A Self-Feeding Robot

by

Eduardo R. Torres-Jara

Ing., Escuela Politécnica del Ejército (1995)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Masters of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2002

© Massachusetts Institute of Technology 2002. All rights reserved.

February 4, 2002

Certified by..... Rodney Brooks Fujitsu Professor of Computer Science and Engineering Thesis Supervisor

Accepted by Arthur C. Smith Chairman, Department Committee on Graduate Students

A Self-Feeding Robot

by

Eduardo R. Torres-Jara

Submitted to the Department of Electrical Engineering and Computer Science on February 4, 2002, in partial fulfillment of the requirements for the degree of Masters of Science in Electrical Engineering and Computer Science

Abstract

This thesis describes the design and construction of a self-feeding robot. Biological models have been the inspiration for much of the current Embedded Robotics research. However, some aspects of living creatures have not been completely explored in these models. One of these aspects is the need that living creatures have for food in order to maintain an adequate energy level to function. This thesis proposes to build a robot capable of acquiring its "food." In this case the "food" is electric energy acquired from the power outlets in the walls. The energy from the power outlets is used to recharge the robot's batteries. Therefore, the robot has skills such as: searching for power outlets, connecting to power outlets, and recharging its batteries.

Thesis Supervisor: Rodney Brooks Title: Fujitsu Professor of Computer Science and Engineering

Acknowledgments

Working in robotics at MIT has been a life-long goal, and there is nothing better than working with the leader in this field, Rod Brooks. Thank you Rod for inspiring me to work on robotics.

I also would like to thank the people with whom I have been working closely during these years. They are the Humanoid Robotics Group: Jessica, Aaron, Paulina, Paul, Juan, Charlie, Naoki, Artur, Lijin, Scaz, BP, Maddog, Cindy, Una-May, Martin and Giorgio. They have been a source of inspiration, support, and knowledge – especially Paul, Giorgio, and Artur who were of great help throughout this project.

Going from the "Athens" of Ecuador to the "Athens" of North America has been an interesting journey for me, which I would never have made without the support of my family: my Dad Teodoro, my Mom Cecilia and my sisters Patricia and Bibiana. I love you.

My office mates Jessica and Paulina deserve special mention. Jessica has been of great help, not only proof-reading this thesis but also being there in difficult times. And Paulina always has the right words to encourage me to continue.

Life brings us a lot of surprises; good and bad things happen, but for making things better, God gave us friends. People who are with you for the good and the bad moments, people who really love you. Thanks to God and Maria Auxiliadora for my family and all my good friends. And thanks to my good friends for being there: Santiago, Samantha, Cristina, Alvaro, Vlady, Chichi, Pablo, Marco, Marcelo, Mercy, Jose, Ursula, Juan, Martha, Maria Fernanda, Michael, Marisol, Rafa, Victor, Paul, Giorgio, Karen and all the ones I am forgetting to mention. Without them life would be hard.

Contents

1	Intr	oduction	9
	1.1	Scope of Investigation	10
	1.2	Review of Thesis Contents	10
2	Lite	erature Survey	12
	2.1	Overview	12
	2.2	Self-Feeding	12
	2.3	Vision	15
		2.3.1 Pattern Recognition	16
		2.3.2 Obstacle Detection	18
	2.4	Force Control	20
3	Har	dware: Design and Implementation	21
3	Har 3.1	dware: Design and Implementation Overview	21 21
3	Har 3.1 3.2	dware: Design and Implementation Overview	21 21 21
3	Har 3.1 3.2	dware: Design and Implementation Overview Mechanical Design 3.2.1	 21 21 21 22
3	Har 3.1 3.2	dware: Design and Implementation Overview	 21 21 21 22 25
3	Har 3.1 3.2	dware: Design and Implementation Overview	 21 21 21 22 25 27
3	Har 3.1 3.2	dware: Design and Implementation Overview	 21 21 22 25 27 29
3	Har 3.1 3.2	dware: Design and Implementation Overview	 21 21 21 22 25 27 29 30
3	Har 3.1 3.2	dware: Design and Implementation Overview Mechanical Design 3.2.1 Arm 3.2.2 Plug 3.2.3 Mobile Platform 3.2.4 Camera for Navigation 3.2.5 Camera for Outlet Detection Computer Hardware	 21 21 21 22 25 27 29 30 30
3	Har 3.1 3.2 3.3	dware: Design and Implementation Overview	 21 21 22 25 27 29 30 30 32

		3.3.3 Motor Controllers and Input/Output Board	35
	3.4	Power System	35
4	Visi	on Algorithms 3	3 9
	4.1	Introduction	39
	4.2	Obstacle detection	39
	4.3	Outlet detection	41
5	Sup	port Software 4	18
	5.1	Overview	48
	5.2	Introduction	48
	5.3	Operating System	49
	5.4	Base B12 daemon	49
	5.5	Arm daemon	50
	5.6	Obstacle detection daemon	50
	5.7	Outlet detection daemon	51
6	Soft	ware: Design and Implementation 5	52
6	Soft 6.1	ware: Design and Implementation5Overview5	52 52
6	Soft 6.1 6.2	ware: Design and Implementation 5 Overview	52 52 52
6	Soft 6.1 6.2 6.3	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5	52 52 52 54
6	Soft 6.1 6.2 6.3	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5	52 52 52 54 54
6	Soft 6.1 6.2 6.3	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5	52 52 52 54 54 54
6	Soft 6.1 6.2 6.3 6.4	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 Behavior: Searching for an Outlet 5	52 52 52 54 54 56 56
6	Soft 6.1 6.2 6.3	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 Behavior: Searching for an Outlet 5 6.4.1 The Outlet Detection Agent 5	52 52 52 54 54 56 56 56
6	Soft 6.1 6.2 6.3	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 6.3.2 The Obstacle Avoidance Agent Behavior: Searching for an Outlet 5 6.4.1 The Outlet Detection Agent 5 6.4.2 The Scanning Agent 5	52 52 52 54 54 56 56 57 57
6	Soft 6.1 6.2 6.3 6.4	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 6.4.1 The Outlet Detection Agent 5 6.4.2 The Scanning Agent 5 Behavior: Connecting to an Outlet 5	52 52 52 54 54 56 56 57 57 58
6	Soft 6.1 6.2 6.3 6.4	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 Behavior: Searching for an Outlet 5 6.4.1 The Outlet Detection Agent 5 6.4.2 The Scanning Agent 5 Behavior: Connecting to an Outlet 5 6.5.1 The Arm Motor Control Agent 5	52 52 54 54 56 56 57 57 58 58
6	Soft 6.1 6.2 6.3 6.4	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 Behavior: Searching for an Outlet 5 6.4.1 The Outlet Detection Agent 5 6.4.2 The Scanning Agent 5 8 6.5.1 The Arm Motor Control Agent 5 6.5.2 The Orientation Agent 5	52 52 54 54 56 56 57 57 58 58 58
6	Soft 6.1 6.2 6.3 6.4	ware: Design and Implementation 5 Overview 5 Introduction 5 Behavior: Wandering Around 5 6.3.1 The Base Motor Control Agent 5 6.3.2 The Obstacle Avoidance Agent 5 Behavior: Searching for an Outlet 5 6.4.1 The Outlet Detection Agent 5 6.4.2 The Scanning Agent 5 8 6.5.1 The Arm Motor Control Agent 5 6.5.2 The Orientation Agent 5 6.5.3 The Approaching Agent 5	52 52 54 54 56 56 57 58 58 58 59 59

		6.5.5	The Plugging Agent	60
		6.5.6	The Unplugging Agent	61
7	Cor	clusio	n and Future Work	62
	7.1	Analy	sis and Results	62
	7.2	Concl	usions and Recommendations For Future Work	69

List of Figures

2-1	Tortoise of W.G. Walter	13
2-2	Hopkins beast	14
3-1	Robot	22
3-2	Arm	23
3-3	In/Out DOF. Top view of the arm	24
3-4	Up/Down DOF. Lateral view of the arm	24
3-5	Plug	27
3-6	Mechanism for passive compliance.	28
3-7	Detail of an FSR	28
3-8	Construction of an FSR	29
3-9	Characteristic resistance of an FSR	29
3-10	Hardware architecture.	31
3-11	Power System	38
4-1	Obstacle detection.	40
4-2	Classes of features.	42
4-3	Positive examples	44
4-4	Negative examples	45
4-5	Boosting algorithm.	46
4-6	Outlet features.	47
4-7	Example of the detector working	47
6-1	Software architecture	53

6-2	Wandering around behavior block diagram	55
6-3	Search for an outlet behavior block diagram	57
6-4	Connecting to an outlet behavior block diagram	58
7-1	Features chosen by the learning algorithm.	63
7-2	Additional features	64
7-3	Example 1	66
7-4	Example 2	68
7-5	Trajectory of the robot.	69

Chapter 1

Introduction

This thesis describes the design, implementation, and evaluation of a self-feeding robot. The motivation behind this work is to build a robotic system that allows us to research living organisms. One of the most important aspects of living organisms is the fact that they are completely autonomous; they are able to survive without external help in their environments. Consequently, one of the main drives of living systems is to obtain the energy necessary for their operation from their environment. They do this by acquiring and processing food in a very efficient way. Beyond this basic skill, living organisms have developed many "intelligent" behaviors. Many researchers in the field of Artificial Intelligence such as Brooks (1989), Maes (1989), Arbib (1992), Arkin (1998) and Breazeal (2000) have studied these behaviors and their interactions. However, the study of the "feeding" aspect of living organisms has not received much attention.

Therefore, we built a robot that emulates the self-feeding behavior of a living organism. The robot works with batteries. Consequently, the kind of "food" (energy) that it needs is electrical energy. The environment in which the robot operates is the 9th floor of the MIT AI Laboratory.

Acquiring food in an efficient way is inherent to living organisms. Nevertheless, this is not an easy task for a robot. The skills that our robot needs to have are: searching for power outlets, connecting to power outlets, and recharging its batteries.

When the robot needs to recharge its batteries, a visual search detects power

outlets. Once an outlet is detected, a visual navigation system directs the robot towards the power outlet. The robot then accesses the power outlet using a plug on its arm and visual feedback. When there is electrical contact between the power outlet and the robot, the robot's batteries start to recharge.

Building a robot with such skills presents many challenges such as: on-board computation and power, visual searching, motor-vision coordination, and navigation.

Vision is probably one of the most challenging parts. Animals in general rely a lot on their vision capabilities and can respond quickly to visual stimuli. However, for a robot, achieving the same performance is not an easy task because of both the differences between the computational model used by the robot and the living brain as well as the complexity of the problem. Another challenge is the "hand-eye" coordination necessary to connect the robot's plug into the outlet. This task demands force control of the actuator involved.

1.1 Scope of Investigation

Creating a robot that can represent a living organism is a complicated challenge about which we do not have much knowledge. We implement a robot that uses visual input to recognize outlets and to avoid obstacles. A supervised learning algorithm is used to learn the pattern of the outlet. It is based on an algorithm used for face detection since the two problem are fairly similar. A plug mounted on an arm is used to connect the robot to an outlet. The force applied to the plug is controlled in order to facilitate this task. In order to have enough computation for running all the algorithms in real-time, the robot has an on-board computer network. The behaviors of the robot are implemented using agents.

1.2 Review of Thesis Contents

The thesis is organized as follows:

• Chapter 2 provides a survey of background material supporting this work.

- Chapter 3 describes the design and implementation of the hardware of the robot.
- **Chapter 4** describes the vision algorithms implemented for navigation and pattern recognition.
- **Chapter 5** describes the support software used to develop the behaviors of the robot.
- Chapter 6 describes the software architecture and details modules implemented.
- Chapter 7 presents the results and conclusions.

Chapter 2

Literature Survey

2.1 Overview

In this chapter we review the related literature that supports this thesis. The survey starts with a description of previous robots aimed at the study of different aspects of living creatures. Then, we discuss work done in other areas related to the design of the robot, such as pattern recognition, navigation, and force control.

2.2 Self-Feeding

Many robots have been constructed to study different biological aspects. However, affording a robot the ability of "feed" itself (see Section 4) has not received much attention even though all living creatures are able to feed themselves. The idea of implementing such a capability in a robot can be tracked down to W.G. Walter's robots (Walter 1950, Walter 1951, Walter 1953). Walter named his robots Elmer (ELectro MEchanical Robot) and Elsie (Light Sensitive, with Internal and External stability). These robots were capable of recharging their batteries when they went back to a hutch.

As shown in Figure 2-1, each of the robots consisted of: a shell with a photocell on its top, a bumping sensor, two motors, two radio tubes, and a battery. One of the motors was used for translating, the other for steering. The light sensor was



Figure 2-1: Tortoise of W.G. Walter

connected to the steering motor. This configuration allowed the robot to explore its environment, according to the following description.

When it did not detect any light, the robot turned on its two motors, describing a cycloidal trajectory. The steering motor also rotated the photocell in search of a light stimulus. This combination of actions generated an exploratory behavior. If the photocell detected light, the steering motor was turned off, causing the robot to move toward the light stimulus. However, if the light stimulus became too strong, the robot turned away from the light. This ensured the robot would not stay under the light the entire time.

This last behavior depended on the voltage level of the battery. If the battery level was below a threshold, the photocell amplifier increased its gain to detect the light from farther away. Then, when the robot detected light, it ran towards it. However, since its battery was low, the amplified light signal remained below the threshold that would cause the robot to run away. Therefore, in this low power condition, the robot would continue approaching the light source. In order to use this behavior for feeding the robot, a light bulb and a recharging system were placed inside a hutch. When the robot got into the hutch and the recharging system was activated, the motors were disconnected. Once the battery was recharged, the motors were reconnected and, because both the battery voltage and light intensity were above their thresholds, the robot ran away from the hutch.

The design was simple and effective. However, there is no evidence that the recharging of the batteries was automatic or even that it really existed. In order for the robot to recharge its batteries automatically, the recharging system would have had to have some kind of alignment process that was not described in Walter's literature. It is interesting to note that Walter referred to his robots as an "Imitation of Life" (Walter 1950). He considered his robots rough prototypes of living organisms and even coined them as the species Machine Speculatrix.

In 1964 the Hopkins Beast robots (Figure 2-2) were built at the Johns Hopkins University Applied Physics Laboratory. These robots were able to navigate the corridors using sonar. When they ran out of power, they looked for outlets to recharge themselves. One of the Hopkins Beasts found the power outlets by feeling along the walls. Another used photocells to optically find the outlets from a distance. The outlets had to contrast with the wall in order for this to work.



Figure 2-2: Hopkins beast

More recently, Brooks proposed the idea of Embodied Intelligence (Brooks 1986)

which radically changed the approach to robotics. Brooks' idea was implemented in many creature-like robots such as Genghis (Brooks 1989) and Attila (Angle 1991). These robots had the goal of "surviving" in their environment. As a consequence, robotics research turned its attention to biology, looking for models and inspiration to create versatile, robust, and intelligent robots. The idea of robots as creatures was especially successful because robots were able to deal with real-time tasks.

Brooks' research extended from single creature-like robots to groups of robots (Maes & Brooks 1990) searching for collaborative behaviors, such as those observed in many colonizing creatures like ants. This work influenced not only the robotics community, but space exploration as well (Brooks & Flynn 1989), with the creation of Sojourner to explore Mars (Golombek, Cook, Economou, Folkner, Haldemann, Kallemeyn, Knudsen, Manning, Moore, Parker, Rieder, Schofield, Smith & Vaughan 1997). Sojourner is a well-known representative of a robot that survives in its environment. However, it does not need to look for its energy sources because its solar panels automatically relay power as soon as the sunlight hits them.

Other autonomous robots that power themselves use different strategies. For example, SAGE (Nourbakhsh, Bobenage, Grange, Lutz, Meyer & Soto 1999) uses electrical energy while Gastrobot (Wilkinson 2000) uses chemical energy. SAGE gives tours in the Carnegie Museum of Natural History. The robot recharges itself when it returns to its base. However, it uses very visible artificial markers to identify its base and to recharge its battery. Unlike SAGE, Gastrobot has an artificial stomach that processes sugar to recharge its batteries. However, this robot is not capable of obtaining its own food.

2.3 Vision

Vision is the most important sense in many animals. Thus, many behaviors are associated with visual stimuli. Additionally, vision allows animals to react quickly to a changing environment around them . Therefore, developing a system that uses visual information is one of our goals. However, processing visual information is not an easy task. It is difficult to represent the information in an image and, even more so, to do this task with the same speed and accuracy with which animals and humans do. Besides the problem of representation, we have to face the problem of computation. Biological vision is processed by a highly parallel computer architecture. The robot, on the other hand, uses a serial computational model. This means that all the information has to go through a single machine. In order to try to simulate the same operation that a parallel system does, we need a very fast computer able to handle the adequate throughput.

2.3.1 Pattern Recognition

One of the tasks involved in this project is pattern recognition. Specifically, we must detect pattern of the power outlet. It is necessary to recognize this pattern at different scales and in slightly different orientations, while the robot is both stationary and moving. This task, simple for humans, is not an easy one for the robot's vision system.

Many implementations of pattern recognition deal with frontal face detection. In the following we survey methods that primarily use shape instead of color information. The assumptions made for frontal face detection are useful in the outlet detection. For example, we assume that an outlet will be vertically oriented mounted on a flat wall.

One way to solve the pattern recognition problem is with the method proposed by Rowley, Baluja & Kanade (1996). This method uses several neural networks that cover different areas of a 20×20 pixel image. Some of the areas covered by the neural networks overlap each other. After an intensive training over a sample set with thousands of various faces, the neural networks contain the information that allows the system to detect a face. In order to detect faces in an image at different scales, the image is downsampled. Each of the 11 steps in the downsampling process reduces the image by 10%. Though this method is effective, the response time is high. Because it is necessary to evaluate the neural networks at each point of the image at all different scales, the response time is not good enough to work in a realtime system. To improve the performance, Rowley et al. (1996), trained two different sets of networks. One is trained to preclassify windows that may contain faces; and the other, discriminates the faces from the preclassified windows. The second set of neural networks does a more complicated job since the images that are evaluated have features in common with the faces. A much simpler version of this method was used in Torres-Jara & Edsinger (1999). In this work just one neural network was used. However, the accuracy decreased as was expected.

A support vector machine has also been used to discriminate between face and non-face images (Osuma, Freund & Girosi 1997). The advantage of using a support vector machine is that the answer (the support vectors) defines the optimal boundary between the positive and negative samples. However, the training involves solving a quadratic optimization problem. The application of this method to an image is similar to the one described previously. In short, a window is used to scan the whole image. This window is evaluated using the support vector machine to determine whether there is a face present or not. The same procedure is repeated for various scales.

Sung & Poggio (1994) have proposed a mixture of Gaussian model to detect faces. This method, as with the previous, search for faces in various scales using downsampling.

Another way to approach the problem of pattern recognition is to create a template made up of edges of a pattern (e.g., a face or an outlet). Then an image can be correlated with the pattern at different scales. Correlation can be accelerated using the Fast Fourier Transform. However, depending on the size of the template, the number of operations needed to compute the correlation makes this method unattractive for real-time detection.

The method of pattern recognition used in this project has been presented in (Viola & Jones 2001). In this method, it is assumed that an image is made up of a combination of rectangular features. These rectangular features form a base of the image; however, this base is over determined. In order to recognize a pattern it is necessary to find the rectangular features that best describe the pattern. This is

done by training various discriminant functions and then boosting the answer using a variation of the Adaboost(Freund & Schapire 1995) algorithm. The big advantage of this method is that once the rectangular features have been determined, evaluation of features is fast. To do this, an integral of the image is calculated and then the area under this integral is easily found by differentiating the limits of the rectangular features (as to be explained in Section 4).

In Viola & Jones (2001), the method presented in Rowley et al. (1996) and the integral image method are compared. The results show that the latter is approximately fifteen times faster. One of the reasons for this performance is that to evaluate multiples scales it is not necessary to downsample in order to evaluate the image at multiple scales.

2.3.2 Obstacle Detection

Obstacle detection is obviously a necessary feature for a mobile robot. Many methods have been developed to detect obstacles, depending on the kind of sensors used. For instance infrared, ultrasonic sensors and cameras are common sensors. In this work the robot uses a camera to detect obstacles.

Outdoor and indoor environments have both been considered in the developing of obstacle detection methods using visual information. For outdoor environments one of the most well-known platforms is the CMU NAVLAB (Thorpe, Kanade & Shafer 1987). This platform is a modified Chevy Van equipped with cameras in the front and the rear, laser rangefinder, and an inertial navigational system. The goal of this platform is to follow a road. This was achieved by using a neural network system to process the images and determine the direction to drive. This neural network system, called ALVINN (Autonomous Land Vehicle In a Neural Network), was developed by Pomerleau (1993). Another system that uses vision and lasers to detect hazard conditions is the one used on the rover of the Mars Pathfinder Project (Matthies, Gat, Harrison, Wilcox, Volpe & Litwin 1995a).

Navigation systems based only on stereo vision have also been developed and tested in rovers (Matthies, Kelly & Litwin 1995b). A different approach for outdoor

navigation is presented in Lorigo, Brooks & Grimson (1997). A 64×64 image is used and the following constraints are assumed:

- The robot is on flat terrain. Therefore, the obstacles that are farther away appear at a higher position in the image than the obstacles that are closer to the robot.
- Obstacle boundaries are visible.
- The floor near the robot has no obstacles.

The image is scanned from bottom to top with a 20×10 pixel window. For each consecutive step, the window is placed one pixel above its previous location. For each window, histograms of the brightness gradient magnitude, the normalized RGB color, and the normalized HSV color are calculated. The histograms of each successive window placement in a column are compared with those of the bottom window. When the difference is greater than a given threshold an, obstacle is detected. This analysis is repeated for each column in the image.

Many approaches have been developed for indoor navigation as well. Two approaches that use vision as their only input are described in Santos-Victor, Sandini, Curotto & Garibaldi (1995) and Horswill (1993). Santos-Victor et al. (1995) uses optical flow to navigate the robot along corridors; the robot computes the velocity of variation of the left and right halves of the image, and steers toward the half with lower velocity. This method is based on the centering reflex of a bee. Horswill (1993) used a different method based on the constraints of corridors. It is assumed that the robot is on a flat floor and that the floor has little to no texture. Because the robot is on a flat floor, the distance from the bottom of the image to the closest point of an obstacle can be used to measure the distance from the robot to the obstacle. Also, because the floor has low texture, an edge detector can be used to separate the floor from obstacles. Consequently, the robot computes the edges of a grey-scale image and selects the edges whose distances are closest to the bottom of the image. These edges represent obstacles. A combination of this method and the one presented in Lorigo et al. (1997) is used in this thesis.

2.4 Force Control

In order to connect the plug to the outlet, it is necessary to control the force exerted by the arm. This can be done either by using passive systems or by actively controlling the forces.

One of the most well-known passive systems is Remote Center of Compliance(RCC) (Whitney & Nevins 1978), which was developed for assembling parts. In Whitney (1982), there is a description of the the geometric problem of assembling rigid parts in real environments in which conditions such as perfect alignment are difficult to achieve.

For an active system, the controlled parameter is either only the force or a ratio between the force and another variable. If the variable is position, it is called stiffness control; if the variable is velocity it is called damping control. A survey of these methods is presented in Whitney (1987) and in Zeng & Hemami (1997). They include: combined position and force control (Raibert & Craig 1981), stiffness control (Salisbury 1990), damping control (Whitney 1977), and impedance control (Hogan 1985).

RCC devices are less complex than active force controlled devices which is why they are preferred in simple assembly tasks. In this thesis an RCC device is used to connect the plug to the outlet. Its design is described in Section 3.

Chapter 3

Hardware: Design and Implementation

3.1 Overview

In this chapter we describe the design and implementation of the hardware of the robot from the global architecture to the details of its main components such as actuators, sensors, controllers and processors.

3.2 Mechanical Design

The mechanical design of the robot comprises the following parts: an electrical outlet plug, an arm, two cameras, a mobile platform, three on-board processors, an ethernet switch, an ethernet-wireless converter, motor controllers, and a power system. A picture of the robot is shown in Figure 3-1.

The main function of robot, its being able to acquire energy to recharge itself, revolves around a plug which is mounted on an arm. The arm has two degrees of freedom (DOF): up/down and in/out. There are sensors that allow the robot to detect when the plug has made electrical and/or mechanical contact. The arm and the plug are further mounted on a mobile platform that can translate and rotate. One of the cameras is mounted on the lower front part of the platform and is used for



Figure 3-1: Robot

detecting obstacles. The other camera is mounted on the up/down DOF of the arm (see Figure 3-1). This camera is used to detect power outlets. Each of these parts are described in detail in the following subsections.

There are also three computers, an ethernet switch, an ethernet-wireless converter, motor controllers, and a power system all located on the three layers of the robot's base. Their details and connections are explained in section 3.3.

3.2.1 Arm

The arm is used to reach the power outlets. It has two linear DOF: one for the up/down movement and the other for the in/out movement. The shape of the arm is shown in Figure 3-2.

The in/out DOF uses two 12-inch hardened anodized aluminium shafts mounted



Figure 3-2: Arm

on four frelon-lined linear bearings. Each shaft uses two bearings, so that when the arm is extended, the bars will have enough support so as not to bend. This configuration is shown in Figure 3-3. The combination of these specific bearings and shafts keeps the friction low and compensates for slight misalignments. The in/out DOF uses a cable system driven by a DC brushed motor with a 400 counts per rotation (cpr) optical encoder and a 134:1 gear head. The characteristics of the motor are in Table 3.1.

The up/down DOF is built with two 12-inch hardened anodized aluminium shafts mounted on two frelon-lined linear bearings. This DOF lifts the in/out apparatus as shown in Figure 3-4. The up/down DOF uses a ball screw that is driven through a belt by a brushed DC motor. The ball screw is mounted with shaft reducers and bearings and keeps the arm at a constant height with no power consumption. (In



Figure 3-3: In/Out DOF. Top view of the arm.



Figure 3-4: Up/Down DOF. Lateral view of the arm.

Motor MicroMo 2224			
Power	3.8W		
Voltage	12V		
Torque constant	2.05oz-in/A		
Stall Torque	2.80oz-in		
No load Speed	7800rpm		
Weight	1.62oz		
Efficiency	0.82		
Inertia	$0.382 \times 10^{-4} \text{oz-in-sec}^2$		
GearBox MicroMo 23/1			
Reduction	134:1		
Continuous Maximum Torque	99.13oz-in		
Intermittent Maximum Torque	141.61oz-in		
Efficiency	0.60		
Gearbox Weight	3.53oz		

Table 3.1: Motor Characteristics for the in/out DOF

other words, the motor is off when the arm is not moving.) This helps to conserve the battery charge. The DC motor driving this DOF has a 16 cpr magnetic encoder and a 141:1 gear head which enables it to lift the 2 Kg mass of the other DOF's components. The characteristics of the motor are presented in Table 3.2.

3.2.2 Plug

The plug consists of two copper prongs mounted in a plastic housing. The plastic housing provides electrical insulation. The plug is mounted on the arm using a mechanism that provides passive compliance. The passive compliance compensates for slight misalignments that may arise between the plug and the sockets of a power outlet.

The mechanism that provides the passive compliance is shown in Figure 3-6. The mechanism consists of a spring, a plug support, the plug itself, and a cable. The plug and the case remain together due to the normal force exerted by the spring through a cable. If lateral forces are applied to the plug, it moves or slides while remaining in contact with the plug support. Therefore, when the robot is trying to plug into an

Motor MicroMo 1524			
Power	1.4W		
Voltage	12V		
Torque constant	1.614oz-in/A		
Stall Torque	0.957oz-in		
No load Speed	9900rpm		
Weight	0.74oz		
Efficiency	0.76		
Inertia	$9.2 \times 10^{-6} \text{oz-in-s}^2$		
GearBox MicroMo 15/5			
Reduction	141:1		
Continuous Maximum Torque	14.2oz-in		
Intermittent Maximum Torque	21.2oz-in		
Efficiency	0.66		
Gearbox Weight	0.741oz		

Table 3.2: Motor Characteristics for the up/down DOF

outlet, even if the plug is misaligned, the plug's prongs will touch the funnels of the outlet. The mechanical contact will thus align the plug and the outlet enabling the robot to complete the insertion.

When the robot is extending its arm, the arm may touch a wall or other surface, making it necessary to detect this state. This is done using a Force Sensing Resistor (FSR). The FSR is a device made of a thick film polymer that decreases in resistance when the force applied to it increases. The way an FSR is constructed is shown in Figures 3-7 and 3-8.

An FSR has advantages over piezofilm and conductive rubber. An FSR is less sensitive to vibration and heat than piezofilm and has less hysteresis than conductive rubber. Though FSR's are not completely linear, they are suitable for the current application in which only a rough estimate of the force on the plug is needed.

An FSR is placed between the plug and the plug support. The force exerted by the plug is distributed over the sensor surface using a metallic plate and a double-sided laminated adhesive. The laminated adhesive has three functions: to act as a spring, to make the mounting easier, and to electrically insulate the FSR from the metallic



Figure 3-5: Plug.

parts of the plug support. The dynamic range of the sensor's resistance is from $1k\Omega$ to $1M\Omega$ (Figure 3-9). A voltage divider is used to convert the force to a voltage. The output of the voltage divider is connected to an A/D converter.

The plug on the end of the robot's arm is further connected to a switched power supply. This power supply converts the 110V AC from the wall's outlet to a regulated 24 V DC. The output of the power supply is used to power the battery charging system. In order to detect that the plug is under voltage, an optocoupler is connected to the output of the regulated power supply. By reading the output of this optocoupler, the robot can determine if a successful operation has been accomplished.

3.2.3 Mobile Platform

The robot uses a commercial mobile platform (RWI B12) which has a synchronous drive system that allows it to translate and rotate. The drive system includes three wheels that are kept parallel all the time. These three wheels are all driven, providing good traction.

The B12 platform includes an on-board NEC 78310 microcontroller. This microcontroller controls the motors and the battery system. The motor control for steering and driving uses optical feedback from the motors' encoders and pulse width modulation (PWM) to power the motors. The microcontroller also reads the voltage



Figure 3-6: Mechanism for passive compliance. The figure shows a cut of the plug in Figure 3-5. The cable that connects the spring with the plug is not shown in the figure.



Figure 3-7: Detail of an FSR. The figure shows the parts of an FSR.

and current from the batteries. Communication is through an RS-232 serial interface using ASCII commands. The command format is given by a two letter mnemonic command and a hexadecimal number. The commands allow the robot to read the status of the base and to change the motors' positions, velocities, and accelerations. Information about the commands can be found in (Real World Interface Inc. 1994).

The platform is powered by four lead-acid batteries. The electronics necessary to charge the batteries according to specifications is included on the platform. The charging system requires a connection to a regulated power supply not initially included on the platform.



Figure 3-8: Construction of an FSR. The figure shows the assembling process of an FSR



Figure 3-9: Characteristic resistance of an FSR. The figure illustrates the relation between the force applied and the resistance of an FSR. The resistance decreases as the force increases.

3.2.4 Camera for Navigation

The camera used for navigation is an analog color camera with NTSC output. The characteristics of the camera are shown in Table 3.3.

The camera uses a lens whose maximum aperture is F2.8 and whose horizontal field angle is 115°. This wide angle makes it possible to visualize a space as big as the width of the robot which is convenient for detecting obstacles.

This camera is mounted on the front of the platform and tilted to have a better image of the floor.

Camera Specifications		
Model	GP-CX161-15	
TV Standard	NTSC	
Horizontal scanning frequency	15.734 kHz	
Vertical scanning frequency	59.94 kHz	
Number of scanned lines	525 lines	
Frames per second	30 frames	
Synchronization	Internal	
Resolution	Horizontal: Min. 330 lines (center area)	
	Vertical: Min. 350 lines (center area)	
S/N ratio	46 dB (with ACG off)	
AGC	On (preset)	
Video output	VBS 1V(p-p) / 75 Ω composite signal	
External terminal	6-pin cable connector	
Power supply range	DC 4.8 $\tilde{5}.5$ V	
Power consumption	DC 5.0 V 160 mA (Typ.)	
Field angle	115.0°	
Max. aperture	F2.8	

Table 3.3: Specifications: Camera for Navigation

3.2.5 Camera for Outlet Detection

The task of this camera is different from the one used for navigation. It is mounted on the up/down DOF of the arm, so that it can detect the outlets and the plug. The camera used is an analog color camera with NTSC output. The characteristics of the camera are shown in Table 3.4.

The camera uses a lens whose maximum aperture is F2.8 and whose horizontal field angle is 45°. Unlike the previous camera, the narrow angle gives less deformation and better image resolution. These features are appealing for pattern recognition.

3.3 Computer Hardware

The robot has three PC104 computers on its top layer, which were chosen for their small form factor $(3.55" \times 3.755")$. A PC104 computer uses a connector with 104 pins to replace the ISA standard in a normal PC. There is also an extension called



Figure 3-10: Hardware architecture. The figure illustrates the connections between all the components of the robot. A detailed diagram of the Battery System is illustrated in Figure 3-11

Camera Specifications		
Model	GP-CX161-45	
TV Standard	NTSC	
Horizontal scanning frequency	15.734 kHz	
Vertical scanning frequency	59.94 kHz	
Number of scanned lines	525 lines	
Frames per second	30 frames	
Synchronization	Internal	
Resolution	Horizontal: Min. 330 lines (center area)	
	Vertical: Min. 350 lines (center area)	
S/N ratio	46 dB (with ACG off)	
AGC	On (preset)	
Video output	VBS 1V(p-p) / 75 Ω composite signal	
External terminal	6-pin cable connector	
Power supply range	DC 4.8 $\tilde{5}.5$ V	
Power consumption	DC 5.0 V 160 mA (Typ.)	
Horizontal field angle	45.0°	
Vertical field angle	33.8°	
Max. aperture	F2.8	
Focal length	4.6 mm	

Table 3.4: Specifications: Camera for Outlet Detection

PC104+, which is equivalent to the PCI standard.

These computers are running QNX as their operating system, one of them acting as a server and the other two as nodes. All three computers are connected to each other via an ethernet switch and to an external network via an ethernet-wireless converter (see Figure 3-10). The network in the Artificial Intelligence Laboratory is the external network.

The server also connects via a serial port to a network of controllers to control the arm motors and read the sensors.

3.3.1 QNX Server

The QNX server in the robot stores the programs to be run as well as the boot configuration of all the nodes in the network, provides network services, and acts as a bridge between the internal and external networks.

The computer used as a server has a 400MHz AMD K6-2 processor, 256MB RAM, and on-board I/O devices. A more complete description of the computer is presented in Table 3.5. Additionally, a 20GB 3.5" hard drive is connected using an IDE interface so that the server can store the operating system, configurations, and programs.

The on-board I/O devices used in this project are the two RS-232 serial ports and the ethernet card. One of the RS-232 ports connects to the B12 base making it possible to read the encoders of the motors as well as the charge of the battery. It is also possible to set position, velocity, and acceleration for the translational and rotational motors. The other serial RS-232 port connects to the network of controllers using an RS232 to RS485 converter. In this network, the server is the master controller and the other boards are slaves. The server is able to read the encoders from each of the motors, the values of the force sensors, and "the presence of voltage" sensor. The server also can set the PID gains, position, velocity, acceleration, and some additional parameters of the controllers.

The ethernet card in the server is connected to an ethernet-wireless converter using a UTP cable. This ethernet converter interfaces the ethernet board with a IEEE 802.11 PCMCIA wireless card. Therefore, the server can communicate with the wireless network available in the AI Laboratory.

A second ethernet card is attached to the server using the PC104 expansion. This card links to the 100/10Mps ethernet switch via a UTP cable. The ethernet switch also links to the other two nodes to form a QNX network which will be referred to as the internal network. Given that the server connects to both the internal and external networks, it is capable of behaving as a bridge between these two networks.

3.3.2 QNX Nodes

Each node in the robot is dedicated to processing images from the connected camera. The two nodes are computers with the same characteristics running as QNX nodes. A node comprises a computer and an external frame grabber.

The two computers are the same as the server: an AMD K6-2 400MHz processor

Computer specifications			
Processor	AMD K6-2 400MHz		
Cache	512KB Level 2		
RAM	256MB		
Ethernet card	dual speed 10/100 Mbps		
Serial/Parallel ports	2 COM + 1 LPT		
Disk On Chip	Socket 32 pin. Flash technology		
Keyboard and PS2 Mouse port	Available		
Video	PCI		
IDE expansion	Available		
Power Supply	5V		
Power Requirements	$+5V \pm 5\% @ 4.25A typ. (21.3W)$		
(32MB RAM, keyboard, mouse, Win95 and ethernet)			

Table 3.5: Computer specifications.

with 256MB RAM (again see Table 3.5 for more specifications). A 16MB flash disk on chip (DOC) is mounted in each computer. This DOC has the information necessary to boot the computer and to connect to the network. The connection of each node to the network is done through their respective on-board ethernet cards.

The external frame grabber, whose characteristics are shown in Table 3.6, is attached to the computer using a PC104 connector. Each node (computer and frame grabber) only requires 5V DC to operate.

Frame grabber specifications			
Form factor	PC104+		
Input video format	NTSC, PAL, SECAM, S-video		
Output formats	Color: YcrCb 4:2:2 and 4:1:1; RGB 32, 24, 16, 15; Monochrome: Y8		
Resolution	NTSC: 640 x 480; PAL/SECAM: 768 x 576		
Video noise	1 LSB (least significant bit)		
Power	5 VDC, 500 mA		

Table 3.6: Frame grabber specifications

3.3.3 Motor Controllers and Input/Output Board

The controllers used for the DC brushed motors are PIC-SERVO Motion Controller Boards. These boards are based on PIC microcontrollers which makes them small and have low power consumption. The controllers are networked using a full duplex RS-485 protocol. This characteristic allows for control of the motors using one serial connection to the RS-485 network. Some of the features of the controllers are:

- Position and velocity control
- Trapezoidal profiling
- 16-bit PID gains
- RS-485 serial interface
- Two-channel incremental encoders

In order to read the force and voltage-presence sensors, a PIC-I/O Multifunction Digital and Analog I/O Board is used. This board can read twelve digital inputs and three 8-bit A/D input channels. Additionally, this board can be connected to the motor controller network. The motors and the sensor inputs are interfaced with the QNX server through an RS-232 port.

The sensors are connected in the following way:

- The outputs of each FSR voltage divider are connected to a different A/D input channel.
- The output from the optocoupler, which detects the voltage presence in the plug, is connected to one of the digital inputs of the board.

3.4 Power System

The power system of the robot must supply energy to the controllers, the motors, the base, the cameras, the computers, the hard drive, the switch, and the ethernet converter. The diagram of the power system is shown in Figure 3-11. The controllers and the arm motors have low power consumption. Therefore, it is possible to use the base's lead-acid batteries to power them. The base supplies an output that starts at 12V DC when the batteries are fully charged. This voltage decreases as the batteries discharge. In order to provide constant voltage to the motors, the robot has a DC/DC converter which keeps the voltage constant. The controllers, on the other hand, have voltage regulators which makes it possible to directly connect them to the battery output.

To power the other devices, the robot uses four ED369 battery systems. These systems have outputs at 16 or 19V DC regulated or 12V DC unregulated. The charge of the system is 5800 mAh. The characteristics of this battery system are shown in Table 3.8. The distribution diagram is also shown in Figure 3-11.

One of the ED369 power systems powers the ethernet switch, the ethernet converter, the hard drive, and the cameras. The 19V DC output is transformed to 12V DC and 5V DC using DC/DC converters. The voltage requirements are shown in Table 3.7.

Power requirements			
Device	Voltage(V)	Current(mA)	Power(W)
Color camera	5	160	0.80
Hard drive	12	240	2.88
	5	500	2.50
Ethernet switch	12	600	7.20
Ethernet converter	5	400	2.00
	15.38		

Table 3.7: Power requirements

Each of the three computers is also powered by a ED369 battery system. The output voltage is converted to 5V DC regulated. The current required by each computer is 4.25 A when no serial port is used. The DC/DC converter used to drive this power is a 30 W TRI-MAG 980ZW1205S which outputs 5V DC regulated from an input between 9-18 V DC.

All these batteries must be recharged when the robot connects to an outlet.
Battery system specifications	
Model	ED369
Battery cell	High density Lithium Ion Battery
Capacity	5400 mAh
Power input	DC 15V $\tilde{1}8V$ / 1.0 A min
Power output	16V / 2.5A
	19V / 2.2 A
	10V Ĩ2V / 3.0A
Dimensions	$19.7~\mathrm{cm} \times 10.1~\mathrm{cm} \times 2.8~\mathrm{cm}$
Weight	620 g
Charging time	6 h

Table 3.8: Battery system specifications

The B12 base has a recharging circuit on-board for its lead acid batteries. This circuit operates properly with a regulated source that provides 24V DC/1.5A. This energy is obtained from a switched power supply that converts 110V/1.5A AC to 24V/2.5A DC regulated. The 110V AC is obtained from the robot's plug when it is connected to a power outlet. Furthermore, the power supply is mounted on board. This is possible because a switched power supply is much lighter than a linear one.

All the ED369 battery systems are rechargeable using an external regulated power supply. As in the B12 recharger, the power is provided by an on-board switched power supply. The power supply converts 110 V AC to 15V DC. However, the battery system is not capable of both recharging its batteries and supplying energy to the load at the same time. Furthermore, the battery system is inactive for about one second when it switches from charging to supplying energy to the load. This situation occurs when the robot is disconnecting from the power outlet. Therefore, an external circuit that takes care of these commutations is connected to the battery system.



Figure 3-11: Power system block diagram.

Chapter 4

Vision Algorithms

4.1 Introduction

In this chapter we describe the algorithms implemented for obstacle detection and outlet detection. Obstacle detection runs on one of the nodes and outlet detection on the other, so that each algorithm can take full advantage of their respective processor.

4.2 Obstacle detection

The method implemented for obstacle detection is similar to the works of Horswill (1993) and Lorigo et al. (1997). We assume the same constraints as in the latter paper. That is:

- The robot is on flat terrain. Therefore, the obstacles that are farther away appear in a higher position in the image than the obstacles that are closer to the robot.
- Obstacle boundaries are visible.
- The floor near the robot has no obstacles.

The images acquired by the camera are color images with a resolution of 128×128 pixels. During processing, an image is first transformed into a gray-scale image and



Figure 4-1: Obstacle detection. (A)Initial image. (B) Averaged image using windows. (C) Differentiating the averaged image. (D) Image of obstacles obtained from the values of image (C) that are above the threshold.

an integral image is computed as described later. Then a window with a height of 10 pixels and a width of 20 pixels is moved from the bottom to the top by placing each consecutive window one pixel above the previous one. The average of the pixels inside one window is calculated and subtracted from the previous one. If this value is above a threshold, an obstacle is detected at that location. This works based on the assumption that the average of each adjacent window should be similar unless there is an obstacle present. The same process is repeated for each column in the image in which the window can fit. Thus the number of columns scanned is 128 - 20 + 1 = 109. All the space in front of the robot is scanned in this manner and the height of each obstacle detected is stored in an array. The height of each obstacles can be at most 128 - 10 + 1 = 119.

The average of each window is computed quickly because an integral image is used. The integral image is defined in Viola & Jones (2001) as:

$$II(x,y) = \sum_{i_x < x, i_y < y} I(i_x, i_y)$$
(4.1)

where $I(i_x, i_y)$ is the original image and II(x, y) is the integral image. This operation can be done in one pass over the image using the following recurrences:

$$S(x,y) = S(x,y-1) + I(x,y)$$
(4.2)

$$II(x, y) = II(x - 1, y) + S(x, y)$$
(4.3)

Computing the average value of the window requires four memory accesses to its corners in the integral image. In Figure 4-1 we can see the results of the algorithm applied to an image.

4.3 Outlet detection

To solve the problem of detecting an outlet in an image, we use the method presented in Viola & Jones (2001). In this method the object to be recognized is represented by a few of its most relevant features. A feature is defined in terms of operations between rectangular areas. The configuration of those rectangular areas determines a class of feature.

The classes of features considered in our case are: a two-rectangle feature oriented in either the horizontal and vertical direction, a three-rectangle feature with similar choice of orientation, and a four-rectangle feature. These classes are illustrated in Figure 4-2. The value of a feature is obtained by subtracting the average value of the image within the black rectangle from the one within the white rectangle.

In order to find out the most relevant features of an outlet, many samples are collected. These samples include outlet and non-outlet images, called positive and negative examples respectively.

For each sample, the value of all possible features drawn from each class is calcu-



Figure 4-2: Classes of features. The features are shown inside a window. (A) Two-rectangle horizontal. (B)Two-rectangle vertical. (C) Three-rectangle horizontal. (D) Three-rectangle vertical. (E) Four-rectangle.

lated and stored in a vector. For each one of these features a classifier is then trained. A classifier is characterized by a threshold and a sign. The threshold separates the positive and negative examples and the sign indicates if the positive examples are above or below the threshold value.

It is not expected that one of these classifiers alone will perfectly separate positive from negative examples. However, a combination of some of them may do a sufficient job. In order to find which classifiers to combine for improving the final classification, a variation of the Adaboost (Freund & Schapire 1995) algorithm is used. In this algorithm, the first classifier chosen is the one that yields the lowest classification error on the samples, where error is defined as the number of misclassified samples.

In order to improve the final classification, the next classifier must do a better job on the samples than the first did. To implement this, the samples are weighted. The weights of the samples misclassified by the first classifier are made greater than those of the samples correctly classified. These weights are normalized so that they behave as a distribution. Using these weighted samples, new classifiers are trained for each feature and again the classifier with the lowest error is chosen. This procedure is known as boosting. Since there is a classifier for each feature, by selecting the best classifiers we select the most relevant features.

Each classifier selected also has a weight associated with it that depends on the number of classifications it got correct. This weight is used to combine all the selected classifiers. The output of each classifier (0 or 1) multiplied by its associated weight are added. The answer divided by the sum of these weights is compared to a threshold to decide whether or not the image contains an outlet. This is the output of the outlet detector. The details of this algorithm are presented in Figure 4-5.

For our implementation, 846 positive examples (outlet samples) and 1400 negative examples (non-outlet samples) were collected. These samples are 28×20 gray-scale images extracted from pictures taken in the laboratory. The outlets were centered in the window and labelled by hand using a program written in Matlab. All the samples were normalized with respect to their variance in order to reduce the effects of illumination. Some of the samples are shown in Figure 4-3.

After collecting samples, all the features were calculated for each sample. The class of feature illustrated in Figure 4-2 (D) which was not included in Viola & Jones (2001) is also considered.

The number of features values for each sample is large. For example for the total number of features calculated for a 28×20 image is 153,398. Since we have 2246 samples, the memory requirement is 344,531,908 floating point numbers or 1.3 GB. Therefore, it was necessary to swap the data between memory and disk often on a PC with 256 MB of memory.

For each of these features a classifier was trained. The classifier used consists of two Gaussian functions fitted to the positive and negative samples respectively. The intersection of the two Gaussian functions determines the threshold, and the difference of the means determines the sign.

This process, although simple in principle, takes considerable time due to the amount of features. The training algorithm was coded in Matlab. For 2246 samples,



Figure 4-3: Positive examples. Some of the examples used for training.

150 iterations took about 38 hours. Some of the features that the training program found are shown in Figure 4-6.

Having found the relevant features, the search for an outlet in an image is done as follows. First, an integral of the image and an integral of the squared image are calculated using the method describe in Section 4.2. Then, the image is scanned using a 28×20 window for the scale 1:1. Each window is variance normalized and then the classifiers are applied to it. The integral of the image is used to compute the value of the features associated with the classifiers. The integral of the image squared is used to compute the variance of the window according to the identity $\sigma^2 = E[x^2] - m^2$. Where *m* is the mean and *x* represents the pixels of the window.

This scanning is repeated at several scales. However, with this method it is not necessary to downsample the image. Instead the size of the window and its features increase without affecting the evaluation of the feature. This is because the time taken to compute the average over a window with an integral image is independent



Figure 4-4: Negative examples. These samples were obtained randomly from images that did not contain outlets.

- Given the examples $(x_1, y_1), ..., (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where *m* and *l* are the number of negatives and positives respectively.
- For t = 1, ..., T:
 - 1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$ so that w_t is a probability distribution.
 - 2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) y_i|$.
 - 3. Choose the classifier, h_t , with the lowest error ϵ_t .
 - 4. Update the weights: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

where $\alpha_t = \log(\frac{1}{\beta_t})$

Figure 4-5: Boosting algorithm. Reproduced from Viola & Jones (2001) without permission.

of the size of that window. The program searches at 7 scales starting with a scale of 1 and increasing by 20%.

In order to achieve an accurate detection around 30 features are needed. Evaluating these features in every window at each scale takes enough time to make this method inadequate for real-time. To reduce the time required for evaluation, a cascade of detectors is used. In this cascade, only the windows positively classified by a detector in one stage are evaluated by the classifier in the next stage. Those windows that pass all the detectors in the cascade contain an outlet.

The time response of a cascade detector improves because the first detector uses only two features instead of 30. More features are evaluated only for those windows that pass the first detector, which are much fewer than the total number of windows



Figure 4-6: Some of outlet features extracted by the algorithm.

in the image. The second, the third and the fourth stages use 2, 5 and 20 features respectively.

An example of the result is shown in Figure 4-7.



Figure 4-7: Example of the detector working

Chapter 5

Support Software

5.1 Overview

This chapter describes some features of the operating system used and the processes that were written to support the software architecture of the robot.

5.2 Introduction

In order to take advantage of the three computers available, programs run on different nodes. However, to create behaviors it is necessary to communicate among these programs. The communication among programs can be complicated and slow. To make this more efficient, QNX has primitive functions for communication. Using these functions, code has been written to allow other programs to access the B12 base, the arm, and the frame grabbers. These programs known as daemons, run in the background, listening for requests from another program (referred to as a client). The daemons are capable of accessing the hardware and supplying the information requested. Daemons take advantage of the idle time of the processor, access the hardware asynchronously (if needed), reply to requests with the data acquired in the latest update, and reply to requests from different processes.

In this chapter we will start describing some features of the operating system, continuing with a description of the following daemons: base B12, arm, obstacle detection, and outlet detection.

5.3 Operating System

The QNX operating system has features needed for real time applications. These features are multitasking, preemptive scheduling, and fast context switching. To accomplish such tasks, QNX was designed using a microkernel architecture and messagebased interprocess communication.

The QNX microkernel is a small kernel that is dedicated to message passing and scheduling. The microkernel is called from a process or a hardware interrupt. Therefore, all QNX services are standard QNX processes. The device drivers are also standard QNX processes. The advantage of handling services and device drivers as standard processes is that they can be started or stopped without affecting the operating system.

In order to provide communication among processes, QNX uses message passing. A message is a packet of bytes whose content is defined by the sender and the receiver processes. The basic primitive functions for communication are: Send(), Receive(), and Reply(). The message passing is independent of the location of the processes. They can be in the same processor or in different processors.

Finally, the daemons mentioned above are implemented as QNX processes. These processes are identified by global names, thanks to the QNX Network Manager. This service registers and locates global names.

5.4 Base B12 daemon

This daemon connects the base B12 to a client. When this daemon starts, it registers a global name so that any client can send requests.

A client communicates with the daemon using predefined messages that include the parameters needed to control the base B12. When the daemon receives these messages, it sends the corresponding commands to the base in ASCII code via serial port. The daemon accesses a serial port in a raw mode at 9600 bps. That is with no flow-control and no terminal functions.

Additionally, the daemon polls the battery voltage with an interval of one minute. It also reads periodically the value of the encoders of the base. Therefore, when any of these values are requested by a client, the value stored in memory is immediately sent. This gives better time response to the client.

5.5 Arm daemon

This daemon connects the motor controllers of the arm to a client. The daemon communicates with the motor controllers via a serial port in raw mode, at the speed of 19200 bps. The motor controllers have a proprietary protocol for communication. This protocol is coded in a library for Windows that was translated for QNX.

As in the previous section, the client uses a predefined set of messages and parameters to communicate with the daemon. The daemon identifies the message and sends the corresponding command to the motor controllers.

The client that accesses this daemon can also obtain the values of the force and voltage presence sensors.

5.6 Obstacle detection daemon

This daemon uses YARP (Fitzpatrick & Metta forthcoming) connections to communicate with other modules. In order to establish a YARP connection, it is necessary to define input and output ports in the modules to connect.

This daemon has an BGR image as input port and a vector as an output port. The program waits for a connection with a module that sends images. When an image is received, the algorithm described in section 4.2 is applied. The output of this algorithm is a vector containing the distances to obstacles detected.

YARP also provides a module (grabber) that communicates with a frame grabber and outputs images. The output of the grabber is connected to the input of this daemon. The connection of the output of this daemon is explained in chapter 6.

In order to connect to the daemon output port to a client input port, both ports must be of the same kind, in this case a vector. For debugging purposes it is possible to define an image as a second output of the daemon. This output is usually connected to a YARP display for visualization.

5.7 Outlet detection daemon

This daemon recognizes outlets using the method described in section 4.3. As in the daemon described in section 5.6, YARP connections are used to communicate with other modules.

An image is defined as an input port for this daemon. The input port is connected to the output of a YARP grabber module to receive images. The outlet detection algorithm is applied to each image. Its result is stored in a structure that contains the position, the scale, and the degree of certainty of a detected outlet. This structure is defined as an output port of the daemon.

Chapter 6

Software: Design and Implementation

6.1 Overview

This chapter describes the software architecture used to control the robot. It starts with a global description of the architecture and the strategy. The chapter continues with a detailed description of the implementation of the software.

6.2 Introduction

In the field of Artificial Intelligence, the design of "brains" for robots has often been inspired by neuroscience and biology. One of the main works is by Brooks (1986), in which he proposes to arbitrate several behaviors based on subsumption. Some behaviors inhibit the output of other modules, but all the modules are active at the same time. Another characteristic of the subsumption architecture is the fact that it directly connects sensors with actuators. In other words, not all the sensors report to a central information processor that decides what sequences of actions to execute. Subsumption architecture proposes distributed processing instead of central processing. This model has been corroborated by biological studies, such as Koopowitz & Keenan (1982). In these experiments, the brain of a flatworm (*Notoplana acticola*)



Figure 6-1: Software architecture. The block diagram illustrates agents, daemons and shared data structures. The shared data structures are also called connections. All the agents are executed in the server (N1). The daemon blocks include the number of the node where they are executed. They are described in Chapter 5

was removed but the animal still showed its ability to acquire food. However, the flatworm lost its ability to determine a satiation point at which no more food was necessary.

By applying this example to our model, the architecture proposed consists of many agents. Each agent is able to receive inputs from any other agent. The agent executes simple tasks that connect sensors with actuators in a direct way. The combination of these agents yields complex behaviors.

A block diagram of a software architecture based on agents is shown in Figure 6-1, where each arrow represents a shared data structure between agents. At each iteration, all the shared data structures are updated and then the code of each agent is executed.

This software architecture implements the behaviors wandering around, looking for an outlet, and connecting to an outlet which are explained in the following sections.

6.3 Behavior: Wandering Around

This behavior moves the robot around the laboratory. It is activated when the energy level is high enough for operation. The behavior turns and/or translates the robot in a direction chosen to avoid collisions with the obstacles on the floor. This behavior uses the *base motor control* and *obstacle avoidance* agents.

6.3.1 The Base Motor Control Agent

The *base motor control* agent is a client of the base B12 daemon (see section 5.4); therefore, it is capable of sending commands to the base B12 and reading its encoders and battery voltage.

As described in the following sections, many agents send motion commands to this agent. The commands can be either rotations or translations. However, only one rotational and one translational command is selected and sent to the base B12. This selection process, known as arbitration, can be implemented using several criteria such



Figure 6-2: Wandering around behavior block diagram.

as winner-take-all, action-selection, voting, etc. A winner-take-all criteria is used in this particular implementation.

Each of the agents that connect to the Base Motor Control agent sends a command and an associated priority. The command with the highest priority is sent to the base B12. This selection process is done independently for translational and rotational commands.

In many cases, agents repeatedly sent the same command to the base, for example, a translation using a constant speed. The motion of the base will not change when the command is received the second time. This agent detects this situation and sends only those commands that change the current motion of the robot's base. This optimization reduces the serial port traffic and consequently reduces the power consumption.

This agent also reads and stores the values of the encoders (translational and rotational) and the voltage battery. This data is shared with other agents as illustrated in figure 6-1.

6.3.2 The Obstacle Avoidance Agent

This agent connects to the obstacle detection daemon (section 5.6) to obtain the information about the obstacles. This information consists of a vector where the heights of the obstacles detected are stored (section 4.2). This vector is divided into three regions (left, center and right) and the minimum point of each region is determined.

If any of these three points is less than 23 pixels, the robot stops translating because the obstacle is too close. Otherwise, the minimum value of the three regions is multiplied by a gain and sent to base motor control agent. The rotation magnitude is calculated as the difference between the left and right minimum point multiplied by a gain.

The priority of the stopping command has the highest value in this architecture. The priority of the other commands depends on the voltage level of the robot's batteries.

When the voltage level of all the batteries is above 11V, there is enough energy to make the **wandering around** behavior predominant. This is achieved by rising the priority of the motor commands of this agent. On the other hand, if the voltage level is below the 11V the priority is set to the lowest value. This makes it possible for motor commands from agents other than the *obstacle avoidance* agent to be executed. As a consequence the other behaviors, that try to acquire energy, can be executed.

6.4 Behavior: Searching for an Outlet

The **searching for an outlet** behavior is activated when the robot needs to recharge its batteries and an outlet is not detected. The purpose of this behavior is to help find an outlet. The agents used by this behavior are shown in Figure 6-3. They are: *detect outlet, base motor control* (see section 6.3.1), and *scanning*.



Figure 6-3: Search for an outlet behavior block diagram.

6.4.1 The Outlet Detection Agent

The outlet detection agent is a client of the outlet detection daemon (see section 4.3). This agent receives the position, the scale and the degree of certainty of a detection.

6.4.2 The Scanning Agent

The scanning agent reads the voltage level of the batteries. When any voltage value is under 11V and no outlet is detected, this agent stops the robot's translation and rotates the robot 90° to one side. Later, it rotates the robot in the opposite direction in steps of 18°. The rotation stops after covering 180° in 10 steps.

If at any time an outlet is detected by the *detect outlet* agent (see section 6.4.1), the rotation is stopped and no more commands are sent to the base. The agent remains in this state until either two minutes have passed or the robot has translated a given distance. It is expected that the *obstacle avoidance* agent will be chosen by arbitration process of the *base motor control* agent. Therefore, the robot will move before a new scanning process happens.

6.5 Behavior: Connecting to an Outlet



Figure 6-4: Connecting to an outlet behavior block diagram.

When an outlet has been located by the *detection outlet* agent, the robot tries to connect to the outlet. The first thing that the robot does is approach the outlet until it is close enough to be reached by the arm. In this position, the plug is aligned with the outlet and the arm is extended for plugging.

The behavior described above is implemented with the following agents: *outlet* detection, arm motor control, orientation, approaching, reaching, plugging, unplugging, and avoidance obstacles.

6.5.1 The Arm Motor Control Agent

This agent is similar to the *base motor control* agent described in section 6.3.1. The *arm motor control* agent is a client of the arm daemon (see section 5.5) and, there-

fore, it is capable of sending commands to the network of motor controllers used for controlling the arm.

As we can see in Figure 6-1, there are several agents connected to the arm motor control agent. These agents either send commands to the motors or read sensors values.

A command is sent together with a priority value which is used to arbitrate the inputs. The command with the highest priority is executed. The arbitration process is applied to the up/down and in/out commands independently. As for the *base motor control* agent, this agent also optimizes the serial traffic by sending only the commands that change the motion of the arm.

This agent also reads the values of the encoders, the force sensor and the "voltage presence" sensor from the arm daemon.

6.5.2 The Orientation Agent

The *orientation* agent controls the base and the robot's arm to place a detected outlet in the center of the image. This condition aligns the robot's camera with the outlet. Due to the configuration of the arm, when the camera is aligned with the outlet, the plug is also horizontally aligned and the vertical distance between the plug and the outlet is a known value.

The information about a detected outlet is obtained from the *outlet detection* agent. The horizontal difference between the outlet and the center of the image is used to rotate the base. Likewise, the vertical difference between the outlet and the center of the image is used to move the arm up or down. The commands to the base and arm are sent through the *base* and *arm motor control* agents respectively.

6.5.3 The Approaching Agent

The *approaching* agent moves the robot towards a detected outlet. The information about the detected outlet is obtained from the *outlet detection* agent.

This agent sends a translation command to the base motor control agent when

both the detected outlet is centered on the image and the scale of the outlet is less than three times the size of the minimum outlet detected.

When the size of the outlet detected is equal or greater than three times the minimum detectable size, a stop command is sent because the outlet is close enough to be reached by the robot's arm. At this point, the *avoidance obstacle* agent also stops the robot because the wall is detected as an obstacle too close to the robot. This fact adds robustness to the robot behavior.

6.5.4 The Reaching Agent

The *reaching* agent receives the position and the scale of a detected outlet from the *outlet detection* agent. When an outlet is detected at a given scale and it is in the center of the image, the arm is extended as much as possible. The action of extending stops when either the force sensor detects contact or when the arm has been completely extended. The force sensor and arm encoders values are obtained from the *arm motor control* agent.

6.5.5 The Plugging Agent

The *plugging* agent detects if the plug is in contact with a surface (i.e. a wall) and if there is electrical contact. This information is read from the force and "voltage presence" sensors available from the *arm motor control* agent.

When there is both physical and electrical contact, it is assumed that the robot successfully connected to an outlet. Therefore, this agent stops all the motion of the robot until its batteries have been recharged. The stop command is constantly sent to all the motors of the robot. When the robot's batteries are recharged, this agent stops sending commands to the in/out motor of the arm to allow the other agents (i.e., *unplugging*) to act.

If there is only physical contact and the arm was extending before this contact, it is assumed that the robot missed the outlet. This situation can arise when the robot orientation is not perpendicular to the wall where the outlet is mounted. However, since the passive compliance mechanism can help to solve this situation, the information about obstacles is consulted to find out the orientation of the wall. If the wall is not perpendicular to the robot's orientation a small rotation is performed. The direction of this rotation is chosen to increase the misalignment between the robot and the outlet but it helps to align the plug (a passive compliance mechanism) with the outlet.

If the wall is perpendicular to the robot orientation or no wall is detected in front of the robot, this agent stops all the motors of the robot, assuming that the robot has collided with an obstacle.

The obstacle information is obtained through a connection with the *avoidance obstacle* agent.

6.5.6 The Unplugging Agent

The unplugging agent retracts the arm and then turns the robot 180° . This agent assumes that if the arm is extended it is because a plugging trial has happened. To determine the state of the arm, the encoder of the in/out motor is read. If the encoder value is different from zero, a command is sent to the motor to retract the arm. However, this command is executed only when the *plugging* agent, which has a higher priority, stops sending commands to the in/out motor.

Once the arm is completely retracted, a rotation command is sent to the base. The completion of the rotation is detected by reading the rotational encoder of the base. The information about the base and arm encoders is obtained from the *base* and the *arm motor control* agents.

Chapter 7

Conclusion and Future Work

7.1 Analysis and Results

The learning algorithm used for outlet recognition chooses a feature and an associated weight in each iteration (see Figure 4-5). The features selected were used to build a one-stage classifier. This classifier was tested on an evaluation set of 20 images. The algorithm worked well at different scales when the image did not contain too many details. In images such as that of the corridor, in which many details of the ceiling and pictures on the wall show up, the classifier yields many false detections, especially at greater scales.

Many 28×20 windows of various images were collected and incorporated into the training set. The features obtained after running the learning algorithm are shown in Figure 7-1. As we can see the first features obtained from each set were basically the same. Unlike in Viola & Jones (2001) the interpretation of the features obtained is not evident. However, the interpretation of features obtained in the following iterations in a set are more evident. For example, in Figure 7-2(A) the feature chosen finds the vertical edge the outlet's face plate, while the feature in Figure 7-2(B) tests for symmetry of the lower outlet.

To eliminate false detections, more features have to be evaluated which implies a longer evaluation time. In order to maintain a reasonable evaluation time and to use more features, a cascade of four classifiers was implemented. The classifiers of the



Figure 7-1: Features chosen by the learning algorithm. (A) First feature from the first set. (B) Second feature from the first set. (C) First feature from the second set. (D) Second feature from the second set.



Figure 7-2: Additional features chosen by the learning algorithm.

cascade evaluate 2, 2, 5, and 20 features. The thresholds for each stage are 1.0, 0.5, 0.25, 0.8. These values were fine-tuned by hand after visualizing the response of the algorithm on images of the corridor.

In order to ensure good time response, the outlet recognition algorithm and the image grabber are the only processes run on Node 3 of the QNX network. However, even with the whole processor available, a careful implementation of the algorithm is needed in order to achieve real time response.

The original implementation scanned a 128×128 image pixel by pixel using a 28×20 window for the scale 1:1. Two features were evaluated in each window which had been variance normalized. The complete scanning process lasted 0.7s. This time response was not acceptable for our implementation. In order to reduce this time response, the dimensions and position of each feature used for the classifiers at each scale were precomputed and stored. Also, the function used in the main loop was compiled as "inline". The only operations done in this main loop of the algorithm are a "switch" to select the kind of feature and the computation of the offset of the window. Since the "switch" is implemented as a PC-relative jump, most of the

processing time is spent in adding the offset of the window to the coordinates of a feature and accessing the integral image. This results in a 0.6 s decrease in response time (to 0.1s) for evaluation at one scale. Using this implementation, the evaluation time of a cascade of four classifiers at seven scales is 0.3s. The evaluation time remains small because only a few windows that are positively classified by the first stage of the cascade are evaluated by the subsequent stages. This is the advantage of a cascade of classifiers. This time response is good enough to track an outlet when the robot is moving towards it.

The minimum size of an outlet considered in the recognition algorithm fits comfortability in a 28×20 window. This makes it possible to detect outlets that are about four feet away from the robot. Assuming that an outlet is mounted on a flat wall, we use the angle between the wall and the robot's orientation as reference to further evaluate the recognition algorithm. This angle is 90° when the robot is directly orientated toward an outlet. An outlet is recognized when this angle is between 60° to 120°. The slight deformations of the outlet between these angles have been learned by the algorithm.

The outlet recognition algorithm was initially trained with samples obtained from a digital camera. The parameters obtained from these samples did poorly when applied to images acquired by the analog camera on the robot's arm. Therefore, new samples obtained with the analog camera were used to train the outlet recognition algorithm. This was not an expected problem since both cameras used CCD sensors and DSP's for preprocessing. However, after analyzing the quality of the image from both cameras, we noticed that an image from the analog camera is a lot noisier than one from the digital camera.

The obstacle detection algorithm finds objects in a wide field of view. It can detect obstacles that are as close as 7 inches. In the current implementation objects that are approximately 1.5 inch \times 1.5 inch can be detected. This resolution is given by the size of the scanning window and the displacement between the scanned columns. The current displacement between scanned columns is one pixel obtaining 109 values (see Section 4.2). This number can be reduced, sacrificing resolution. For example, Martin (2001) uses only 12 columns to guide the robot. The obstacle detection algorithm runs alone on Node 2, therefore it is possible to have high resolution without reducing the time response of the system.



Figure 7-3: Example 1. The colors that represent to each of agents are shown in the top of the figure. The first plot from the top illustrates the agents that won access to each of the motors. The name of each motor is in the vertical axis of this plot. The next plots present the values of the up/down encoder, in/out encoder, force sensor, and rotation encoder. The horizontal axis represents the iteration number.

The software architecture implemented uses agents that compete for accessing the motor control agents. The arbitration of these agents is based on priorities. In the following two examples, the interaction of these agents can be observed.

In the first example, the robot is about 5 feet in front of a wall with an outlet. Figure 7-3 illustrates the results recorded. In the top of this figure, the agents that won access to the rotation, translation, up/down, and in/out motors are shown. The first two motors control the base and the other two control the arm. Figure 7-3 also illustrates the readings from the force sensor and the motor encoders.

In this sequence we can observe that the *obstacle avoidance* agent controls the rotation and the translation motors at the beginning. Later the *approaching*, the *orientation*, and the *obstacle avoidance* agents alternate the control of the translation motor. The *orientation* agent wins when the vertical distance between the outlet detected and the center is beyond a threshold. Otherwise, the *orientation* agent controls the translation motor. When the robot is too close to the wall, we can observe that *obstacle avoidance* agent takes control to avoid hitting the wall. During this period, the up/down motor is controlled by the *orientation* agent, which moves to vertically center the outlet in the image. The plot of the up/down encoder shows this movement.

Once the outlet is centered in the image, the *reaching* agent takes control of the up/down, in/out, and rotation motors. The rotation motor is stopped, the up/down motor is moved upwards, and then the arm is extended. These movements are shown in the encoder plots.

When the plug hits the outlet, (see the force sensor reading), the *plugging* agent stops the in/out, up/down and rotation motors. It also calculates the orientation of the robot with respect to the wall to rotate the base if needed. In this example, the base is perpendicular to the robot's orientation; therefore, no rotation is needed. Since there is no more to do to help the plugging, the *plugging* agent stops sending commands to the motors. At this point, the *unplugging* agent retracts the arm and later moves it down. Finally, it rotates the robot.

Another example of the robot moving towards an outlet is illustrated in Figures 7-4. The trajectory of the robot, reconstructed from the readings of the encoders, is shown in Figure 7-5. In this example the rotation of the *unplugging* agent was reduced to 10°.

Two versions of this robot were constructed. In the first one, a 600 MHz laptop computer with Windows as its operating system was used. However, the computer alone was not enough to run all the algorithms in real-time and the operating system crashed often. Learning from this experience, in the second version, which has been



Figure 7-4: Example 2. The colors that represent to each of agents are shown in the top of the figure. The first plot from the top illustrates the agents that won access to each of the motors. The name of each motor is in the vertical axis of this plot. The next plots present the values of the up/down encoder, in/out encoder, force sensor, and rotation encoder. The horizontal axis represents the iteration number.

described in this thesis, the computation on board was changed to three 400 MHz PC104 computers running QNX as their operating system. This operating system works in real-time and is very robust. Its only drawback is that it supports limited compatible hardware. This was one of the main reasons to use analog cameras instead of digital ones.

The batteries used for powering the robot are enough to power the three computers for about 2 hours. However, when all the parts of the robot were operating, the power requirements increased and the batteries only can support the robot for about 1.5 hours. The components that draw the most power, after the computers, are the translational motor of the base and the serial ports. Regrettably, the RS232 ports



Figure 7-5: Trajectory of the robot. The robot approaches the outlet and later goes away. This trajectory was reconstructed form the readings of the translational and rotational encoders. The detail of the agents involved in the control of the robot are illustrated in Figure 7-4.

implemented in the PC104 computers require a lot of energy.

7.2 Conclusions and Recommendations For Future Work

The learning algorithm in Viola & Jones (2001) is described as a general algorithm for detecting objects. Frontal face detection is used as an example to demonstrate its performance. This work has shown that the learning algorithm also works well when applied to a different object, e.g., a frontal outlet. A detector constructed using this learning algorithm can work faster than other known algorithms, however, it has to be carefully implemented.

The obstacle detection algorithm does a very good job of finding obstacles in the environment. However, there are cases in which it fails. For example, it identifies shadows and changes in color and texture on the floor as an obstacle. These failing conditions are discussed in Horswill (1993), Lorigo et al. (1997), and Martin (2001). Using additional information from stereo vision or other ranging methods, these problems can be solved. Though there are many benefits of stereo vision, its main drawback is its heavy computational requirements. This is a critical factor in mobile robots where adding computers implies adding batteries.

The behaviors were implemented using agents which run in parallel and compete for access to the motor control agents. This configuration has the advantage that new agents can be added to create more behaviors without changing the existent implementation. Besides, some of the agents are used to generate more than one behavior. For example, the *obstacle avoidance* agent described in Section 6.3.2 is used by all three behaviors. This also increases robustness since some behaviors will help each other. For example, the *approaching* agent guides the robot towards an outlet until it is reachable by the robot's arm then stops the robot. However, if the detected outlet is an object on the wall that looks like an outlet from far away, the robot will continue approaching and will hit the wall. Nevertheless, due to the fact that the *obstacle avoidance* agent is running in parallel, it will detect the wall and stop the robot avoiding the collision. The stop command sent by the *obstacle avoidance* agent will have the highest priority to overwrite the commands of any other agent given the importance of the action to execute. Therefore, this architecture is adequate to increase the number of behaviors performed by the robot.

Energy is still a big issue in mobile robotics for two main reasons. First, most of the hardware with enough processing power for performing tasks such as vision is neither power-aware nor low-power. Also the kind of batteries capable of providing enough electrical power require a careful and usually long recharging time.

Finally, having an autonomous self-powered robot provides a simple example of an organism which can be used to further study living organisms.

Bibliography

- Angle, M. C. (1991), Design of an Artificial Creature, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Arbib, M. A. (1992), 'Schema Theory', The Encyclopedia of Artificial Intelligence pp. 1427 – 43.
- Arkin, R. (1998), Behavior Based Robotics, The MIT Press, Cambridge, MA.
- Breazeal, C. (2000), Sociable Machines: Expressive Social Exchange Between Humans and Robots, PhD thesis, MIT Department of Electrical Engineering and Computer Science.
- Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', IEEE Journal of Robotics and Automation 2, 14–23.
- Brooks, R. A. (1989), 'A Robot That Walks: Emergent Behavior from a Carefully Evolved Network', *IEEE Journal of Robotics and Automation* 2, 253–262.
- Brooks, R. A. & Flynn, A. M. (1989), 'Fast, Cheap and Out of Control: A Robot Invasion of the Solar System'.
- Fitzpatrick, P. & Metta, G. (forthcoming), YARP, Technical report, MIT, Artificial Intelligence Laboratory.
- Freund, Y. & Schapire, R. E. (1995), A decision-theoretic generalization of on-line learning and an application to boosting, in 'In Computational Learning Theory: Eurocolt '95', Springer-Verlag, pp. 23–37.

- Golombek, M. P., Cook, R. A., Economou, T., Folkner, W. M., Haldemann, A. F. C., Kallemeyn, P. H., Knudsen, J. M., Manning, R. M., Moore, H. J., Parker, T. J., Rieder, R., Schofield, J. T., Smith, P. H. & Vaughan, R. M. (1997), 'Overview of the Mars Pathfinder Mission and Assessment of Landing Site Predictions', *Science* 278, 1743–1748.
- Hogan, N. (1985), 'Impedance Control: An Approach to Manipulation: Part I– Theory', Transacions of the ASME, Journal of Dynamic Systems, Measurement and Control 107, 1–7.
- Horswill, I. (1993), Polly: A Vision-Based Artificial Agent, in 'Proceedings of the 11th National Conference on Artificial Intelligence', AAAI Press, Menlo Park, CA, USA, pp. 824–829.
- Koopowitz, H. & Keenan, L. (1982), 'The Primitive Brains of platyhelminthes', Trends in Neurosicenes 5, 77–79.
- Lorigo, L. M., Brooks, R. A. & Grimson, W. E. L. (1997), Visually-Guided Obstacle Avoidance in Unstructured Environments, *in* 'Proceedings of the IEEE Conference on Intelligent Robots and Systems', Grenoble, France.
- Maes, P. (1989), The Dynamics of Action Selection, in 'Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Detroit, MI', pp. 991 – 97.
- Maes, P. & Brooks, R. A. (1990), 'Robot Insect Societies'.
- Martin, M. C. (2001), Simulated Evolution of Robot Perception, PhD thesis, Carnegie Mellon University, The Robotics Institute.
- Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R. & Litwin, T. (1995a), Mars
 Microrover Navigation : Performance Evaluation and Enhancement, in 'Proc.
 IEEE Int'l Conf. Intelligent Robots and Systems', Vol. 1, pp. 433–440.
- Matthies, L., Kelly, A. & Litwin, T. (1995b), 'Obstacle Detection for Unmanned Ground Vehicles: A Progress Report'.
- Nourbakhsh, I., Bobenage, J., Grange, S., Lutz, R., Meyer, R. & Soto, A. (1999), 'An Affective Mobile Educator with a Full-time Job', Artificial Intelligence 114(1 -2), 95 – 124.
- Osuma, E., Freund, R. & Girosi, F. (1997), Training support sector machines: an application to face detection, in 'In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition'.
- Pomerleau, D. A. (1993), Neural Network Percerption for Mobile Robot Guidance, Kluver Academic Publishers, Norwell, MA.
- Raibert, M. & Craig, J. (1981), 'Hybrid Position/Force Control of Manipulators', Transacions of the ASME, Journal of Dynamic Systems, Measurement and Control 102, 126–133.
- Real World Interface Inc. (1994), 'B12 Base Manual'.
- Rowley, H., Baluja, S. & Kanade, T. (1996), Human Face Detection in Visual Scenes, in 'Advances in Neural Information Processing Systems 8', pp. 875 – 881.
- Salisbury, J. K. (1990), 'Active Stiffness Control of a Manipulator in Cartesian Coordinates', 19th IEEE Conference on Decision and Control.
- Santos-Victor, G., Sandini, G., Curotto, F. & Garibaldi, S. (1995), Divergent Stereo in Robot Navigation: Learning from Bees, in 'Proc. IEEE CS Conf. Computer Vision and Pattern Recognition'.
- Sung, K.-K. & Poggio, T. (1994), 'Example Base Learning for view-based face detection', A.I. Memo.
- Thorpe, C., Kanade, T. & Shafer, S. A. (1987), Vision and Navigation for the Carnegie-Mellon Navlab, *in* 'Proc. Image Understand Workshop', pp. 143–152.

- Torres-Jara, E. & Edsinger, A. (1999), 'Face Recogniton'.
- Viola, P. & Jones, M. J. (2001), 'Robust Real-time Object Detection'. Cambridge Research Laboratory, Technical Report Series, CRL 2001/01.
- Walter, W. (1950), 'An Imitation of Life', Scientific American 182(5), 42–45.
- Walter, W. (1951), 'A Machine That Learns', Scientific American 185(5), 60–63.
- Walter, W. (1953), The Living Brain, Duckworth, London, United Kingdom.
- Whitney, D. & Nevins, J. (1978), 'What is the remote center compliance and what can it do?'.
- Whitney, D. E. (1977), 'Force Feedback and Control of Manipulator Fine Motions', Transacions of the ASME, Journal of Dynamic Systems, Measurement and Control pp. 91–97.
- Whitney, D. E. (1982), 'Quasi-Static Assembly of Compliantly Supported Rigid Parts', Journal of Dynamics Systems, Measurement, and Control 104, 65–77.
- Whitney, D. E. (1987), 'Historical Perspective and State of the Art in Robot Force Control', The International Journal of Robotics Research 6(1), 3–14.
- Wilkinson, S. (2000), Benefits and Challenges of Microbial Fuel Cells in Food Powered Robot Applications, in 'Journal of Autonomous Robots, paper 99-026'.
- Zeng, G. & Hemami, A. (1997), 'Overview of Force Control', *Robotica* 15(5), 473–482.