Chien, S. [2008]. Automatic detection of dust devils and clouds on mars, *Machine Vision and Applications* **19**(5): 467–482.

Chaumette, F. and Hutchinson, S. [2006]. Visual servo control, Part I: Basic approaches, *IEEE Robotics & Automation Magazine* **13**(4): 82–90.

Chaumette, F. and Hutchinson, S. [2007]. Visual servo control, Part II: Advanced approaches, *IEEE Robotics and Automation Magazine* **14**(1): 109–118.

Chella, A., Frixione, M. and Gaglio, S. [2008]. A cognitive architecture for robot self-consciousness, *Artificial Intelligence in Medicine* **44**(2): 147–154.

Cheng, N., Lobovsky, M., Keating, S., Setapen, A., Gero, K., Hosoi, A. and Iagnemma, K. [2012]. Design and analysis of a robust, low-cost, highly articulated manipulator enabled by jamming of granular media, *Proceedings of the International Conference of Robotics and Automation (ICRA)*, pp. 4328–4333.

Chu, P. C. [2011]. SMART Underwater Robot (SUR) Application & Mining. Technical Presentation.
URL: `http://faculty.nps.edu/pcchu/web_paper/conference/11/wof_2011_chu.pdf`

Ciliberto, C., Smeraldi, F., Natale, L. and Metta, G. [2011a]. Online multiple instance learning applied to hand detection in a humanoid robot, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Ciliberto, C., Smeraldi, F., Natale, L. and Metta, G. [2011b]. Online multiple instance learning applied to hand detection in a humanoid robot, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 1526–1532.

Cipolla, R., Battiato, S. and Farinella, G. M. [2010]. *Computer Vision: Detection, Recognition and Reconstruction*, Vol. 285, Springer.

Cipriani, C., Controzzi, M. and Carrozza, M. C. [2011]. The smarthand transradial prosthesis, *Journal of Neuroengineering and Rehabilitation* **8**(1): 29.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L. and Schmidhuber, J. [2011]. Flexible, high performance convolutional neural networks for image classification, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* **22**(1): 1237.

Cliff, D., Husbands, P. and Harvey, I. [1993]. Explorations in evolutionary robotics, *Adaptive Behavior* **2**(1): 73–110.

Corke, P. [2011]. *Robotics, Vision and Control*, Vol. 73 of *Springer Tracts in Advanced Robotics*, Springer.

Cramer, N. L. [1985]. A representation for the adaptive generation of simple sequential programs, *Proceedings of the First International Conference on Genetic Algorithms*, pp. 183–187.

Cutler, R. and Turk, M. [1998]. View-based interpretation of real-time optical flow for gesture recognition, *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 416–421.

Dachwald, B. [2004]. Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol, *Journal of Guidance, Control, and Dynamics* **27**(1): 66–72.

Dautenhahn, K. and Saunders, J. [2011]. *New Frontiers in Human-Robot Interaction*, John Benjamins Publishing Company.

Davison, A. J. and Murray, D. W. [2002]. Simultaneous localization and map-building using active vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(7): 865–880.

Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O. [2007]. MonoSLAM: Real-time single camera SLAM, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **29**(6): 1052–1067.

De Santis, A., Albu-Schaffer, A., Ott, C., Siciliano, B. and Hirzinger, G. [2007]. The skeleton algorithm for self-collision avoidance of a humanoid manipulator, *Proceedings of the IEEE/ASME International Conferenced on Advanced Intelligent Mechatronics*.

Denavit, J. and Hartenberg, R. [1955]. A kinematic notation for lower-pair mechanisms based on matrices., *Transactions of the ASME. Journal of Applied Mechanics* **22**: 215–221.

DeSouza, G. N. and Kak, A. C. [2002]. Vision for mobile robot navigation: A survey, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **24**(2): 237–267.

Devol, J. [1961]. Programmed article transfer. US Patent 2,988,237.
URL: `http://www.google.com/patents/US2988237`

Dickinson, S. J., Stevenson, S., Amdur, E., Tsotsos, J. K. and Olsson, L. [1993]. Integrating task-directed planning with reactive object recognition, *Optical Tools for Manufacturing and Advanced Automation*, International Society for Optics and Photonics, pp. 212–224.

Dickmanns, D., Schmidhuber, J. and Winklhofer, A. [1987]. Der genetische Algorithmus: Eine Implementierung in Prolog. Seminararbeit/Technical Report.

Dickmanns, E. D. [1997]. Vehicles capable of dynamic vision, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1577–1592.

Dietrich, A., Wimbock, T., Taubig, H., Albu-Schaffer, A. and Hirzinger, G. [2011]. Extensions to reactive self-collision avoidance for torque and position controlled humanoids, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 3455–3462.

Diftler, M., Mehling, J., Abdallah, M., Radford, N., Bridgwater, L., Sanders, A., Askew, R., Linn, D., Yamokoski, J., Permenter, F., Hargrave, B., Piatt, R., Savely,

R. and Ambrose, R. [2011]. Robonaut 2 - the first humanoid robot in space, *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Doncieux, S., Mouret, J., Bredeche, N. and Padois, V. [2011]. Evolutionary robotics: Exploring new horizons, *New Horizons in Evolutionary Robotics* pp. 3–25.

Dreyfus, S. [1962]. The numerical solution of variational problems, *Journal of Mathematical Analysis and Applications* **5**(1): 30–45.

Duda, R. O. and Hart, P. E. [1972]. Use of the hough transformation to detect lines and curves in pictures, *Communications of the ACM* **15**(1): 11–15.

Edsinger, A. and Kemp, C. [2006]. Manipulation in human environments, *Proceedings of the International Conference on Humanoid Robots*, pp. 102–109.

Ehara, Y., Fujimoto, H., Miyazaki, S., Tanaka, S. and Yamamoto, S. [1995]. Comparison of the performance of 3d camera systems, *Gait & Posture* **3**(3): 166–169.

Fanello, S. R., Ciliberto, C., Natale, L. and Metta, G. [2013]. Weakly supervised strategies for natural object recognition in robotics, *Proceedings of the International. Conference on Robotics and Automation (ICRA)*.

Fausett, L. [1994]. *Fundamentals of neural networks: architectures, algorithms, and applications*, Prentice-Hall.

Fofi, D., Sliwa, T. and Voisin, Y. [2004]. A comparative survey on invisible structured light, *Proc. SPIE* **5303**: 90–98.

Fogel, L. J., Owens, A. J. and Walsh, M. J. [1966]. *Artificial intelligence through simulated evolution*, John Wiley & Sons.

Forlizzi, J. and DiSalvo, C. [2006]. Service robots in the domestic environment: a study of the roomba vacuum in the home, *ACM SIGCHI/SIGART Conference on Human-Robot-Interaction*, pp. 258–265.

Forssberg, H., Eliasson, A., Kinoshita, H., Johansson, R. and Westling, G. [1991]. Development of human precision grip i: basic coordination of force, *Experimental Brain Research* **85**(2): 451–457.

Forsyth, D. and Fleck, M. [1997]. Finding people and animals by guided assembly, *Proceedings of the International Conference on Image Processing*, Vol. 3, pp. 5–8.

Frank, M. [2014]. *Learning To Reach and Reaching To Learn: A Unified Approach to Path Planning and Reactive Control through Reinforcement Learning*, PhD thesis, Universitá della Svizzera Italiana, Lugano.

Frank, M., Förster, A. and Schmidhuber, J. [2012]. Reflexive Collision Response with Virtual Skin - Roadmap Planning Meets Reinforcement Learning, *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*.

Frank, M., Leitner, J., Stollenga, M., Föster, A. and Schmidhuber, J. [2014]. Curiosity driven reinforcement learning for motion planning on humanoids, *Frontiers in Neurorobotics* **7**(25).

Frey, C. B. and Osborne, M. A. [2013]. The Future of Employment: How susceptible are jobs to computerisation?
URL: *http://www.oxfordmartin.ox.ac.uk/publications/view/1314*

Galvani, L. [1791]. *De viribus electricitatis in motu musculari: Commentarius*, Bologna: Tip. Istituto delle Scienze. translated by R. Montraville Green, 1953.
URL: *http://www.liberliber.it/mediateca/libri/r/redi/esperienze_intorno_a_diverse_cose_naturali_etc/pdf/esperi_p.pdf*

Gatsoulis, Y., Burbridge, C. and McGinnity, T. M. [2011]. Online unsupervised cumulative learning for life-long robot operation, *Proceedings of the International Conference on Robotics and Biomimetics (ROBIO)*.

GeRT [2012]. EU Project Consortium: Generalizing Robot Manipulation Tasks, http://www.gert-project.eu/project/the-vision/.

Gharbi, M., Cortés, J. and Siméon, T. [2009]. Roadmap composition for multi-arm systems path planning, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 2471–2476.

Gill, M. and Zomaya, A. [1998]. *Obstacle Avoidance in Multi-Robot Systems: Experiments in Parallel Genetic Algorithms*, Vol. 20, World Scientific Pub Co Inc.

Glasmachers, T., Schaul, T., Yi, S., Wierstra, D. and Schmidhuber, J. [2010]. Exponential natural evolution strategies, *Proceedings of the International Conference on Genetic and Evolutionary Computation (GECCO)*, ACM, pp. 393–400.

Gloye, A., Wiesel, F., Tenchio, O. and Simon, M. [2005]. Reinforcing the Driving Quality of Soccer Playing Robots by Anticipation, *IT - Information Technology* **47**(5).

Gonzalez-Aguirre, D., Hoch, J., Rohl, S., Asfour, T., Bayro-Corrochano, E. and Dillmann, R. [2011]. Towards shape-based visual object categorization for humanoid robots, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 5226–5232.

Gonzalez, R. C. and Woods, R. E. [2006]. *Digital Image Processing*, 3rd edn, Prentice-Hall.

Gori, I., Fanello, S., Odone, F. and Metta, G. [2013]. A compositional approach for 3d arm-hand action recognition, *Proceedings of the IEEE Workshop on Robot Vision (WoRV)*.

Gupta, K. [1986]. Kinematic analysis of manipulators using the zero reference position description, *The International Journal of Robotics Research* **5**(2): 5.

Hager, G., Chang, W.-C. and Morse, A. [1995]. Robot hand-eye coordination based on stereo vision, *IEEE Control Systems* **15**(1): 30–39.

Halatci, I., Iagnemma, K. et al. [2008]. A study of visual and tactile terrain classification and classifier fusion for planetary exploration rovers, *Robotica* **26**(6): 767–779.

Handley, S. [1993]. Automatic learning of a detector for alpha-helices in protein sequences via genetic programming, *in* S. Forrest (ed.), *Proceedings of the International Conference on Genetic Algorithms (ICGA)*, Morgan Kaufmann, University of Illinois at Urbana-Champaign, pp. 271–278.

Harding, S. [2008]. Evolution of image filters on graphics processor units using cartesian genetic programming, *in* J. Wang (ed.), *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, Hong Kong, pp. 1921–1928.

Harding, S. and Banzhaf, W. [2008]. Genetic programming on GPUs for image processing, *International Journal of High Performance Systems Architecture* **1**(4): 231–240.

Harding, S., Banzhaf, W. and Miller, J. F. [2010a]. A survey of self modifying cartesian genetic programming, *in* R. Riolo, T. McConaghy and E. Vladislavleva (eds), *Genetic Programming Theory and Practice VIII*, Vol. 8, Springer, pp. 91–107.

Harding, S., Graziano, V., Leitner, J. and Schmidhuber, J. [2012]. Mt-cgp: Mixed type cartesian genetic programming, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Harding, S. L. and Banzhaf, W. [2009]. Distributed genetic programming on GPUs using CUDA, *in* I. Hidalgo, F. Fernandez and J. Lanchares (eds), *Workshop on Parallel Architectures and Bioinspired Algorithms*, Raleigh, NC, pp. 1–10.

Harding, S. L. and Banzhaf, W. [2011]. Hardware acceleration for cgp: Graphics processing units, *in* J. F. Miller (ed.), *Cartesian Genetic Programming*, Natural Computing Series, Springer Berlin Heidelberg, pp. 231–253.

Harding, S., Miller, J. F. and Banzhaf, W. [2010b]. Developments in cartesian genetic programming: self-modifying CGP, *Genetic Programming and Evolvable Machines* **11**(3/4): 397–439. Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines.

Hardyck, C. D., Petrinovich, L. F. and Ellsworth, D. W. [1966]. Feedback of speech muscle activity during silent reading: Rapid extinction, *Science* **154**(3755): 1467–1468.

Hart, P. E., Nilsson, N. J. and Raphael, B. [1968]. A formal basis for the heuristic determination of minimum cost paths, *Systems Science and Cybernetics, IEEE Transactions on* **4**(2): 100–107.

Hart, S., Ou, S., Sweeney, J. and Grupen, R. [2006]. A framework for learning declarative structure, *Proceedings of the RSS Workshop: Manipulation in Human Environments*.

Hartley, R. and Zisserman, A. [2000]. *Multiple view geometry in computer vision*, 2nd edn, Cambridge University Press.

Harvey, I., Husbands, P., Cliff, D., Thompson, A. and Jakobi, N. [1997]. Evolutionary robotics: the Sussex approach, *Robotics and Autonomous Systems* **20**(2): 205–224.

Hebert, P., Burdick, J., Howard, T., Hudson, N. and Ma, J. [2012]. Action inference: The next best touch, *Proceedings of the RSS Workshop: Mobile Manipulation*.

Himmelsbach, M., Müller, A., Lüttel, T. and Wünsche, H.-J. [2008]. LIDAR-based 3D object perception, *Proceedings of the International Workshop on Cognition for Technical Systems*.

Hoffmann, H., Schenck, W. and Möller, R. [2005]. Learning visuomotor transformations for gaze-control and grasping, *Biological Cybernetics* **93**: 119–130.

Holland, J. H. [1975]. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press.

Holland, O. and Knight, R. [2006]. The anthropomimetic principle, *AISB Symposium on Biologically Inspired Robotics*.

Homer and Fagles, R. [1990]. *The Iliad*, Penguin Classics, Viking.
URL: *http://books.google.ch/books?id=TpSM2PG4JfYC*

Horn, B. [1986]. *Robot Vision*, MIT Press.

Hornberg, A. [2007]. *Handbook of machine vision*, Wiley.

Hudson, N., Howard, T., Ma, J., Jain, A., Bajracharya, M., Myint, S., Kuo, C., Matthies, L., Backes, P., Hebert, P., Fuchs, T. and Burdick, J. [2012]. End-to-end dexterous manipulation with deliberate interactive estimation, *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Hulse, M., McBrid, S. and Lee, M. [2009]. Robotic hand-eye coordination without global reference: A biologically inspired learning scheme, *Proceedings of the International Conference on Developmental Robotics*.

Hülse, M., McBride, S., Law, J. and Lee, M. [2010]. Integration of active vision and reaching from a developmental robotics perspective, *IEEE Transactions on Autonomous Mental Development* **2**(4): 355–367.

Huntress, W., Moroz, V. and Shevalev, I. [2003]. Lunar and planetary robotic exploration missions in the 20th century, *Space science reviews* **107**(3): 541–649.

Hutchinson, S., Hager, G. D. and Corke, P. I. [1996]. A tutorial on visual servo control, *IEEE Transactions on Robotics and Automation* **12**(5): 651–670.

Isard, M. and Blake, A. [1998]. Condensation—conditional density propagation for visual tracking, *International Journal of Computer Vision* **29**(1): 5–28.

Itti, L., Koch, C. and Niebur, E. [1998]. A model of saliency-based visual attention for rapid scene analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(11): 1254–1259.

Jarrett, K., Kavukcuoglu, K., Ranzato, M. and LeCun, Y. [2009]. What is the best multi-stage architecture for object recognition?, *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 2146–2153.

Jeannerod, M. [1997]. *The cognitive neuroscience of action.*, Blackwell Publishing.

Johnson, M. H. and Munakata, Y. [2005]. Processes of change in brain and cognitive development, *Trends in cognitive sciences* **9**(3): 152–158.

Jones, J. P. and Palmer, L. A. [1987]. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex, *Journal of Neurophysiology* **58**: 1233—1258.

Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P. and Munich, M. [2005]. The vSLAM Algorithm for Robust Localization and Mapping, *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Kaufmann, G. [2010]. *A flexible and safe environment for robotic experiments : a sandbox and testbed for experiments intended for the humanoid robot iCub*, Master's thesis, Università della Svizzera italiana (USI).

Kelley, H. J. [1960]. Gradient theory of optimal flight paths, *ARS Journal* **30**(10): 947–954.

Kemp, C., Edsinger, A. and Torres-Jara, E. [2007]. Challenges for robot manipulation in human environments [grand challenges of robotics], *IEEE Robotics & Automation Magazine* **14**(1): 20–29.

Khatib, O. [1986]. Real-time obstacle avoidance for manipulators and mobile robots, *The International Journal of Robotics Research* **5**(1): 90.

Kim, H., Murphy-Chutorian, E. and Triesch, J. [2006]. Semi-autonomous learning of objects, *Proceedings of the Computer Vision and Pattern Recognition Workshop (CVPRW)*.

Kirstein, S., Wersing, H. and Körner, E. [2008]. A biologically motivated visual memory architecture for online learning of objects, *Neural Networks* **21**(1): 65–77.

Koga, Y. and Latombe, J. [1994]. On multi-arm manipulation planning, *Proceedings of International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 945–952.

Kölsch, M. and Turk, M. [2004]. Robust hand detection, *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, p. 614.

Konidaris, G. [2013]. Designing Intelligent Robots: Reintegrating AI II. **URL:** *http://people.csail.mit.edu/gdk/dir2/*

Kormushev, P., Calinon, S. and Caldwell, D. G. [2013]. Reinforcement learning in robotics: Applications and real-world challenges, *Robotics* **2**(3): 122–148.

Koza, J. R. [1992]. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

Kragic, D. and Vincze, M. [2009]. Vision for robotics, *Foundations and Trends in Robotics* **1**(1): 1–78.

Kuffner Jr, J. J. and LaValle, S. M. [2000]. RRT-connect: An efficient approach to single-query path planning, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 995–1001.

Kuipers, B., Beeson, P., Modayil, J. and Provost, J. [2006]. Bootstrap learning of foundational representations, *Connection Science* **18**(2): 145–158.

Kuperstein, M. and Rubenstein, J. [1989]. Implementation of an adaptive neural controller for sensory-motor coordination, *IEEE Control Systems Magazine* **9**(3): 25–30.

Kusuda, Y. [2008]. Toyota's violin-playing robot, *Industrial Robot: An International Journal* **35**(6): 504–506.

Langdon, W. B. and Nordin, P. [2001]. Evolving Hand-Eye Coordination for a Humanoid Robot with Machine Code Genetic Programming, *Proceedings of the European Conference on Genetic Programming (EuroGP)*.

LaValle, S. [2006]. *Planning algorithms*, Cambridge University Press.

Leitner, J. [2009]. *Multi-robot formations for area coverage in space applications*, Master's thesis, Luleå tekniska universitet, Sweden.

Leitner, J., Ampatzis, C. and Izzo, D. [2010]. Evolving ANNs for spacecraft rendezvous and docking, *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

Leitner, J., Bernardino, A. and Santos-Victor, J. [2008]. A benchmark on stereo disparity estimation for humanoid robots, *Proceedings of the International Conference on Autonomous Robot Systems and Competitions (Robotica)*.

Leitner, J., Chandrashekhariah, P., Harding, S., Frank, M., Spina, G., Forster, A., Triesch, J. and Schmidhuber, J. [2012a]. Autonomous learning of robust visual object detection and identification on a humanoid, *Proceedings of the International Conference on Development and Learning and Epigenetic Robotics (ICDL)*.

Leitner, J., Harding, S., Chandrashekhariah, P., Frank, M., A. Förster, A., Triesch, J. and Schmidhuber, J. [2013a]. Learning visual object detection and localization using icVision, *Biologically Inspired Cognitive Architectures* **5**: 29 – 41.

Leitner, J., Harding, S., Förster, A. and Schmidhuber, J. [2012b]. Mars terrain image classification using cartesian genetic programming, *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2012c]. icVision: A Modular Vision System for Cognitive Robotics Research, *Proceedings of the International Conference on Cognitive Systems (CogSys)*.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2012d]. Learning spatial object localization from vision on a humanoid robot, *International Journal of Advanced Robotic Systems (ARS)* **9**.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2012e]. Towards spatial perception: Learning to locate objects from vision, *in* J. Szufnarowska (ed.), *Proceedings of the Post-Graduate Conference on Robotics and Development of Cognition*, pp. 20–23.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2012f]. Transferring spatial perception between robots operating in a shared workspace, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2013b]. An Integrated, Modular Framework for Computer Vision and Cognitive Robotics Research (icVision), *in* A. Chella, R. Pirrone, R. Sorbello and K. Jóhannsdóttir (eds), *Biologically Inspired Cognitive Architectures 2012*, Vol. 196 of *Advances in Intelligent Systems and Computing*, Springer Berlin Heidelberg, pp. 205–210.

Leitner, J., Harding, S., Frank, M., Förster, A. and Schmidhuber, J. [2013c]. Humanoid learns to detect its own hands, *Proceedings of the IEEE Conference on Evolutionary Computation (CEC)*, pp. 1411–1418.

Levinson, S., Silver, A. and Wendt, L. [2010]. Vision based balancing tasks for the icub platform: A case study for learning external dynamics, *Workshop on Open Source Robotics: iCub & Friends. International Conference on Humanoid Robotics*.

Li, T. and Shie, Y. [2007]. An incremental learning approach to motion planning with roadmap management, *Journal of Information Science and Engineering* **23**(2): 525–538.

Lima, P., Nardi, D., Kraetzschmar, G., Berghofer, J., Matteucci, M. and Buchanan, G. [2014]. Rockin innovation through robot competitions [competitions], *Robotics & Automation Magazine, IEEE* **21**(2): 8–12.

Linnainmaa, S. [1970]. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, Master's thesis, Univ. Helsinki.

Lowe, D. [1999]. Object Recognition from Local Scale-Invariant Features, *Proceedings of the International Conference on Computer Vision (ICCV)*.

Lozano-Pérez, T. and Wesley, M. A. [1979]. An algorithm for planning collision-free paths among polyhedral obstacles, *Communications of the ACM* **22**(10): 560–570.

Lungarella, M., Metta, G., Pfeifer, R. and Sandini, G. [2003]. Developmental robotics: a survey, *Connection Science* **15**(4): 151–190.

Maitin-Shepard, J., Cusumano-Towner, M., Lei, J. and Abbeel, P. [2010]. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2308–2315.

Maltarollo, V. G., Honório, K. M. and da Silva, A. B. F. [2013]. Applications of artificial neural networks in chemical problems, *in* K. Suzuki (ed.), *Artificial Neural Networks – Architectures and Applications*, InTech.

Marey, E.-J. [1890]. "des appareils enregistreurs de la vitesse", *La Nature* **878**.

Marr, D. [1982]. *Vision: A Computational Approach*, Freeman & Co.

Martínek, T. and Sekanina, L. [2005]. An evolvable image filter: Experimental evaluation of a complete hardware implementation in fpga., *in* J. M. Moreno, J. Madrenas and J. Cosp (eds), *ICES*, Vol. 3637 of *Lecture Notes in Computer Science*, Springer, pp. 76–85.

Masci, J., Giusti, A., Ciresan, D., Fricout, G. and Schmidhuber, J. [2013]. A fast learning algorithm for image segmentation with max-pooling convolutional networks, *Proceedings of the International Conference on Image Processing (ICIP)*.

Matthews, B. W. [1975]. Comparison of the predicted and observed secondary structure of T4 phage lysozyme., *Biochimica et Biophysica Acta* **405**(2): 442–451.

McCarty, M., Clifton, R., Ashmead, D., Lee, P. and Goubet, N. [2001]. How infants use vision for grasping objects, *Child development* **72**(4): 973–987.

McGill, S. G., Zhang, Y., Vadakedathu, L., Sreekumar, A., Yi, S.-J. and Lee, D. D. [2012]. Comparison of obstacle avoidance behaviors for a humanoid robot in real and simulated environments, *Proceedings of the Workshop on Humanoid Soccer Robots at the International Conference on Humanoid Robots*.

Meeden, L. and Blank, D. [2006]. Introduction to developmental robotics, *Connection Science* **18**(2): 93–96.

Meltzoff, A. [1988]. Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli., *Developmental Psychology* **24**(4): 470.

Merletti, R. and Parker, P. A. [2004]. *Electromyography: physiology, engineering, and non-invasive applications*, Vol. 11, John Wiley & Sons.

Metta, G. and Fitzpatrick, P. [2003]. Better vision through manipulation, *Adaptive Behavior* **11**(2): 109–128.

Metta, G., Fitzpatrick, P. and Natale, L. [2006]. YARP: Yet Another Robot Platform, *International Journal of Advanced Robotics Systems, Special Issue on Software Development and Integration in Robotics* **3**(1).

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A. and Montesano, L. [2010]. The iCub humanoid robot: An open-systems platform for research in cognitive development, *Neural Networks* **23**(8-9): 1125–1134.

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

Mikolajczyk, K. and Schmid, C. [2003]. A performance evaluation of local descriptors, *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*.

Miller, J. [1999]. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1135–1142.

Miller, J. F. (ed.) [2011]. *Cartesian Genetic Programming*, Natural Computing Series, Springer.

Miller, J. F. and Smith, S. L. [2006]. Redundancy and computational efficiency in cartesian genetic programming, *IEEE Transactions on Evoluationary Computation*, Vol. 10, pp. 167–174.

Miller, J. F. and Thomson, P. [2000]. Cartesian genetic programming, *Proceedings of the European Conference on Genetic Programming (EuroGP)*, pp. 121–132.

Miller, J. F., Thomson, P. and Fogarty, T. [1998]. Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study, *in* D. Quagliarella, J. Périaux, P. C. and G. Winter (eds), *Genetic algorithms and evolution strategies in engineering and computer science*, John Wiley & Sons.

Mishkin, A., Morrison, J., Nguyen, T., Stone, H., Cooper, B. and Wilcox, B. [1998]. Experiences with operations and autonomy of the Mars pathfinder Microrover, *Proceedings of the Aerospace Conference*, pp. 337–351.

Moors, M., Rohling, T. and Schulz, D. [2005]. A probabilistic approach to coordinated multi-robot indoor surveillance, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 3447–3452.

Morton, O. [2014]. Immigrants from the future, *The Economist Special Report: Robots* .

Nagi, J., Ducatelle, F., Caro, G. A. D., Ciresan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J. and Gambardella, L. M. [2011]. Max-pooling convolutional neural networks for vision-based hand gesture recognition, *Proceedings of the International Conference on Signal and Image Processing Applications*, pp. 342–347.

Natale, L., Nori, F., Metta, G., Fumagalli, M., Ivaldi, S., Pattacini, U., Randazzo, M., Schmitz, A. and Sandini, G. [2013]. The iCub platform: A tool for studying intrinsically motivated learning, *in* G. Baldassarre and M. Mirolli (eds), *Intrinsically Motivated Learning in Natural and Artificial Systems*, Springer, pp. 433–458.

Natale, L., Nori, F., Sandini, G. and Metta, G. [2007]. Learning precise 3d reaching in a humanoid robot, *Proceedings of the International Conference on Development and Learning (ICDL)*, pp. 324–329.

Nelson, G., Saunders, A., Neville, N., Swilling, B., Bondaryk, J., Billings, D., Lee, C., Playter, R. and Raibert, M. [2012]. Petman: A humanoid robot for testing chemical protective clothing, *Journal of the Robotics Society of Japan* **30**(4): 372–377.

Neuronics AG [2008]. Katana user manual and technical description.

Nilsson, N. [1969]. A mobile automaton: An application of artificial intelligence techniques, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Nilsson, N. [1984]. Shakey the robot, *Technical report*, DTIC Document.

Nolfi, S. and Floreano, D. [2000]. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*, The MIT Press.

Nolfi, S., Floreano, D., Miglino, O. and Mondada, F. [1994]. How to evolve autonomous robots: Different approaches in evolutionary robotics, *Artificial Life IV*, MIT Press, pp. 190–197.

Nori, F., Natale, L., Sandini, G. and Metta, G. [2007]. Autonomous learning of 3d reaching in a humanoid robot, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 1142–1147.

Oikonomidis, I., Kyriazis, N. and Argyros, A. [2011]. Efficient model-based 3D tracking of hand articulations using Kinect, *Proceedings of the British Machine Vision Conference*.

Oltean, M. [2005]. Evolving evolutionary algorithms using linear genetic programming, *Evolutionary Computation* **13**(3): 387–410.

Ott, C., Eiberger, O., Englsberger, J., Roa, M. A. and Albu-Schäffer, A. [2012]. Hardware and control concept for an experimental bipedal robot with joint torque sensors, *Journal of the Robotics Society of Japan* **30**(4): 378–382.

Ott, C., Eiberger, O., Friedl, W., Bauml, B., Hillenbrand, U., Borst, C., Albu-Schaffer, A., Brunner, B., Hirschmuller, H., Kielhofer, S. et al. [2006]. A humanoid two-arm system for dexterous manipulation, *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pp. 276–283.

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

Pal, N. R. and Pal, S. K. [1993]. A review on image segmentation techniques, *Pattern recognition* **26**(9): 1277–1294.

Panin, G., Ladikos, A. and Knoll, A. [2006]. An efficient and robust real-time contour tracking system, *Proceedings of the International Conference on Computer Vision Systems*.

Pathak, S., Pulina, L., Metta, G. and Tacchella, A. [2013]. Ensuring safety of policies learned by reinforcement: Reaching objects in the presence of obstacles with the icub, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Pattacini, U. [2011]. *Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub*, PhD thesis, Italian Institute of Technology, Genova.

Peters, J. and Schaal, S. [2008]. Learning to control in operational space, *The International Journal of Robotics Research* **27**(2): 197.

Peters, J., Vijayakumar, S. and Schaal, S. [2003]. Reinforcement learning for humanoid robotics, *Proceedings of the International Conference on Humanoid Robots*.

Pfeifer, R., Bongard, J. and Grand, S. [2007]. *How the body shapes the way we think: a new view of intelligence*, The MIT Press.

Piatt, R., Burridge, R., Diftler, M., Graf, J., Goza, M., Huber, E. and Brock, O. [2006]. Humanoid mobile manipulation using controller refinement, *Proceedings of the RSS Workshop: Manipulation in Human Environments*, pp. 94–101.

Plumert, J. and Spencer, J. [2007]. *The emerging spatial mind*, Oxford University Press.

Poli, R. [1996]. Genetic programming for image analysis, *Technical Report CSRP-96-1*, University of Birmingham, UK.

Pollen, D. A. and Ronner, S. F. [1981]. Phase relationship between adjacent simple cells in the visual cortex, *Science* **212**: 1409–1411.

Porges, O., Hertkorn, K., Brucker, M. and Roa, M. A. [2012]. Robotic hand pose estimation using vision, Poster Session at the IEEE RAS Summer School on “Robot Vision and Applications”.

Posner, M. [1989]. *Foundations of cognitive science*, The MIT Press.

Reaz, M. B. I., Hussain, M. S. and Mohd-Yasin, F. [2006]. Techniques of emg signal analysis: detection, processing, classification and applications, *Biological Procedures Online* **8**.

Rechenberg, I. [1965]. Cybernetic solution path of an experimental problem.

Redi, F. [1686]. *Esperienze intorno a diverse cose naturali e particolarmente a quelle che ci sono partat dalle Indie*. taken from J. R. Cram, M. D. Durie. The Basics of Surface Electromyography.

Rizzolatti, G. and Craighero, L. [2004]. The mirror-neuron system, *Annual Review of Neuroscience* **27**: 169–192.

Rosenblatt, F. [1961]. Principles of neurodynamics. perceptrons and the theory of brain mechanisms, *Technical report*, Cornell Aeronautical Lab. DTIC Document No. VG-1196-G-8.

Rosheim, M. E. [2006]. *Leonardo’s lost robots*, Springer.

Rosten, E. and Drummond, T. [2006]. Machine learning for high-speed corner detection, *Computer Vision–ECCV 2006*, Springer, pp. 430–443.

Rosten, E., Porter, R. and Drummond, T. [2010]. Faster and better: A machine learning approach to corner detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(1): 105–119.

Rublee, E., Rabaud, V., Konolige, K. and Bradski, G. [2011]. Orb: An efficient alternative to sift or surf, *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 2564–2571.

Russell, S. J. and Norvig, P. [2010]. *Artificial Intelligence: A Modern Approach*, 3rd edn, Prentice Hall.

Sadikov, A., Možina, M., Guid, M., Krivec, J. and Bratko, I. [2007]. Automated chess tutor, *in* H. Herik, P. Ciancarini and H. Donkers (eds), *Computers and Games*, Vol. 4630 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 13–25.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N. and Fujimura, K. [2002]. The intelligent ASIMO: System overview and integration, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Sauser, E. and Billard, A. [2005]. View sensitive cells as a neural basis for the representation of others in a self-centered frame of reference, *International Symposium on Imitation in Animals and Artifacts*.

Saut, J.-P., Ivaldi, S., Sahbani, A. and Bidaud, P. [2014]. Grasping objects localized from uncertain point cloud data, *Robotics and Autonomous Systems* . in press - available online.

Saxena, A., Driemeyer, J. and Ng, A. [2008]. Robotic grasping of novel objects using vision, *The International Journal of Robotics Research* **27**(2): 157.

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

Schmidhuber, J. [1991]. Curious model-building control systems, *Neural Networks, 1991. 1991 International Joint Conference on*, pp. 1458–1463.

Schmidt, M. and Lipson, H. [2009]. Distilling Free-Form Natural Laws from Experimental Data, *Science* pp. 1–5.

Schoner, G. and Dose, M. [1992]. A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion, *Robotics and Autonomous Systems* **10**(4): 253–267.

Schutte, A., Spencer, J. and Schöner, G. [2003]. Testing the dynamic field theory: Working memory for locations becomes more spatially precise over development, *Child Development* **74**(5): 1393–1417.

Schwefel, H.-P. [1965]. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik, *Master's thesis, Hermann Föttinger Institute for Hydrodynamics, Technical University of Berlin* .

Sekanina, L., Harding, S. L., Banzhaf, W. and Kowaliw, T. [2011]. Image processing and CGP, *in* J. F. Miller (ed.), *Cartesian Genetic Programming*, Natural Computing Series, Springer, chapter 6, pp. 181–215.

Seo, K. and Kim, Y. [2010]. Scale and rotation-robust genetic programming-based corner detectors, *in* C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekart, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuss, J. Togelius and G. N. Yannakakis (eds), *EvoINTELLIGENCE*, Vol. 6024 of *Lecture Notes in Computer Science*, Springer, Istanbul, pp. 381–391.

Shanahan, M. [2006]. A cognitive architecture that combines internal simulation with a global workspace, *Consciousness and Cognition* **15**(2): 433–449.

Shang, C. and Barnes, D. [2012]. Classification of Mars McMurdo Panorama Images Using Machine Learning Techniques, *Acta Futura* **5**: 29–38.

Shang, C., Barnes, D. and Shen, Q. [2011]. Facilitating efficient mars terrain image classification with fuzzy-rough feature selection, *International Journal of Hybrid Intelligent Systems* **8**(1): 3–13.

Shirakawa, S. and Nagao, T. [2007]. Feed forward genetic image network: Toward efficient automatic construction of image processing algorithm, *in* G. Bebis, R. Boyle, B. Parvin, D. Koracin, N. Paragios, S.-M. Tanveer, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Muller and T. Malzbender (eds), *Advances in Visual Computing: Proceedings of the 3rd International Symposium on Visual Computing (ISVC 2007) Part II*, Vol. 4842 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 287–297.

Shirakawa, S., Nakayama, S. and Nagao, T. [2009]. Genetic image network for image classification, *in* M. Giacobini, A. Brabazon, S. Cagnoni, G. A. D. Caro, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado, J. McCormack, M. O'Neill, F. Neri, M. Preuss, F. Rothlauf, E. Tarantino and S. Yang (eds), *Applications of Evolutionary Computing, EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG*, Vol. 5484 of *Lecture Notes in Computer Science*, Springer, pp. 395–404.

Sicard, G., Salaun, C., Ivaldi, S., Padois, V. and Sigaud, O. [2011]. Learning the velocity kinematics of iCub for model-based control: XCSF versus LWPR, *Proceedings of the International Conference on Humanoid Robots*.

Silva, S., Vasconcelos, M. J. and Melo, J. B. [2010]. Bloat free genetic programming versus classification trees for identification of burned areas in satellite imagery, *in* C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekart, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuss, J. Togelius and G. N. Yannakakis (eds), *EvoIASP*, Vol. 6024 of *Lecture Notes in Computer Science*, Springer, Istanbul, pp. 272–281.

Slaný, K. and Sekanina, L. [2007]. Fitness landscape analysis and image filter evolution using functional-level CGP, *in* M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi and A. I. Esparcia-Alcázar (eds), *Proceedings of the European Conference on Genetic Programming (EuroGP)*, Vol. 4445 of *Lecture Notes in Computer Science*, Springer, Valencia, Spain, pp. 311–320.

Smith, S. L., Leggett, S. and Tyrrell, A. M. [2005]. An implicit context representation for evolving image processing filters, *in* F. Rothlauf, J. Branke, S. Cagnoni, D. W. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. D. Smith and G. Squillero (eds), *Applications of Evolutionary Computing, EvoWorkshops2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, Vol. 3449 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 407–416.

Solis, J. and Takanishi, A. [2011]. Wind instrument playing humanoid robots, *in* J. Solis and K. Ng (eds), *Musical Robots and Interactive Multimodal Systems*, Vol. 74 of *Springer Tracts in Advanced Robotics*, Springer Berlin Heidelberg, pp. 195–213.

Spina, T. V., Montoya-Zegarra, J. A., Falcao, A. X. and Miranda, P. A. V. [2009]. Fast interactive segmentation of natural images using the image foresting transform, *Proceedings of the International Conference on Digital Signal Processing*, pp. 1–8.

Stasse, O., Foissotte, T., Larlus, D., Kheddar, A., Yokoi, K. et al. [2008]. Treasure hunting for humanoids robot, *Proceedings of Workshop on Cognitive Humanoid Vision at the International Conference on Humanoids Robots*.

Stollenga, M., Pape, L., Frank, M., Leitner, J., Fõrster, A. and Schmidhuber, J. [2013]. Task-relevant roadmaps: A framework for humanoid motion planning, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

Stückler, J., Badami, I., Droeschel, D., Gräve, K., Holz, D., McElhone, M., Nieuwenhuisen, M., Schreiber, M., Schwarz, M. and Behnke, S. [2013]. Nimbro@home: Winning team of the robocup@home competition 2012, *Robot Soccer World Cup XVI*, Springer, pp. 94–105.

Suckling, J., Parker, J., Dance, D., Astley, S., Hutt, I., Boggis, C., Ricketts, I., Stamatakis, E., Cerneaz, N., Kok, S., Taylor, P., Betal, D. and Savage, J. [1994]. The mammographic images analysis society digital mammogram database, *Experta Medica International Congress Series* **1069**: 375–378.

Sutton, R. and Barto, A. [1998]. *Reinforcement learning: An introduction*, Cambridge Univ Press.

Szymanski, J. J., Brumby, S. P., Pope, P., Eads, D., Esch-Mosher, D., Galassi, M., Harvey, N. R., McCulloch, H. D. W., Perkins, S. J., Porter, R., Theiler, J., Young, A. C., Bloch, J. J. and David, N. [2002]. Feature extraction from multiple data sources using genetic programming, *in* S. S. Shen and P. E. Lewis (eds), *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII*, Vol. 4725 of *SPIE*, pp. 338–345.
URL: *http://citeseer.ist.psu.edu/540967.html*

Tackett, W. A. [1993]. Genetic programming for feature discovery and image discrimination, *Proceedings of the International Conference on Genetic Algorithms*, pp. 303–309.

Takagi, S. [2006]. Toyota partner robots, *Journal-Robotics Society of Japan* **24**(2): 62.

THE [2012]. EU Project Consortium: The Hand Embodied, `http://www.thehandembodied.eu/`.

Thrun, S. et al. [2002]. Robotic mapping: A survey, *Exploring artificial intelligence in the new millennium* pp. 1–35.

Traver, V. J. and Bernardino, A. [2010]. A review of log-polar imaging for visual perception in robotics, *Robotics and Autonomous Systems* **58**: 378–398.

Trujillo, L. and Olague, G. [2008]. Automated design of image operators that detect interest points, *Evolutionary Computation* **16**(4): 483–507.

Trujillo, L. and Olague, G. [2009]. Detecting scale-invariant regions using evolved image operators, *in* S. Cagnoni (ed.), *Evolutionary Image Analysis and*

Signal Processing, Vol. 213 of *Studies in Computational Intelligence*, Springer, Berlin / Heidelberg, pp. 21–40.

Tsagarakis, N., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., Righetti, L., Santos-Victor, J., Ijspeert, A., Carrozza, M. and Caldwell, D. [2007]. iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research, *Advanced Robotics* **21**: 1151–1175.

Tsagarakis, N., Vanderborght, B., Laffranchi, M. and Caldwell, D. [2009]. The mechanical design of the new lower body for the child humanoid robot icub, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 4962–4968.

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

Uto, K., Kosugi, Y. and Ogatay, T. [2009]. Evaluation of oak wilt index based on genetic programming, *First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, WHISPERS '09*, pp. 1–4.

Vafaie, H. and De Jong, K. [1992]. Genetic algorithms as a tool for feature selection in machine learning, *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 200–203.

Vahrenkamp, N., Asfour, T. and Dillmann, R. [2012]. Simultaneous grasp and motion planning: humanoid robot armar-iii, *IEEE Robotics & Automation Magazine* **19**(2): 43–57.

Vahrenkamp, N., Wieland, S., Azad, P., Gonzalez, D., Asfour, T. and Dillmann, R. [2008]. Visual servoing for humanoid grasping and manipulation tasks, *Proceedings of the International Conference on Humanoid Robots*, pp. 406–412.

van den Bergen, G. [2004]. *Collision detection in interactive 3D environments*, Morgan Kaufmann.

Vasicek, Z. and Sekanina, L. [2007]. Evaluation of a new platform for image filter evolution, *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 577–586.

Čapek, K. [1920]. *Rossum's universal robots (R.U.R.)*, Penguin, London. republished in 2004.

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

Vernon, D., Metta, G. and Sandini, G. [2007b]. The iCub Cognitive Architecture: Interactive Development in a Humanoid Robot, *Proceedings of the International Conference on Development and Learning (ICDL)*.

Wang, J. and Tan, Y. [2011]. Morphological image enhancement procedure design by using genetic programming, *in* N. Krasnogor, P. L. Lanzi, A. Engelbrecht, D. Pelta, C. Gershenson, G. Squillero, A. Freitas, M. Ritchie, M. Preuss, C. Gagne, Y. S. Ong, G. Raidl, M. Gallager, J. Lozano, C. Coello-Coello, D. L. Silva, N. Hansen, S. Meyer-Nieberg, J. Smith, G. Eiben, E. Bernado-Mansilla, W. Browne, L. Spector, T. Yu, J. Clune, G. Hornby, M.-L. Wong, P. Collet, S. Gustafson, J.-P. Watson, M. Sipper, S. Poulding, G. Ochoa, M. Schoenauer, C. Witt and A. Auger (eds), *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO)*, ACM, Dublin, Ireland, pp. 1435–1442.

Watchareeruetai, U., Takeuchi, Y., Matsumoto, T. and Ohnishi, N. [2008]. Transformation of redundant representations of linear genetic programming into canonical forms for efficient extraction of image features, *in* J. Wang (ed.), *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI)*, IEEE Press, pp. 1996–2003.

WAY [2012]. EU Project Consortium: Wearable Interfaces for Hand Function Recovery, `http://www.wayproject.eu/`.

Wegner, S., Harms, T., Builtjes, J. H., Oswald, H. and Fleck, E. [1995]. The watershed transformation for multiresolution image segmentation, *Proceedings of the International Conference on Image Analysis and Processing*.

Welke, K., Issac, J., Schiebener, D., Asfour, T. and Dillmann, R. [2010]. Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot, *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2012–2019.

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

Werbos, P. J. [1982]. Applications of advances in nonlinear sensitivity analysis, *System modeling and optimization*, Springer, pp. 762–770.

Wiener, N. [1948]. *Cybernetics: Or Control and Communication in the Animal and the Machine*, Hermann & Cie, Paris.

Wijesinghe, G. and Ciesielski, V. [2007]. Using restricted loops in genetic programming for image classification, *in* D. Srinivasan and L. Wang (eds), *2007 IEEE Congress on Evolutionary Computation*, IEEE Computational Intelligence Society, IEEE Press, Singapore, pp. 4569–4576.

Wikipedia [2014]. History of robots. [Online; accessed 07-May-2014].
URL: *http://en.wikipedia.org/w/index.php?title=History_of_robots&oldid=607524188*

Wiskott, L., Fellous, J., Krüger, N. and Van Der Malsburg, C. [1997]. Face recognition by elastic bunch graph matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 775–779.

Wolf, M. T., Assad, C., Stoica, A., You, K., Jethani, H., Vernacchia, M. T., Fromm, J. and Iwashita, Y. [2013]. Decoding static and dynamic arm and hand gestures from the jpl biosleeve, *Proceedings of the Aerospace Conference*.

Wolpert, D., Ghahramani, Z. and Flanagan, J. [2001]. Perspectives and problems in motor learning, *Trends in cognitive sciences* **5**(11): 487–494.

Wyatt, J. and Hawes, N. [2008]. Multiple workspaces as an architecture for cognition, *in* A. Samsonovich (ed.), *AAAI Fall Symposium on Biologically Inspired Cognitive Architectures*, The AAAI Press.

Yu, J. and Bhanu, B. [2006]. Evolutionary feature synthesis for facial expression recognition, *Pattern Recognition Letters* **27**(11): 1289–1298.

Zhang, M., Ciesielski, V. B. and Andreae, P. [2003]. A domain-independent window approach to multiclass object detection using genetic programming, *EURASIP Journal on Applied Signal Processing* **2003**(8): 841–859. Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis.

Zhang, Z. [2000]. A flexible new technique for camera calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(11): 1330–1334.
URL: *http://dx.doi.org/10.1109/34.888718*

Zhu, X., Yang, J. and Waibel, A. [2000]. Segmenting hands of arbitrary color, *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 446–453.