Autonomous Motion And Obstacle Detection For The Robosuitcase 015

Diploma Thesis by
Hans Jäckle
Zürich, Schweiz
s97-912-976
jaeckle@gmail.com

at the
Artificial Intelligence Laboratory
Prof. Dr. Rolf Pfeifer
Department of Informatics
University of Zurich

Supervisors:
Andreas Fischer
Marc Ziegler

February, the 9th, 2007

Abstract

In this assignment an obstacle avoidance algorithm for the robotic suitcase *Robosuitcase 015* will be developed. After an introduction into the underlying theories and an analysis of the status-quo of the project, appropriate sensors will be searched and built in. Based on this an obstacle avoidance algorithm will be developed and implemented. With a series of tests the functionality of the obstacle avoidance algorithm will be verified and the used strategies will be optimized.

Zusammenfassung

In dieser Arbeit wird ein Algorythmus zur Erkennung und Vermeidung von Hindernissen für den robotischen Reisekoffer *Robosuitcase 015* entwickelt. Nach einer Einführung in die theoretischen Grundlagen und einer Analyse des Status-quo des Projektes werden passende Sensoren gesucht und eingebaut. Darauf aufbauend wird ein Hinderniserkennungsalgorythmus entwickelt und programmiert. Anhand von Tests wird die Funktionsweise des Hindernisserkennungsalgorythmus verifiziert und die verwendeten Strategien werden optimiert.

Contents

Chapter 1

Introduction

In today's society mobility is one of the crucial components. People need to travel frequently, either for holiday reasons or for business, always accompanied by their suitcases and bags, carrying the cloths and all the other stuff needed for the trip. Arriving at the train station or the airport the hauling begins. The heavy burden has to be carried along long floors and through wide halls.

Imagine a middle aged man, lets call him Hans[1], on a business trip to England. Hans needs to stay in England for a whole month, to manage a project which is important for his career in the firm he works for. Therefore he has a big suitcase with his cloths, a small one for his business materialand a bag around his shoulder with the laptop.

While Hans is rushing through the airport, thinking about the problems that could occur in the project, his mobile phone rings. It could be his chief, with the latest news on the project and some additional information, so he has to answer it. He has to stop, put down his luggage and pick up the phone. It is his wife, telling him not to forget to take pictures to show her the stations of the trip. As they already talked about that in the morning and Hans was already tight on time, he got bothered. The speaker calls out his name and that the plane is waiting for him, so he starts to run to get the flight. At the check in desk he notices that he only has his business suitcase and the laptop bag around his shoulder.

He had to turn and go back where he answered the phone call from his wife to get the big suitcase back. Fortunately it was still there, so he only missed the flight. Still it was a very uncomfortable situation.

For people like Hans it would be more comfortable if they would not have to care about their luggage, it should follow them autonomously.

Figure 1.1 illustrates this scenario in some undefined future. With this inspiration the idea of the robosuitcase 015 was born, which is the subject of this thesis.

[1]Resemblances with people involved in this assignment are pure coincidence.

Figure 1.1: Danifuturo and the flying suitcases.

1.1 The Robosuitcase 015

The robosuitcase 015[2] aims to develop a robotic suitcase that will be able to follow its owner through an unknown environment.

Sure the suitcase will not fly behind its owner like it does in figure 1.1 but todays technology allows a suitcase to be driven by an electric motor and follow on the floor. Therefore the suitcase must be able to locate its owner in such a place and determine the distance and direction to him or her. Above that on busy places, where many people hurry through, autonomous behavior implies an efficient obstacle avoidance system.

For feasibility reasons the price of the technology used should be in a relation to the suitcases price, because nobody would buy the robosuitcase 015 for a significantly higher price. As the project is also meant as a feasibility study, the financial aspect is important and must be considered throughout the thesis.

The task of following the owner can be divided in different subtasks, that can be implemented on their own and connected through a suitable strategy. One of the lowest layers then would be the obstacle avoidance, that must act independently. That means the direction given by the owner walking in advance must be corrected when an obstacle comes into the way of the suitcase. This obstacle avoidance layer is the aim of this thesis. To simplify the task and to make it suitable for a diploma thesis it has been defined that the suitcase should be able to follow a floor. Thereof a research question was extracted which will be described in the following.

[2]This project is also present on the internet under `http://www.015.ch`, where recent information is available.

1.2 Research Question

Out of the preceding project description the following research question has been extracted:

> How can a simple method allowing the robosuitcase 015 to follow a floor, regarding sensors, software and morphology be developed?

In the following subsection this question will be further divided and wrote out to a full assignment.

1.3 Thesis Objectives

The robosuitcase 015 should be able to follow a floor while avoiding the obstacles on its way. Therefore the sensors and the control of the steering will be developed, based on a given micro-controller program.

The assignment can be divided in the following sub-parts:

1. **Evaluation of sensors:**

   For the detection and avoidance of obstacles in the way sensors are necessary. Therefore suitable sensors for this project must be found including the following tasks:

   - What kind of sensors exist and how are they used (similar applications)?
   - Which sensors are suitable according to their characteristics (size, weight, mechanical and environmental liability, speed, cost)?

   Within this selection it is important that the sensors are small and light-weighted, suitable to coordinate driving at walking speed[3] and financially feasible for the comprehensive development of the robosuitcase 015.

2. **Positioning of the sensors:**

   The positioning of sensors has an influence on their functionality. Therefore an evaluation of different sensor-alignments is necessary. This also includes a clarification of how many sensors are necessary by the following questions:

   - How do sensors have to be aligned?
   - What is the minimum number of sensors to reach the aim of following a floor?

[3] Walking speed is not exactly defined. It is somewhere between $3\frac{km}{h}$ (slowly) and $7\frac{km}{h}$ (military march speed) or $0.83\frac{m}{s}$ and $1.94\frac{m}{s}$.

This addresses also the principles of cheap design.

3. **Programming the steering-software:**

   The coordination of sensor-feedback and motor-management must be implemented on the micro-controller in a suitable way. Therefore strategies for an appropriate obstacle avoidance behavior have to be developed. These strategies then must be tested against each-other to find out the most effective. The selected strategy then has to be tested and optimized. This includes the following questions:

   - Which strategies exist?
   - How can these strategies be compared?
   - Which strategies are suitable for the project?

1.4 Applied Theories

This assignment basically is part of the artificial intelligence research domain which is influenced by multiple research fields. The term *artificial intelligence* was first used in 1956 at the Dartmouth conference [20] where scientists of different domains came together to discuss this new discipline. The aim was to combine the research-results and findings from various research streams to get emergent insights through the consideration of multiple domains.

Earlier approaches in artificial intelligence and robotics tried to program intelligence in a pure causal manner. Therefore conditions have to be isolated and formulated in a statement. For each of these conditions or a combination of several the corresponding reaction must be defined in a response. This procedure can be seen as an if-then-instruction. Although this approach can provide good results in a known environment, it results in a very static system that is intolerant to unknown conditions.

To increase the fault-tolerance and stability other approaches have been developed. The given assignment as stated in the prior subsection indicates that there is the need for taking into consideration the following theoretical approaches, which will be explained in chapter 2:

- Autonomous agents.
- Cheap-design.
- Obstacle avoidance.

1.5 Research Approach

In order to solve the research question, first a literature research on the state of the art in theorie and practice has to be done. This will lead to insights on the demands to the sensors that have to be chosen. Then a suitable sensor has to be selected, built into the suitcase and attached to the microcontroller.

When this is done a software has to be implemented, that allows the coupling of the sensors and the motors. This software must provide an interface for an obstacle avoidance strategy that has to be developed and adopted to the robosuitcase 015. With tests these strategy will be verified. The following section will show how these issues will be considered within the scope of this assignment.

1.6 Structure of the Thesis

The text starts with an analysis of the theoretical background, relevant to the assignment in chapter 2. This chapter provides the necessary theoretical knowledge that has an impact on the robosuitcase 015, followed by a short overview over existing autonomous robots.

Chapter 3 gives a detailed description of the robosuitcase 015, its components and the software used as the assignment started. This is also meant to be a documentation of the robot, as it has only been partially described up to now. Therefore also the technical data will be determined.

A crucial component of an autonomous agent is its sensory system. Therefore in chapter 4 an appropriate sensor type will be selected and attached to the robosuitcase 015.

Building on the technical data and the selected sensors the software must be developed. The resulting Null15AIControl program and its main components will be described in chapter 5. This software is the core for the following research on the morphological possibilities and obstacle avoidance algorithms which will be implemented and described in chapter 6.

The algorithms implemented have to be tested in a real environment. Therefore test scenarios have to be determined and performed with the robosuitcase 015. These tests and the subsequent optimizations are part of chapter 7.

Finally the results achieved will be summarized in chapter 8, the conclusion, accompanied by an outlook on further development and research with the robosuitcase 015.

Chapter 2

Theoretical Background

The vast amount of theories influencing the artificial intelligence research makes it necessary to focus on the subjects that are relevant for this assignment. Therefore in this section some selected terms will be described. This section also aims to give an overview on some of the relevant literature within the chosen subjects. After the theoretical overview, some examples for existing products on the market that are somehow related to the robosuitcase 015 will be illustrated. The presented products make use of the theories mentioned in this thesis and attention will be given to the differences between those products and the robosuitcase 015 will be outlined.

2.1 Autonomous Agents

To design an autonomous agent Pfeifer [26] proposes a framework of design-principles that have to be taken into account. This framework has a strong impact on the assignment and therefore is explained in the following enumeration taken from Pfeifer [26]:303[1]:

1. **The three constituents principle.**

   The design of autonomous agents always involves three crucial issues that have to be considered, the so called constituents of design.

   (a) Definition of an ecological niche.

   (b) Definition of desired behaviors.

   (c) Design of the agent.

   Often development does not have the possibility to run through these principles one point after the other. One or more premises can be given and the rest has to be adopted to fit the constituents, but also the given design issues.

[1] The emphasized texts are all taken from the same overview in the book.

In the case of the robosuitcase 015 the ecological niche and the agent design are predefined. The ecological niche is defined through the use of the robot as a suitcase. This means its environment will be an unknown building with flat floors and a lot of static and dynamic obstacles. The design is given through the suitcase, that fits as the robot's body. Therefore, the behavioral aspects seem to be the most demanding in this thesis. These have been described in the assignment and through the research question as avoiding obstacles when driving along a floor.

Once the constituents of design are defined, a developer must think about the morphology, the architecture and the mechanism of the robot as described in the following principles ([26]).

2. **The complete-agent principle.**

   A complete agent must be autonomous, self-sufficient, embodied and situated.

   This means that an agent must be able to function without human intervention (autonomous), sustain itself over an extended period of time (self-sufficient), interact with the real world (embodied) and have its own sensory system to perceive the world (situated) [26].

   Still complete artificial agents that fulfill all these criteria to the same extent do not yet exist [26]. Therefore designing an agent should focus on the particular characteristics, that help to fulfill a desired task or behavior (principle 1).

   Looking at the robosuitcase 015, autonomy is limited to the extent, that the robot must be able to autonomously detect and avoid obstacles, further behavioral strategies are not required at the moment. Self-sufficient behavior is a point that can be completely left aside. The robots hardware is not even designed for self-sufficiency yet. As a robot that must drive through a real floor, the interaction with the real world is crucial and the robot has to be equipped with sensors. Embodiment and situatedness therefore are elementary components of the robosuitcase 015.

3. **The principle of parallel, loosely coupled processes.**

   Intelligence is emergent from an agent-environment interaction based on a large number of parallel, loosely coupled processes.

   This principle is based on Brooks' subsumption architecture [7] and the sense-think-act cycle [26]. When sensing, information is gathered and represented in a central model of the environment, whereof plans for actions can be thought about. The most appropriate action can then be chosen.

This point is highly depending on the software design that will be chosen throughout the assignment. If possible the software architecture should somehow implement a subsumptioin architecture or at least give the possibility to be integrated in a subsumption like architecture when further development takes place.

4. **The principle of sensory-motor coordination.**

   All intelligent behavior is to be conceived as a sensory-motor coordination that serves to structure the sensory input.

   In other words, behavior can not only be viewed from a information-processing perspective, but must be directly guided by the sensory input [26]. This is strongly connected to the embodiment as through the robots interaction with its environment sensory information is generated not just given. Through the morphology of a robot the way the environment is viewed is defined.

   To generate appropriate and useful data the positioning of the sensors has to be considered. This underlines the importance of morphological aspects when developing the robosuitcase 015.

5. **The principle of cheap designs.**

   Designs must be parsimonious and exploit the physics and constraints of the ecological niche.

   This principle states, that an agent can be much more effective by taking into account the desired behavior and the ecological niche. Often no central control is required and thus communication between processes can be reduced to local information exchange [26, 16]. By taking into account the ecological niche assumptions about the environment and its factors can be made and therefore a further reduction in complexity can be achieved. This results in a parsimonious design matched to the desired task and behavior.

   The physical design of the robosuitcase 015 is defined, therefore parsimonious designs must address the software. To what extend these designs can profit depends on the physical characteristics of the suitcase and must be exploited during development.

6. **The redundancy principle.**

   Sensory systems must be designed based on different sensory channels with potential overlap.

   Looking at the sensory channels of an agent this means, that there should be an information overlap within the acquired environmental information.

As the robosuitcase 015 has no sensory channels yet, first a single channel has to be found. This channel can later be extended with additional sensory channels to comply with the redundancy principle.

7. **The principle of ecological balance.**

   The "complexity" of the agents has to match the complexity of the task environment. In particular, given a certain task environment, there has to be a match among the complexity of sensor, motor, system and neural substrate.

   More complex tasks or behaviors therefore need a more complex sensory-motor apparatus.

   Motor, steering and microcontroller boards are defined for the robosuitcase 015. These are simple components, that have to be taken into account when choosing the sensory system.

8. **The value principle.**

   The agent has to be equipped with a value system and with mechanisms for self-supervised learning employing principles of self-organization.

   For the robosuitcase 015 it is more important to fulfill the task than to be a complete autonomous agent. This makes the value principle less important for the assignment.

These eight points define a complete autonomous agent. As the interest of most robots is focused more on the task or behavior side and autonomy is only required as long as it is relied to that task, these points seldom are fulfilled all at once. As Pfeifer states in his book *Understanding Intelligence* [26] complete autonomous agents only exist in nature up to now. In this work the interest is not to build a comprehensive autonomous agent, but rather autonomous behavioral aspects. Therefore some of the principles in the list will be left out during development and only the points used to fulfill the given task will further be used, as described with each of the principles. In chapter 6 this will be further discussed. As the core of the assignment is to avoid obstacles, in the following section an overview on the subject of obstacle avoidance will be provided.

2.2 Obstacle Avoidance

Obstacle avoidance is an essential behavioral aspect of a mobile robot. To achieve this behavior there exists a variety of approaches. While some try to build obstacle avoidance as a result of complex information processing (as the *Stanford Cart* [23]

or *Shakey*[2], as popular examples), others try to make it an emergent behavior of the design constituents mentioned above (like the *Turtles* of W. Grey[3] or the behavioral robotics approach described by Brooks [7] and in Pfeifer's book [26]). The obstacle avoidance behavior is mostly influenced by the three-constituents principle, the robot's morphology, its sensory motor coordination, while it must respect the principle of ecological balance and take advantage of the principle of cheap design [26]. The other principles are certainly important for complete agents but for this assignment they can be omitted.

For the obstacle avoidance the representation of the environment in a world model is crucial, defined through the sensory motor coordination. As the world model is the result of the sensory inputs the principle of ecological balance determines the amount of behavioral aspects that can emerge from the overall behavior *obstacle avoidance*. If the robot is asked to move in a room without getting stuck in local minims, a different morphology is required ([9]) than if it is asked to move as fast as possible from A to B ([17]).

This results in basically two different approaches of obstacle avoidance found in the literature. On the one hand there are algorithms that produce a world model including external data ([4, 9, 15, 14]), where comprehensive path planning algorithms can be performed and on the other hand there are algorithms that only require pure local data, what results in a world model that only respects a *robot's point of view* ([17]).

Through today's technology that provides things such as GPS[4] or other means to determine the position within a room, positioning of an agent is very easy. But an agent making use of external data as a crucial component is not robust, as a failure in the connection to these external systems makes it impossible for the robot to act (redundancy principle). Additionally it is a discrepancy with the complete-agent principle (a robot must have its own sensory system).

It is the local algorithms that are of interest looking at the project 015. But these are not so widely spread, as existing robots often are built for maximum task compliance. Some examples of existing robots will be presented in the following.

2.3 Good Practices

This section aims to give an overview on existing robots that show a degree of autonomy. Technology seems nowadays advanced enough to build robots that work

[2]For further information on Shakey and a list of connected publications visit `http://www.ai.sri.com/shakey/` (29.01.2007)

[3]For info about the turtles see `http://www.extremenxt.com/walter.htm` (29.01.2007)

[4]Global Positioning System, a system to determine the position anywhere around the world.

robust for mass-production. Robots slowly start to find their way into households and into everyday live, as the following examples illustrate.

Automatic lawn mower: the Automower™[5], illustrated in figure 2.1, is designed to automatically mow the lawn. To make this possible, the lawn has to be surrounded with a loop line. Within this loop line the robot mows the lawn by randomly searching its path. For obstacle detection it is equipped with touch sensitive sensors. Touching an obstacle makes is back up a little and continue its task in another direction. When its battery runs low after 40 to 60 min of mowing, it automatically searches its charging station to recharge the batteries. To adapt it to the user's live, it can be programmed to work on certain hours during the day only.

Figure 2.1: The Electrolux Automower™, a robot that automatically mows the lawn.

Autonomous vacuum cleaners: While the Automower™ is designed to mow the lawn there is a variety of robots for vacuum cleaning rooms. Figure 2.2 shows the picture of a model built by Kärcher[6], as representative for the family of robotic vacuum cleaners. They all work more or less after the same principle, already seen with the Automower™. As rooms have a natural constraints through their walls, the vacuum cleaners are not relying on a loop line. They randomly drive around in the rooms, cleaning the floor and when the battery is low, or the dust bag is full they locate the charging station, where the dust is emptied and the batteries are recharged. The user can program operational hours for comfortability. These devices are usually equipped with touch sensors to detect obstacles and with infrared sensors to detect stairways and walls. Again they just back up a little when detecting an obstacle and continue their task in another direction. As there are already quite a number of manufacturers that have automatic vacuum cleaners in their product line, the prices for such robots are reaching an acceptable level.

[5]The Automower™ is built by Elektrolux. For further information visit `www.electrolux.com` (21.12.2006).

[6]For further information on Kärcher visit `www.karcher.de` (21.12.2006).

Figure 2.2: The automatic vacuum cleaner built by Kärcher, as a representative for the family of automatic cleaners.

Electric wheelchair: in current research there is an attempt to make electric wheelchairs safer and easier to use ([11, 18]). Therefore the wheelchairs are equipped with sensors to view the environment and an obstacle avoidance layer is built into the control of the chair. In normal operation the wheelchair acts as it did before. But if an obstacle is detected, the control is taken from the user and the wheelchair acts as an autonomous agent, driving the chair automatically around the obstacle. This is especially helpful for people that have not full control over their movements, what results in erratic steering commands through the user. The sensors used for these projects are mostly ultrasonic range sensors. As electric wheelchairs support the mobility of their users, they must be able to work in unknown environments. This makes them use mostly local obstacle avoidance algorithms. The need to adapt to new environments makes them a good example and starting point for the assignment.

Swisslog TransCar AGV system: The automatic guided vehicle (AGV) system is designed as a health care transport system, made by Swisslog[7]. Figure 2.3 shows a TransCar cart. The system does not require wired paths or any other building modifications, as the carts are equipped with laser scanners on the front and the back which serve to avoid obstacles like walls or persons on its way. Additionally it has touch sensitive sensors on the sides which are out of range of the laser scanners. The building structure is stored in a central system control center, which also coordinates the pick-up and drop-off locations. Through input panels at the pick-up locations a user can define a desired destination and send the loaded cart to the desired drop-off location. Localization of the carts is done through a wireless network in the building. If the cart runs out of battery it moves to the charging station. As there

[7]Further information on Swisslog and the TransCar AGV System can be found under `www.swisslog.com/de/hcs-index/hcs-systems/hcs-agv.htm` (21.12.2006).

usually are many of these carts working in one facility the system control center coordinates the recharging order of the carts according to their battery load and the current request.

This solution is very comprehensive and while traveling the robots react to local changes on their path. But as they are made to work mainly in a static environment (with only slight changes, such as persons walking around) the carts rely on a positioning system that is not part of the robot itself. As there is a big amount of expensive sensor technology and a global positioning system that is required for proper operation, the agents in this system are not of big value for the thesis, in because of the feasibility restrictions. Still it is a very interesting system to study.

Figure 2.3: The cart of the Swisslog TransCar automatic guided vehicle system.

While these robots are all very interesting, each of them exhibits significant differences to the project 015. The mowing robot and the automatic vacuum cleaner do only react on obstacles through tangency, which works good for these, as they only move with slow speed. However, the robosuitcase 015 has a rather high weight and is asked to move with high speeds. This implies a sensory system, that allows reactions before an obstacle is hit to provide the possibility for avoidance behavior. Still they realize a maximum of independent behavior out of very cheap designs considering the computational perspective as well as the budget perspective.

The electric wheelchair and the TransCar system have better solutions, as they have means to react on approaching obstacles on some kind of sight. This is made possible by a more complex sensor network with longer reach and better accuracy realized with either ultrasonic or laser sensors. However, these devices show up with an impressive equipment in a sensory as well as in a computational view.

The robosuitcase 015 is situated somewhere in the middle of these applications. A robust obstacle avoidance is desired, which in case of becoming a product for the market, has only a minor influence on the product price compared to a normal suitcase. As this thesis is not the first work done with the robosuitcase 015, there

are also some premises that must be taken into account. The following chapter 3 concentrates on these premises and gives a detailed view on the robotic suitcase as defined when this assignment started.

Chapter 3

The Robosuitcase 015

The examples of the automatic mower and the robotic vacuum cleaner in chapter 2.3 show that people start to accept everyday products that have a certain degree of autonomy. Therefore the idea of building an autonomous suitcase is not that far from reality.

As this assignment is based on the pre-work of different people this basis must be distinguished from the extensions made to fulfill the assignment.

3.1 Hardware Components of the Robosuitcase 015

This section will describe the main components of the 015 suitcase and how they work together.

3.1.1 Actuation

The robot started as a remote controlled car (RC-car) with an in some ways special body. The body is a hard-shell suitcase with an only slightly modified RC-car base built into it. The RC-car is mounted in a way that the motor can power the rear wheels, while the steering-servo is connected to one of the front wheels. Figures 3.1 and 3.2 show the powering and the steering in detail.

3.1.2 Remote Control

The suitcase can be switched from microcontroller control to usual remote control. Therefore a receiver is built into the suitcase which can be activated through a switch. This has been built in for demonstration purposes and is of no further use in this assignment.

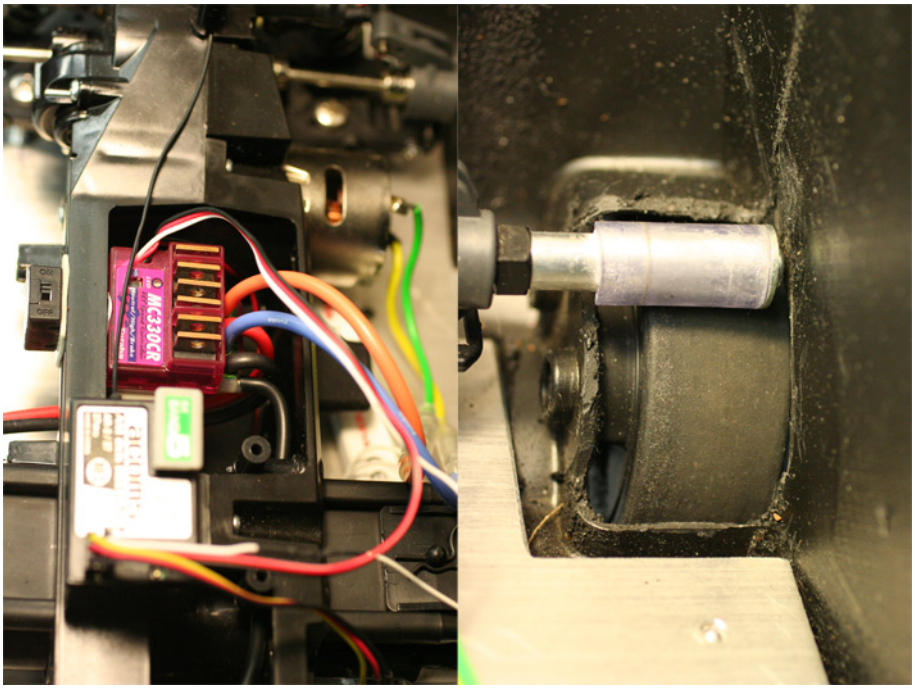

Figure 3.1: Detailed view of the 015 suitcases powering section with the built in RC-car base on the left and the actuation of the wheels on the right.

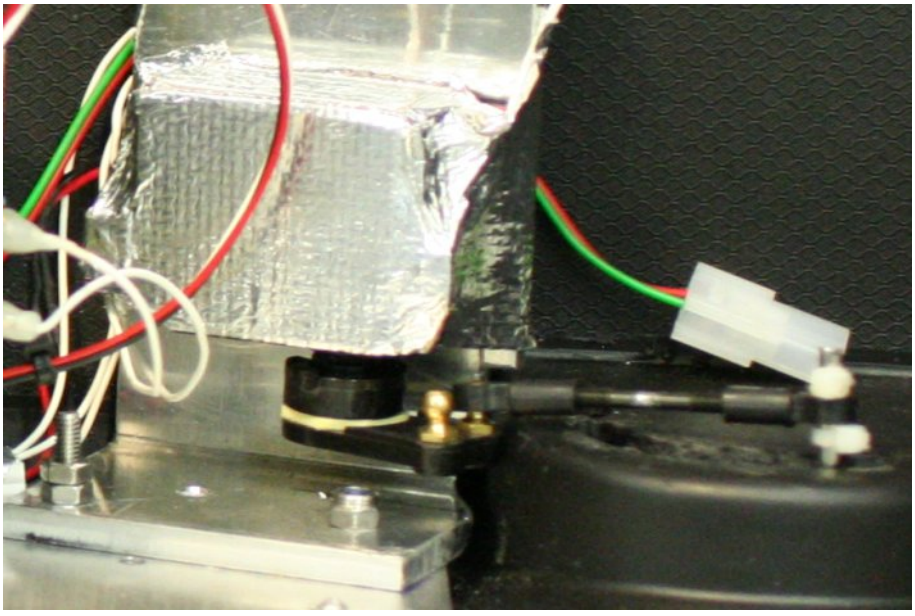

Figure 3.2: Detailed view of the steering section of the 015 suitcase with the servo connected to one of the front wheels.

3.1.3 Microcontroller Boards

The central control is realized with a wireless-lan board (wlan-board). On the wlan-board a PIC18F452 microcontroller provides a variety of in- and outputs. According to the manual [22] this microcontroller has eight analogous inputs and several digital inputs, what makes it a good basis for a robot. The wireless connection board makes it easy to communicate with a personal computer or a laptop. Figure 3.3 shows a schematic view of this layout.

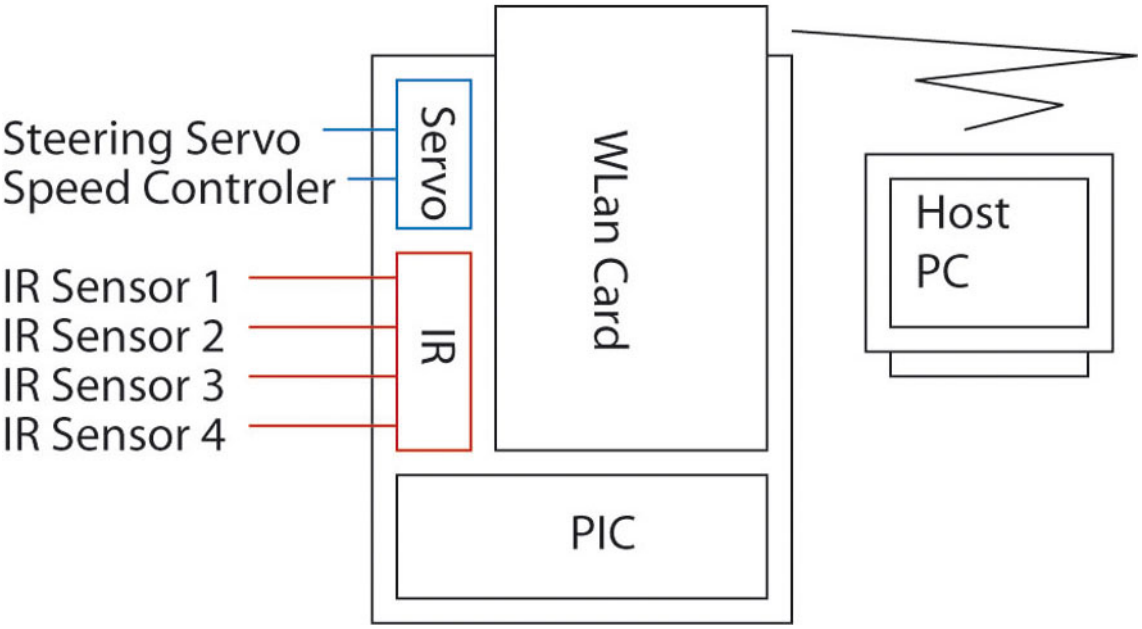

Figure 3.3: Schematic view of the controller-board and the connected components.

The idea behind this architecture is to provide a platform that is easy to modify and where the software can rather be developed on a regular PC than on the board itself. This also makes it possible to use the programing-language and development-tools a user prefers and therefore provides a more independent solution than the direct programming of the PIC microcontroller. In the case of the project 015 the analogous inputs were used for four Sharp GP2D120 sensors [27]. These infrared sensors measure distances from three to 30 centimeters and therefore are good for near-field detection. Additionally a wireless web-cam has been built into the suitcase. This camera is not connected to the controller-board, but has its own wireless connection to the host.

Together this results in a very fast running robot, that can be remote controlled through a computer via the wireless-connection.

The aim of the assignment is a suitcase that avoids obstacles in its way. Therefore it first must be able to detect them in some way. With the given infrared sensors this can not be achieved, because their reach of about 30 centimeters is just not

appropriate and their accuracy is not that good as they are very noisy. To provide a possibility to attach new sensors to the robot more inputs are needed. This was realized through a second development-board with its own PIC microcontroller of the same type on it. To make a distinction between the two boards, the first board with wireless network capabilities will be called wifi-board and the new board which serves as input-extension will be called io-board in the remainder of this thesis.

The microchip PIC development-board [21] gives a good extension for the wifi-board. Equipped with the same controller they communicate over an I^2C-bus[1], which is widely used in digital electronics.

The new io-board has again eight analogue and several digital inputs in addition, what makes it a good extension and a flexible solution for the robot. Figure 3.4 shows the schematic view of the new layout with the io-board attached to the wifi-board.

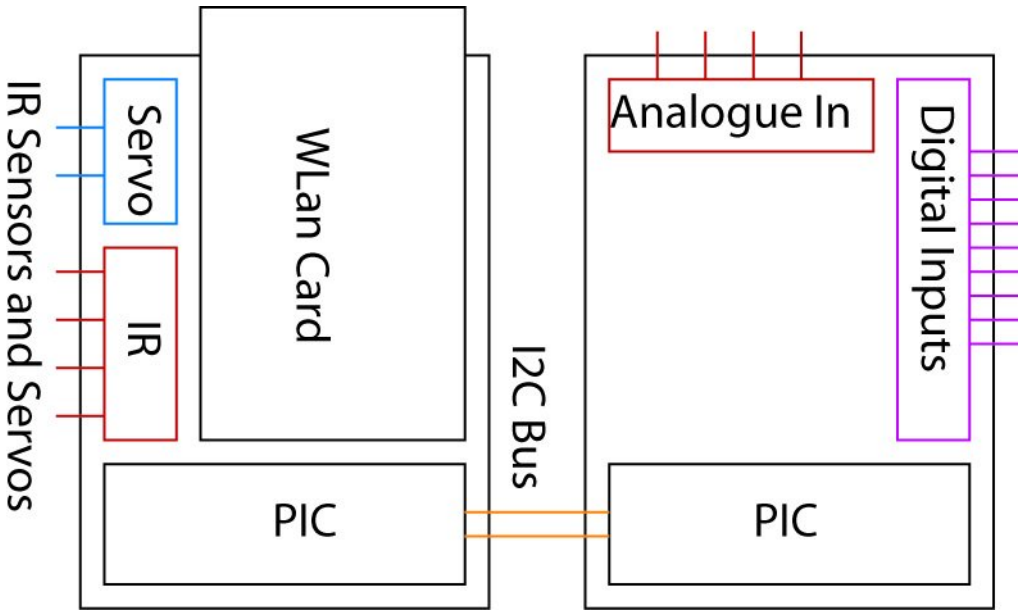

Figure 3.4: Schematic view of the new io-board and the inputs it provides attached to the wifi-board with its components.

To make all the different in- and outputs accessible the boards have been screwed together and fixed to the suitcases. An additional small board equipped with plugs for the sensor attachment has been installed in a way that extensions can easily be made. The new setup can be seen in figure 3.5.

[1] I^2C-bus means Inter IC bus and was developed by Philips. It was designed to provide a communication channel for integrated circuits. See `http://www.elektronik-kompendium.de/public/borchers/i2c/whatis.htm` (01.02.2007) for further information.

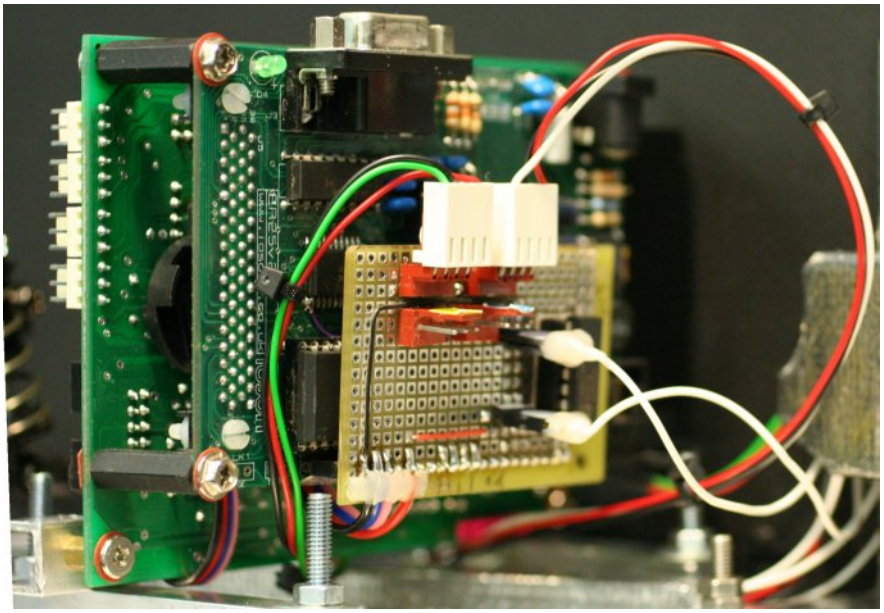

Figure 3.5: Detailed view of the two connected boards. The wlan card is between them and can therefore not be seen.

3.2 Software of the Robosuitcase 015

The hardware of the suitcase has to be supported by a corresponding software to take advantage of the built-in features. The software of the 015 robotic suitcase consists of two parts.

3.2.1 Microcontroller Programms

First there is a C-program on the microcontroller. It delivers a web-interface to retrieve the current sensor-data and provides the possibility to set steering commands, which means speed and direction settings. The largest part of the microcontroller was programmed by Rolf Basler [2, 1] especially for this project. Figure 3.6 shows the web page delivered by the suitcase. Through this site the speed and the direction can be set and a motor stop limit, which sets a limit for the infrared distance sensors, where the motor is stopped automatically.

3.2.2 Java Programs

The second part of the software is the Java interface that works using the robots web-interface. This is where the development takes place. Therefore this program will be explained in more detail. In this section the old program Null15ControlCenter (see appendix C) will be described and the suitability for the assignment will be evaluated.

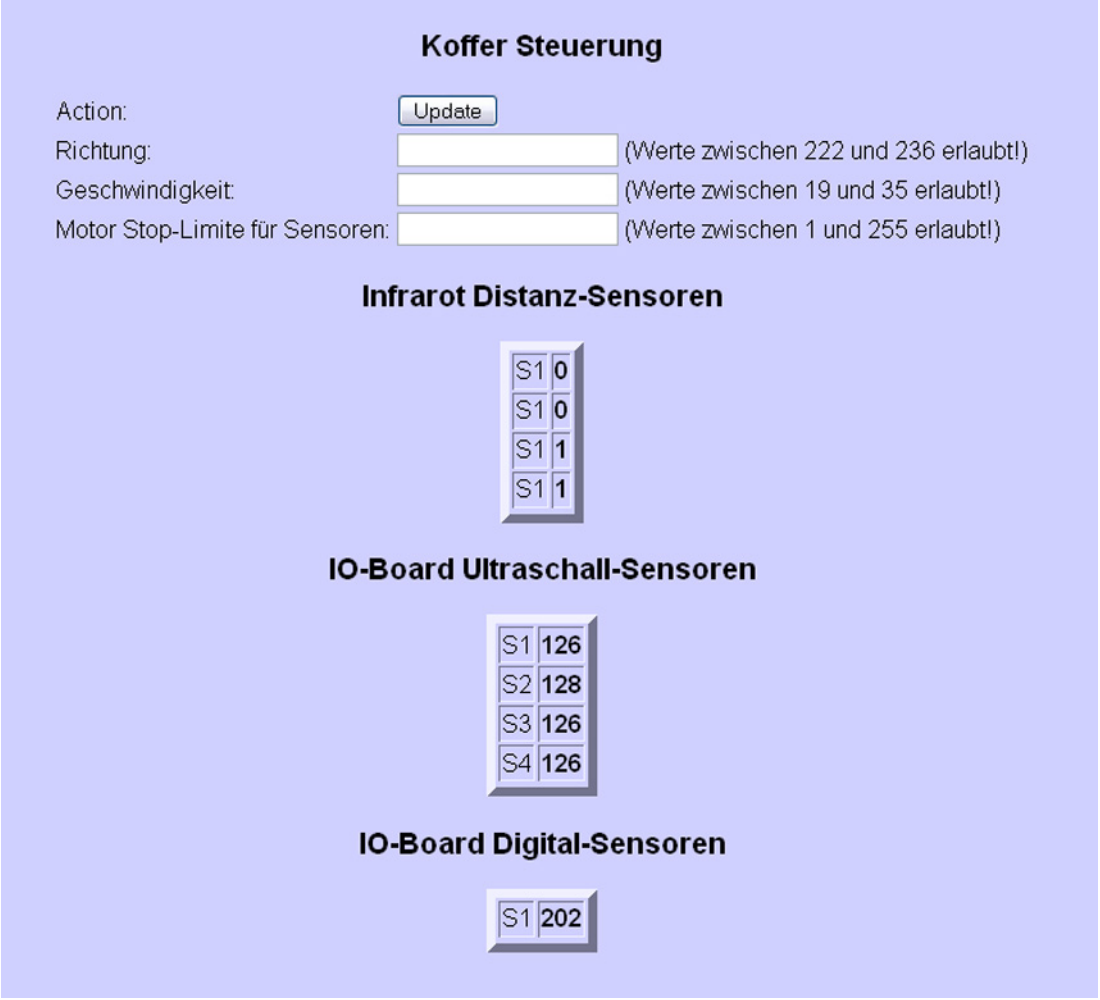

Figure 3.6: The web-page delivered from the suitcase. Through the html-form the controls can be updated.

The Null15ControlCenter program, implemented by Jonas Bösch[2] (see appendix C), has the functionality of a remote control. A graphic user interface displays the pictures from the built-in camera and through the keyboards arrow-keys the suitcase can be controlled like one might be used from computer-games. Figure 3.7 gives an impression of the interface.

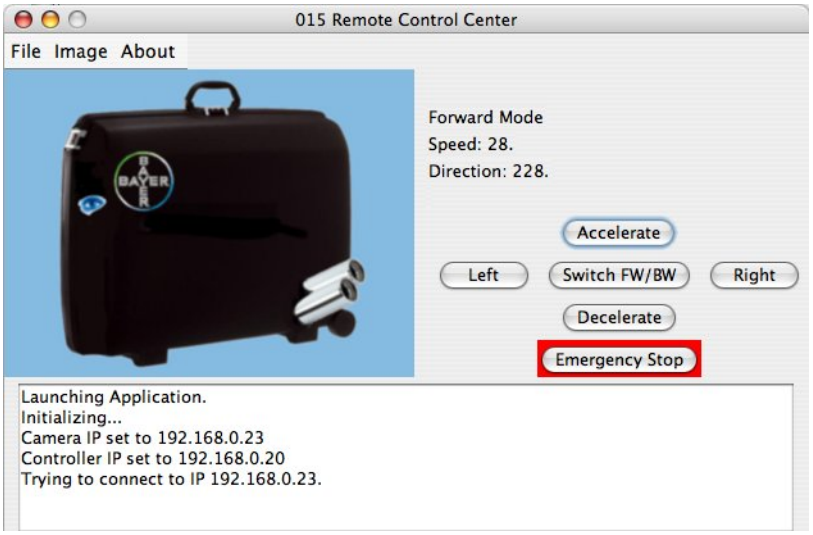

Figure 3.7: The graphical user interface from the Null15ControlCenter.

To control the suitcase the Null15ControlCenter makes use of the web-server on the wifi-board, through a set command within the url. The Null15ControlCenter program is not capable of reading the sensor-data delivered through the web-site, what makes it unusable for autonomous behavior. Additionally the program exhibits significant weaknesses in its design. This made the understanding of the program difficult and further upgrades with new functionality were hard to implement. After some tries it was decided, that the whole software had to be rewritten with a proper architecture and with respect to further development after this assignment.

3.3 Technical Data of the Robosuitcase 015

Although the software is crucial for the success of this assignment, it must respect the mechanical abilities of the 015 robotic suitcase. This is necessary because the behavior is also dependent on the given physics of the robot. As these details were unknown up to now they had to be measured. Table 3.1 gives an overview on the dimensions measured. It is to say that the speed is more or less estimated, by measuring the time by hand of a relatively short distance of fife meters. Although several runs were performed, a reliable speed could not be determined as it has

[2]Jonas Bösch implemented this program upon request of Andreas Fischer under heavy time pressure.

significant differences depending on the surface the suitcase rolls on. The values mentioned in table 3.1 give an impression on the achieved speeds. The other values are accurate.

Steering wheel angle	40° (for each side)
Steering radius	$1m$
Brakedistance	$1.5m$
Wheelbase	$0.5m$
Maximum speed	ca. $3 - 4\frac{m}{s}$

Table 3.1: Summary of the technical data of the 015 robotic suitcase.

The way the suitcase has been constructed also has some inherent disadvantages. Due to the small wheels and the inflexible chassis the robot is very sensitive to uneven surfaces. It happens that one wheel has no contact to the ground and this leads to a discontinuous actuation, especially in combination with the differential gear of the built in car-base. Another disadvantage occurs because only one of the steering wheels is actuated by a servo. Therefore the steering behavior while backing up is not predictable, especially for short distances.

3.4 Summary

To be able to connect appropriate sensors to the robot, new inputs must be available. As the given wifi-board was fully equipped, an extension of inputs was required. This was realized through the newly added io-board, where additional sensors can be connected as described in section 3.1.

These sensors then must be integrated into an obstacle avoidance algorithm, which is part of the control-program. As this was not foreseen in the given Null15-ControlCenter a new application must be developed.

The algorithms depend on the technical and physical data of the suitcase. These had to be measured, what was part of section 3.3. This data also influences the selection of appropriate sensors, what will be described in detail in the next chapter 4.

The 015 hardware has to be connected to the theoretical background described in chapter 2. In the 015 project the ecological niche is defined. For the assignment the niche has been further narrowed to the floor of the artificial intelligence laboratory.

The task of following this floor implies the desired behavior of avoiding obstacles situated in there.

The physical design of the robot is already given for the utmost part. Possible changes encompass the sensors, their positioning and the way of interaction. The

selection of a specific sensor that will be used for the suitcase is part of the following chapter.

Chapter 4

Sensor Technology

Sensors exist in a big variety of types and designs. Each of them with its very own specialties and use. This chapter describes the important steps on the search for a sensor that fits the needs of the 015 project.

4.1 Premises through the Robosuitcase 015

The project 015 has some preliminaries and restrictions that have to be met by the sensors. In this section these will be outlined.

As the robot is asked to travel with an accurate speed the sensors must have a certain reach to allow a reaction on obstacles. This range is depending on the following criteria defined through the robots hardware and the sensors themselves:

1. Stopping distance of the robot d_s.
2. Turn radius of the robot r.
3. Travel speed of the robot v.
4. Time between two sensor measurements t.

The robot must be able to stop or turn in front of an obstacle without touching it. Therefore the reach of the sensors must be bigger than these criteria. Additionally the robot needs some time to react when detecting an obstacle, what further increases the requirements of the sensor's range. The reaction time is depending on the minimal time a sensor requires between two measurements. Therefore an appropriate sensor must have a minimal reach of $max(d_s, r) + (v \cdot t)$. A bigger distance is desirable, but not necessary.

When considering the suitcase's technical data from chapter 3.3 and a slow sensor with only two measurements per second, this leads to the following minimal range: $max(1.5m, 1m) + (3\frac{m}{s} \cdot 0.5s) = 3m$.

This equation is very much simplified, but it is sufficient as a mean to start the selection of a specific sensor. As the range needed is strongly depending on the measurement time it is desirable to have a sensor that is capable of more than two measurement per second. This would increase the reliability and accuracy of the whole robot.

4.2 Sensor Selection

The sensors analyzed were of four types: infrared, radar, laser, ultrasonic. As the robosuitcase 015 was already equipped with infrared sensors (see [27]) it turned out quickly that this kind of sensor type is not usable. Besides the very limited reach of maximally $1m$ for some types (what is far below the desired reach) and only $0.3m$ for the installed sensors, they produce a lot of noise and do not deliver reliable measurements.

The second kind of sensor type analyzed was the radar type. They provide fast and reliable measurements and have a very high reach. These sensors are often used in mobile obstacle avoidance[1]. Unfortunately only industry standard radar sensors were found with prices high above the budget.

Laser sensors are very precise. While the other sensors measure the distance of the closest object within an angle, laser sensors measure one single spot. They can be found as so called scanners, that measure a range of points in a certain angle, typically they measure 1° steps within a 120° range an a radius of $6m$. As with the radar sensors only high priced industry standard models could be found. Besides the price the high amount of data would make it impossible to run the obstacle detection mechanism on the PIC (or only with reduced resolution). This would not comply with the principle of ecological balance (principle 7, chapter 2.1).

The only type of sensor that fulfilled the financial aspect was the ultrasonic type. Here industrial sensors are very expensive, too, but as the technology is simple there are also some models on the market, that are built for home use and even some with regard to robotics. The sensor finally found was the SRF05 ultrasonic range finder, which will be described in the following.

4.3 The SRF05 Ultrasonic Range Finder

According to the data sheet the SRF05 from Devantech has a reach of about four meters and takes about $50ms$ for one measuring cycle, what results in up to 20 cycles in a second. The sensor is equipped with an interface that allows an easy

[1] See [8, 12, 24, 30] for examples.

attachment to any microcontroller and needs a five volts powering, which is common among microcontrollers. Above that it has a reasonably low price of about 22 Euros per piece[2]. Figure 4.1 illustrates the sensor board as it is delivered.

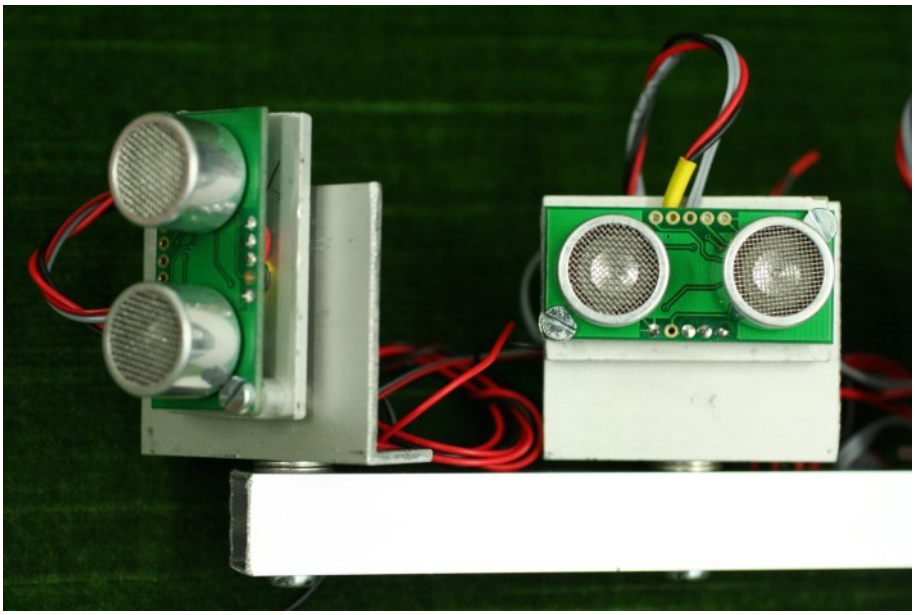

Figure 4.1: The SRF05 ultrasonic range finder module.

The technical data of the SRF05 easily fulfill the criteria defined in advance and therefore seemed to be a good choice (see [28] for technical data). To measure distances of objects in its range, the SRF05 uses the SONAR technology . This technology will be described in the following section.

4.3.1 SONAR Technology

Sonar stands for *sound navigation and ranging* [6]. This technology makes use of the fact, that waves are reflected from surfaces. By measuring the time lapse between sending a ping, what means a short sequence of sound, and the returning echo, the distance from the sender to the reflecting surface can be calculated. Figure 4.2 illustrates the sonar effect.

To determine the distance to an object only the sound-propagation velocity of the waves in the transmitting medium must be known. In this assignment the medium is air. This provides a little problem as the sound-propagation velocity in air is depending on environmental factors such as temperature or atmospheric-pressure. Therefore a DIN[3]-standard exists which says that sound-propagation velocity at sea level is $c_s = 341.2\frac{m}{s}$. Over that it decreases linearly to $c_s = 295.8\frac{m}{s}$ at an altitude of eleven kilometers above sea level [6].

[2]The sensor was acquired through roboter-teile.de. For further information visit `http://www.roboter-teile.de/Shop/` (02.10.2006).

[3]DIN stands for *Deutsches Institut für Normierung*

With a time-difference t in seconds and the sound-propagation velocity c_s in meters per second the distance in meters d can easily be calculated as the result of the following equation: $d = \frac{t*c_s}{2}$.

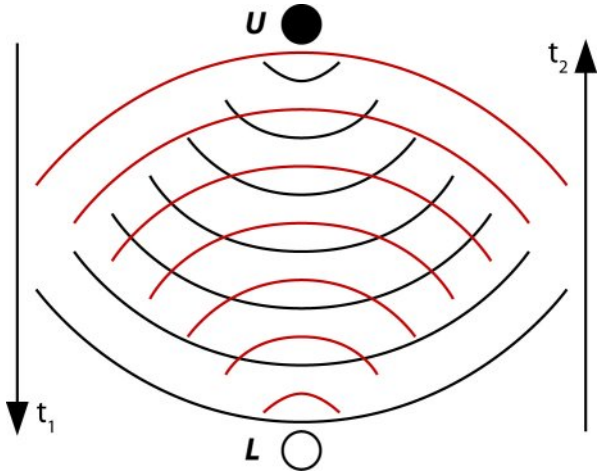

Figure 4.2: Illustration of the sonar principle. U emits a ping that is reflected by L. By measuring the time $t = t_1 + t_2$ the signal takes to travel from U to L and back the distance can be calculated.

If the operation hight of the robosuitcase 015 is limited from sea level to $1000m$ above sea level, the maximum failure can be calculated. The datasheet for the SRF05 sensor [28] recommends usage within $4m$. This means that the distance the waves have to travel is $8m$. Taking the speed at sea level S_0 and the speed at $1000m$ above sea level S_{1000}, the δt in seconds and the δd in meters can be calculated as:

$$\begin{aligned} S_a &= 341.2\tfrac{m}{s} \\ S_b &= 337.1\tfrac{m}{s} \\ \delta S &= S_a - S_b &&= 45.4\frac{m}{s} \\ \delta t &= \tfrac{8m}{S_b} - \tfrac{8m}{S_a} &&= 0.00028s \; or \; 280\mu s \\ \delta d &= \delta t \cdot \delta S &&= 0.013m \; or \; 13mm \end{aligned}$$

δd is the maximum failure cased through differences in air pressure (what actually influences the sound-propagation velocity). This little failure of $13mm$ does not need to be addressed in the software.

4.3.2 Attaching the SRF05 to the Microcontroller Board

The sensors attached to the microcontroller board and a software must be implemented to make use of them. Once initiated the SRF05 sensor does all the measuring on its own an returns the result of its measurement as a digital pulse [28]. By dividing the pulses length in μs through 58 the measured distance in cm can be determined.

To get the time of the pulse a loop on the PIC checks the signal of the SRF05 sensor in time intervals. The datasheet recommends intervals of $100\mu s$ between the checks, what results in a distance value of 232 for a measured distance of $4m$. The whole process of initiating the sensor, waiting for response and measuring the pulse takes about $50ms$ [28] and has to be repeated for each sensor.

Therefore different algorithms have been programmed for the microcontroller. The software tools used for this purpose and a short explentaion on how to install them on Mac OSX can be found in the appendix B. For the program and source code see appendix C. Figure 4.3 shows the first algorithm that was implemented on the microcontroller. The sensors are all invoked in parallel. This approach did not perform appropriate, as there were a lot of false measurements due to cross talking. Crosstalking occurs when a sensor reads the return signal of another sensor as illustrated in figure 4.4.

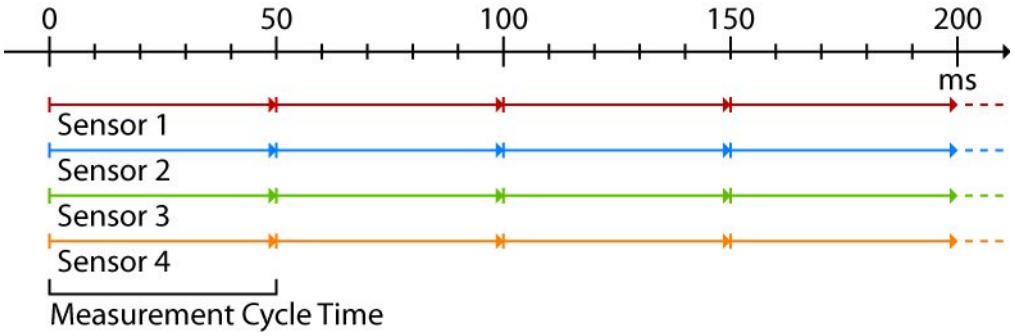

Figure 4.3: Parallel invocation is the most simple way to read from the sensors.

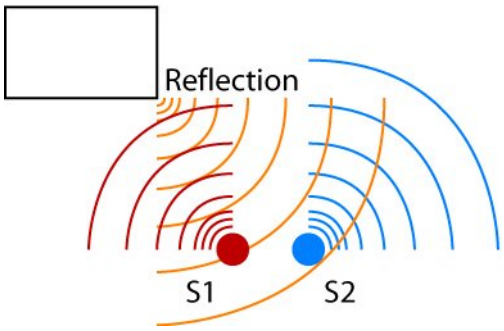

Figure 4.4: Crosstalk produces false measurements. As shown in the illustration a signal reflection sent by sensor $S1$ can also be received by sensor $S2$, if $S2$ is listening.

Therefore the algorithm illustrated in figure 4.5 was implemented. This algorithm performed good, but it takes $200ms$ to get the data of all four sensors, what reduces the amount of measurements per second to five. This is still enough, but it is far below the possibilities of the sensor modules. On the other hand it is safe through the serial invocation of the sensors.

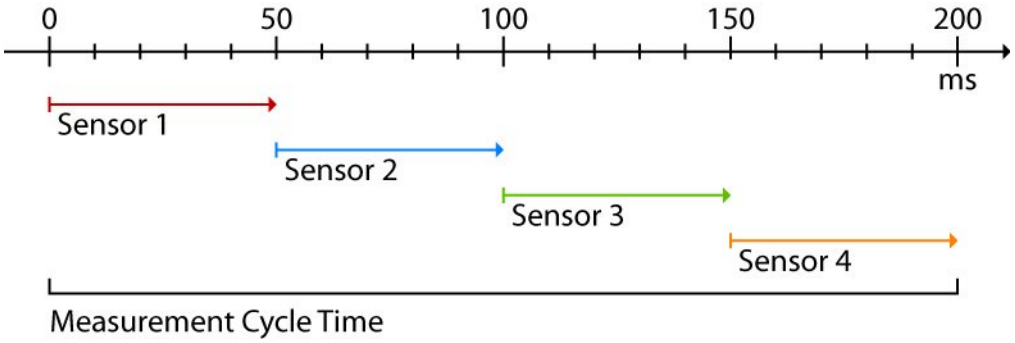

Figure 4.5: Serial invocation is a very simple algorithm, but takes a lot of time.

To reduce the time of measurement cycle an algorithm was designed that introduced a variable time offset between the sensors. But this algorithm suffered serious timing problems and therefore an implementation was not possible. Probably it could be runnable on a PIC with increased frequency[4]

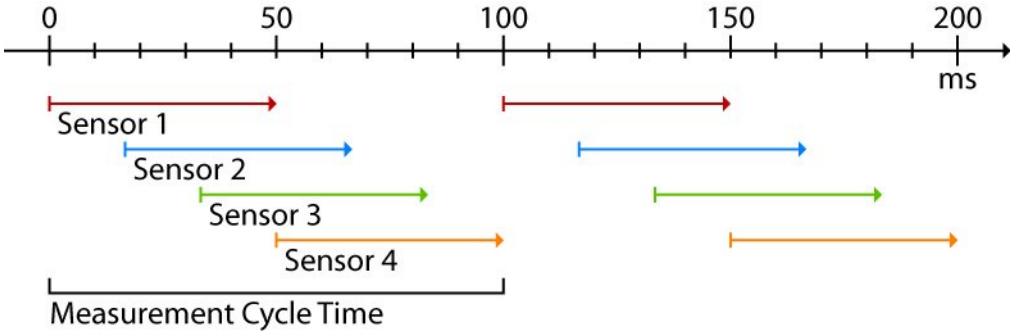

Figure 4.6: Invocation with an offset between the sensors is the most complicated and computationally most demanding way of reading from the sensors. But is combines good accuracy with reasonable repetition speed.

The timing problems could only be omitted by combining parallel and serial invocation of the sensors. This lead to two combined methods that are faster than the serial approach and less error prone than the parallel one. These are shown in figures 4.7 and 4.8.

Having the sensor modules attached to the microcontroller board, the reliability of the measurements of each sensor were tested. This process is described in the following section.

4.3.3 Sensor Calibration and Verification

To get an impression on what the sensors measure, tests were performed. For the tests the serial algorithm was taken, as it guaranteed that only one sensor fired at once. The first test performed was a distance test to compare the measurements of

[4]The PIC used can be stepped at frequencies up to 40MHz, but as the development board used was designed for an older model, it is stepped at 4MHz only.

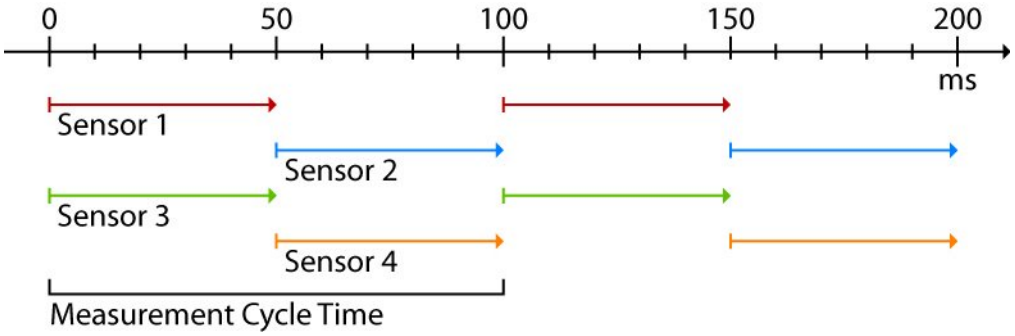

Figure 4.7: Combination of serial and parallel approach, that suffers similar problems as the parallel approach.

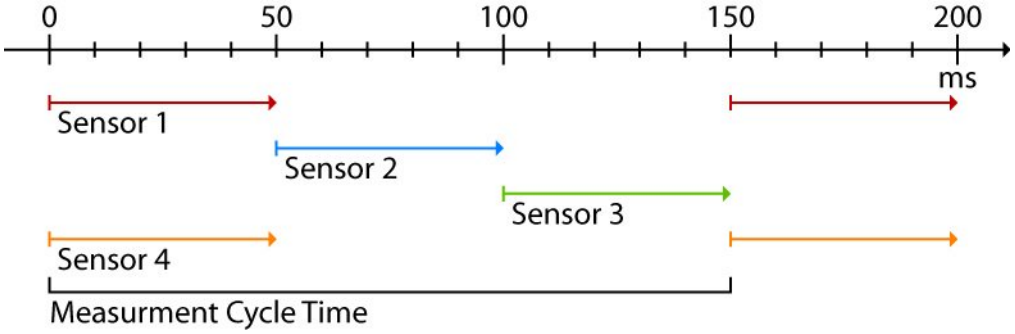

Figure 4.8: Combination of serial and parallel approach, without the major problems of the parallel and faster measurement cycle.

each sensor. It turned out that these sensors must have been calibrated from the manufacturer, as they all measured the same for the same distance. Table 4.1 gives an overview on the distances measured and the values representing this distance returned by the web interface. This value comes from a loop on the io-board that measures the pulse length returned by the sensor. The pulse length represents time the echo needed to return (see section 4.3.1). For a better overview only one sensor is displayed, but as they all did return the same distance value for the same distances this is appropriate.

Real Distance	0,13m	0.51m	1m	2m	3m	4m
Representing Value	7	28	53	108	162	217

Table 4.1: This table shows distances in relation to the measured values from the SRF05 ultrasonic sensor.

Through the characteristics of ultrasonic waves there are some additional measurements that must be taken into account. Especially measurements under angles seem inevitable to determine the physical limitations under which the sensor is able to deliver reliable data. Through the anatomy of the SRF05 with two transceivers[5]

[5]A transceiver is a sender and a receiver at once.

mounted side by side, these measurements must be taken with different sensor alignments, to examine possible influences. The scenarios illustrated in figure 4.9 also pay respect to the upcoming work as the results can be useful in developing an obstacle avoidance system as they deliver additional information about the environment.

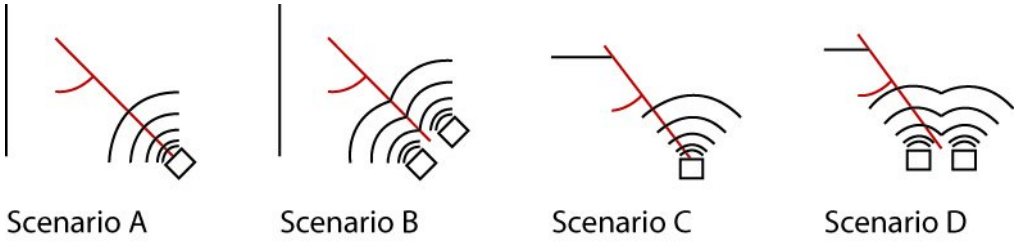

Figure 4.9: To get information about the circumstances under which the sensors work properly the illustrated test scenarios were analyzed. The information of interest is highlighted by the red lines.

The scenarios in figure 4.9 have been tested on different surface materials, listed in the following enumeration:

1. Structured wall
2. Linoleum ground
3. Carpet ground

The setup of the test can be seen in figure 4.10 (for scenario A and B) and figure 4.11 (for scenario C and D). In table 4.2 the results of these measurements are summarized. Again the four sensors showed very similar results, therefore the table has been summarized and only one sensor is displayed.

Scenario	A	B	C	D
Wall	45°	50°	45°	50°
Linoleum	50°	55°	–	–
Carpet	40°	45°	–	–

Table 4.2: This table illustrates the maximum angles the sensors could make measurements according to the scenarios and the surface. Scenarios C and D were not reproducible with carpet and linoleum, therefore there is no value.

The tests showed that before the sensor has and accurate measurement there is an angle of about five degrees where the returned value jumps between no measurement and the distance of the obstacle. The angles displayed in the table are the angles where the measurement stops jumping an constantly stays on the value representing the distance. The angles are rounded to the next 5 degrees, as with the available means no more precise measurements could be performed. Still these tests give

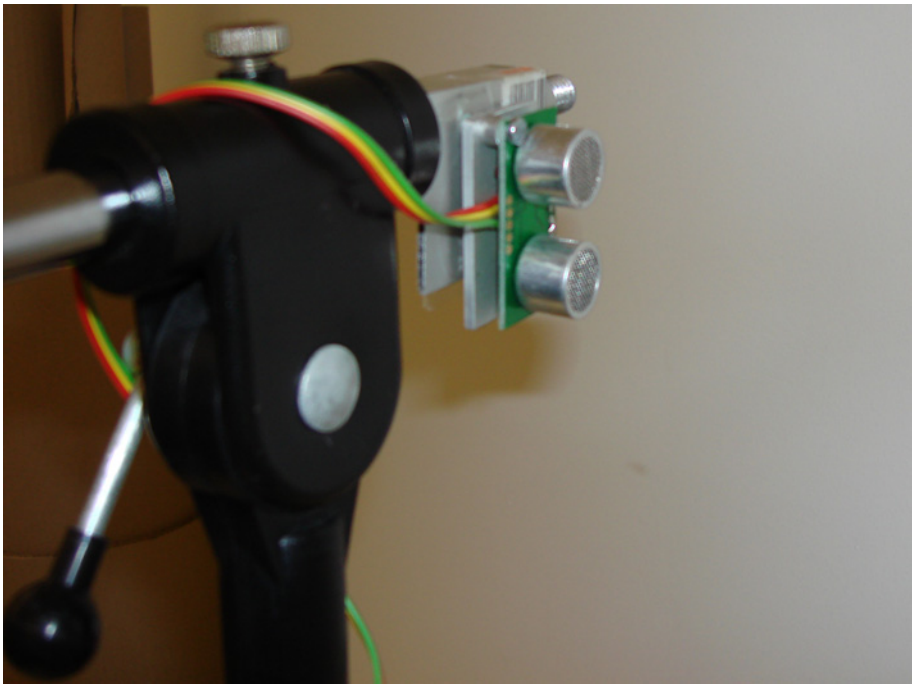

Figure 4.10: The setup to test the scenarios A and B. By turning the sensor toward the wall in the background, the angle can be determined.

Figure 4.11: The setup to test scenarios C and D. The sensor is mounted on a stand and can be turned toward a pile of plastic cases (yellow).

a good overview on the capabilities of the SRF05 sensor modules and helped to develop the Null15AIControl program described in the following chapter.

Chapter 5

The Null15AIControl Program

The microcontroler boards described in chapter 3.1 communicate with a host computer over a wireless-lan connection. On the host computer the steering application of the robosuitcase 015 is running. This control program is called Null15AIControlCenter. This chapter aims to give an overview on the functionality of the application, illustrate the architecture and to serve as a documentation for a possible user.

Therefore first the changes from the old Null15ControlCenter application will be outlined followed by a section concentrating on architectural details, where also the functionality is explained.

5.1 Control Program Changes

The Null15ControlCenter program was only able to send control-data to the suitcase. It had no functionality that made it able to retrieve the sensor-data provided through the web-site. As this program was not properly encapsulated the redesign resulted in a complete reprogramming. The new program Null15AIControl has a similarly looking graphical user interface (GUI), but is completely redesigned in the background. Figure 5.1 shows the new user interface. To give an impression on the changes in the programming a comparison of the class-hierarchy is shown in figure 5.2.

The functionality of the program first had to be extended in a way that it could read the sensor-data from the web-site of the suitcase. Therefore the web-site containing the actual sensor-data has to be downloaded and processed and the data must be extracted and represented in a way that the program can work with it.

An other big lack of the Null15ControlCenter was the absence of an interface to build in the autonomous behavior. The new Null15AIControl provides this and the control structures can be easily modified in one single class. This provides the basis for the development of autonomous behavior for the robosuitcase 015.

Figure 5.1: The new GUI of the Null15AIControl, that provides access to the robots functionality for the user.

Figure 5.3 illustrates the architecture of the Null15AIControl program as class hierarchy, which provides a good overview on the new architecture. In the subsequent section 5.2 the main components of the Null15AIControl program will be explained.

5.2 Main Components of the Null15AIControl

The Null15AIControl is built on three main classes. The first one is the visible component for user interaction, the graphical user interface (GUI). This class allows the user to interact with the second main class, the suitcase. The Suitcase class owns the third main class AICore, that is responsible for autonomous behavior. These classes and their interaction will be described in the following. Additionally some design related issues will be commented at the end of this section.

5.2.1 The Suitcase Class

The Suitcase class provides an interface for all the functionality the suitcase has. Basically the Suitcase class has two operation modes, a remote mode, where the user interface acts as a remote control and a AI mode where the suitcase acts autonomously.

Starting the AI mode invokes the AICore class, which can be equipped with a sensor motor coupling from the abstract class SensorMotorCoupling. The SensorMotorCoupling class implements the actual sensor motor coupling algorithm. In this class the steering commands are calculated from the current sensor data and then returned to the Suitcase class which sends the command to the suitcase via wireless-lan.

This architecture allows to change the sensor motor coupling without the need to restart the software and to easily add further functionality, which is important

Figure 5.2: This comparison illustrates the changes in the new Null15AIControl (B) compared to the old Null15ControlCenter (A).

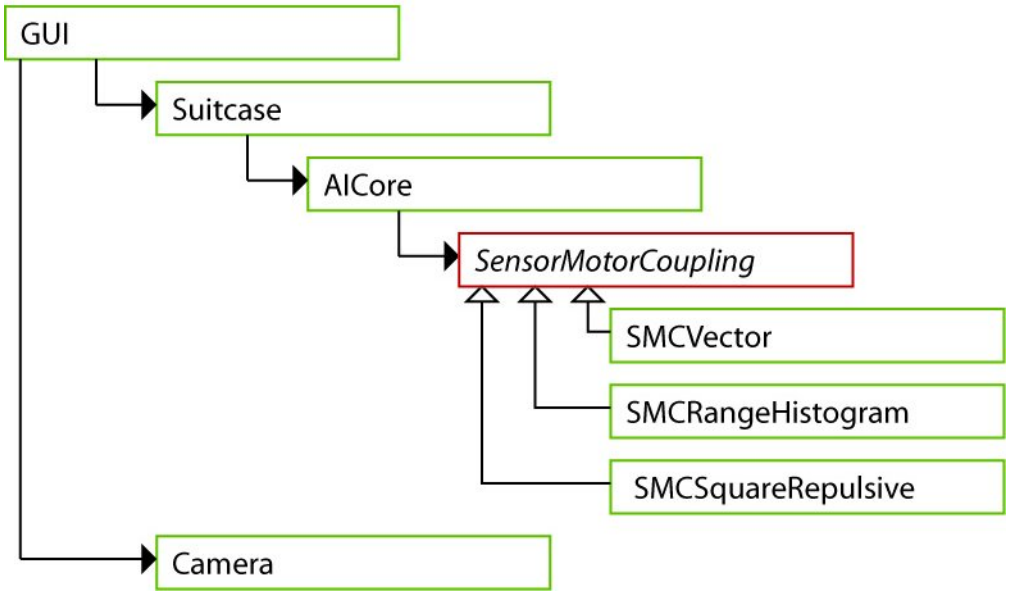

Figure 5.3: Class hierarchy of the Null15AIControl Program. Only the important classes are illustrated. Abstract classes are modeled with their names in italics.

with respect to the ongoing development after this assignment. Additionally the strong encapsulation helps to understand the program code for someone who has to continue developing. For the code and code documentation the reader is referred to appendix C.

5.2.2 The AICore Class

When autonomous motion is required, the Suitcase class initializes a AICore object. This class provides some data structures for further calculations, like a list with the commands returned to the suitcase or a list with a history of the sensor data sent to the AICore class. For noisy sensors it also provides configurable lists with geometric and arithmetic means, to smoothen the sensor data. The following formulas show the calculation of these means in a mathematical sense. The first line shows the arithmetic mean $Mean_A$ and the second line the geometric mean $Mean_G$ with n being the number of values for which the mean should be calculated and x_i the current value.

$$Mean_A = \frac{\sum_{i=0}^{n} x_i}{n}$$

$$Mean_G = \sqrt[n]{\prod_{i=0}^{n} x_i}$$

These lists can then be used for calculations in the SensorMotorCoupling class, in which the obstacle avoidance algorithm is implemented. The SensorMotorCou-

pling class has been implemented as abstract class, which must be implemented by the algorithms. Therefore each algorithm has its own class where it can easily be modified or even exchanged. If multiple algorithms are implemented it is up to the user to choose which one to use in the AICore class.

5.2.3 The GUI Class

The GUI gives the user access to the functions of the software developed. Figure 5.1 shows a screenshot of the current version[1], which will be explained in the following.

After starting the program the GUI appears. The GUI is divided in two sections. On the left the robot's built in webcam has its video output. If the camera is not used a placeholder image is displayed. More interesting is the right section where the controls for user interaction are placed. To get started the Suitcase class has to be invoked. This is done with the *Koffer Start* button. After that a Suitcase object will be created and set to remote mode (assuming the suitcase hardware is turned on). The suitcase can now be controlled via the steering buttons in a remote control manner. The *Stopp* button launches an emergency procedure which slows the robot down till it stops and sets the steering back to middle. To start the autonomous behavior implemented in the AICore class the *KI/RC* button must be pressed. The robot will then search its way on its own. The autonomous mode is stopped by either pressing the *KI/RC* button again or by pressing the *Stopp* button. If the robots comes too close to an obstacle an UnavoidableObjectException is thrown and it will then first perform an emergency stop and then switch back to remote mode.

As the camera is implemented through its own class Camera, it has to be started separately if is used. This is done with the *Kamera Start* button. As the camera is not one of the main components the special functions can only be accessed through the camera menu. Functions to take snapshots and video sequences are provided.

To access the property settings the menu item *Steuerung* has to be chosen. There the current settings can be saved or custom settings can be loaded. An editor to change properties is not yet included, but by saving the current settings and then using any text editor to change the values the same result can be achieved.

5.2.4 Other Design Issues

The source code is documented according to the Javadoc[2] specification. This is especially helpful in combination with the eclipse IDE[3] as it provides on the fly

[1] A more flexible version of the GUI, where components can be added and removed on the fly is currently in development.

[2] For further information on Javadoc see `http://java.sun.com/j2se/javadoc/` (25.11.2006).

[3] IDE means Integrated Development Environment. For information on eclipse see appendix B.

access to Javadoc comments. Additionally the whole source code documentation can be exported as a HTML-site to be viewed outside a text editor what provides a more comfortable view.

For the surveillance of the software, logging has been implemented using the apache log4j framework[4]. Log4j makes use of different log levels what allows to easily adapt the amount of logged information according to the needs, without changes in the code by simply editing a configuration file.

As algorithms always depend on some constants, which can be changed to make it fit to the robot and the circumstances, properties[5] have been introduced. If a configuration has been found that works well, these settings can be saved and reloaded again. As the Properties class already provides methods to load and write files according to the XML[6] standard the reusability of the data is enhanced. Properties are especially helpful while optimizing an algorithm or with respect to different morphologies.

[4]For further information on the log4j framework see `www.apache.org` (25.11.2006).

[5]Information: `http://java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html` (25.11.2006).

[6]For further information on XML see `http://www.w3.org/XML/` (25.11.2006).

Chapter 6

Morphology and Obstacle Avoidance

The importance of morphology and the close connection to the desired task of avoiding obstacles can be illustrated by the following citation:

> The morphology of sensory systems has a number of important implications. In many cases, when the morphology of the sensory systems is suited for the particular task environment, more efficient solutions can be found.
>
> [25]:1

This principle was already mentioned in chapter 2 as the principle of cheap design. To achieve an appropriate result the morphology and the obstacle avoidance have to be taken into account simultaneously.

Therefore in this chapter, first the morphological possibilities and influence factors will be elaborated and then obstacle avoidance algorithms are researched that fit to these surrounding conditions.

6.1 Morphology

The morphology addresses the subject of embodiment seen in chapter 2. In the 015 project some morphological constraints were given in advance, as it was described in chapter 3.3. Therefore changes in the morphology and the embodiment address the positioning and the adjustment of the sensors or the *three constituents principle* (principle 1) from chapter 2.1.

The positioning and adjustment of the sensors defines the perception of the robot and therefore the area in which obstacles are detected, what has a significant influence on the behavior. As the morphology will also influence the algorithm it is useful to think about it in advance. Still these subjects are connected very closely and therefore it makes sense to discuss them in the same chapter.

The body of the suitcase has certain drawbacks as described in chapter 3.3. One of them is the inconvenient behavior on short backwards motion. As one of the front wheels is loosely mounted, it is not predictable how the suitcase behaves when moving backwards for a short distance. Therefore it has been determined that no backwards movements will be allowed. As the aim of the assignment is to provide a dynamic and fast movement through obstacles rather than finding a way out of difficult situations this restriction makes sense and does not affect the thesis' motivation. This increases the angle under which obstacles can be detected, decreases the amount of sensors used in total and it decreases the computational effort for the obstacle avoidance system, what results in a higher *ecological balance principle* (principle 7) and a cheaper complexity of the design. This addresses the *principle of ecological balance* as well as the *cheap design principle* (principles 5, 7).

For development and testing the sensors have been attached on a support fixed on the suitcase's front. This provides the possibility to easily change angles and positions of each sensor and with that the morphological aspects. Additionally it can be removed without any visible traces on the suitcase itself. This is important, because the sensors will be built into the suitcase and fixed there, when an appropriate morphology and algorithm has been found. Figure 6.1 shows a photograph of the support mounted on the front. A schematic top-view is provided by figure 6.2.

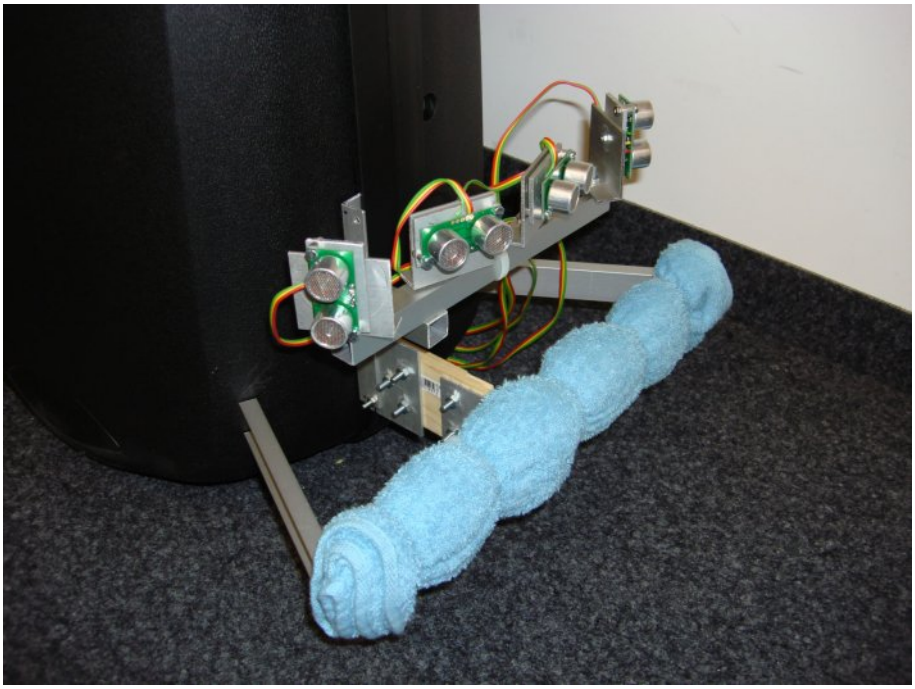

Figure 6.1: Photograph of the support attached to the suitcase to mount and adjust the sensors and the bumper to protect them.

The stand with the sensors provides the possibility to alter the angles of each sensor and the height of all sensors together. Therefore the variable parameters in

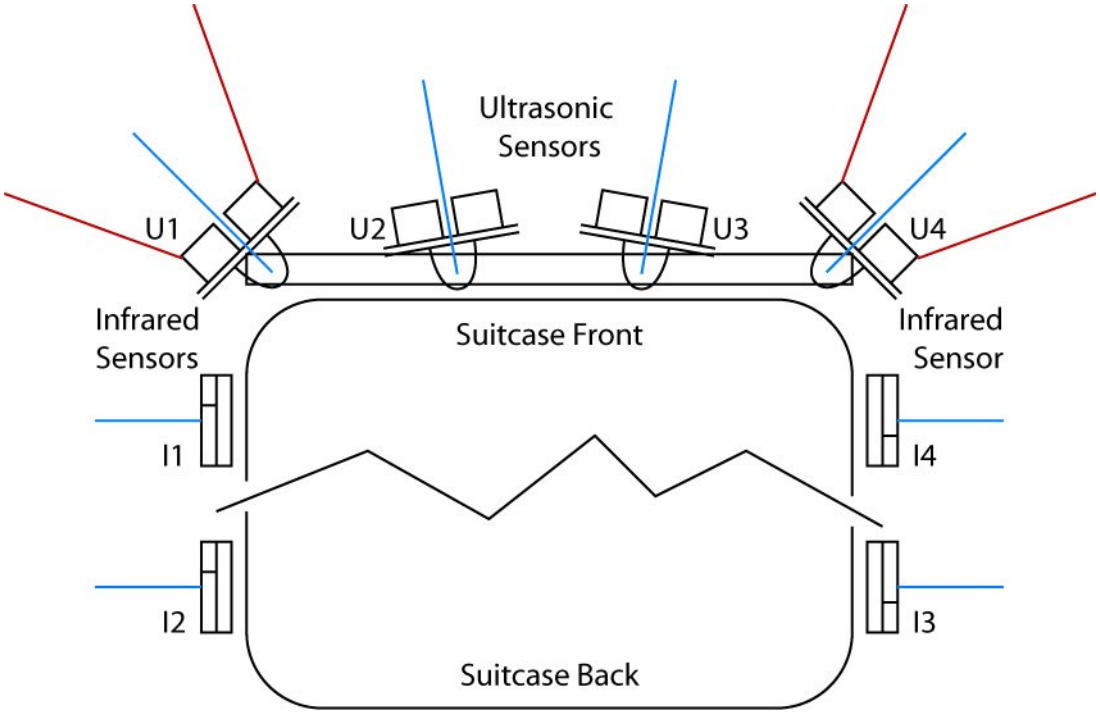

Figure 6.2: Illustration of the attached sensors. The morphology can be changed through different angles of the sensors highlighted by the blue lines.

a morphological point of view are limited to the alignment of the sensors. This has to be taken into account by the obstacle avoidance algorithm that has to be implemented and tested. In the following section two obstacle algorithms are illustrated that respect the morphological restrictions and were then implemented.

6.2 Obstacle Avoidance

From the suitcase, usable information about its state is available only through the distance sensors. Control commands like the direction and speed settings are not usable as an information source, because they are significantly influenced by the structure of the current surface of a floor. Small bumps and grooves reduce speed and change direction.

For the obstacle avoidance algorithm, or the *sensory-motor coordination principle* (principle 4), this implies that there is not enough information to build a reliable virtual map of the environment. To build a virtual map not only the actual relative location of an obstacle, but also information about the translation or location of a robot in a room is needed. This means, a map is based on a (in a way) static reference system where all the acquired information is summed up to a environmental model[1].

The lack of such an environmental model must be compensated through a reac-

[1]As can be seen in various algorithms, like [3, 5, 17, 19, 31].

tive system that results in very direct sensor motor coupling. The spacial information through the distance sensors can be used to build a vector based model of the current situation.

By using the situations the robot must be able to find its way along a floor, dynamically avoiding obstacles as a part of its motion directive. The algorithm has to take the problems mentioned into account.

The information available and the need for a dynamic obstacle avoidance are best reflected by algorithms that make use of some kind of histogram [3, 5, 17, 19, 31]. These methods all map the raw data retrieved from the sensors to vectors that represent the location of an obstacle. Thereof repulsive[2] and attractive forces[3] are calculated or a probability grid[4] is filled. Figure 6.3 illustrates a series of histograms like they are used for the robosuitcase 015.

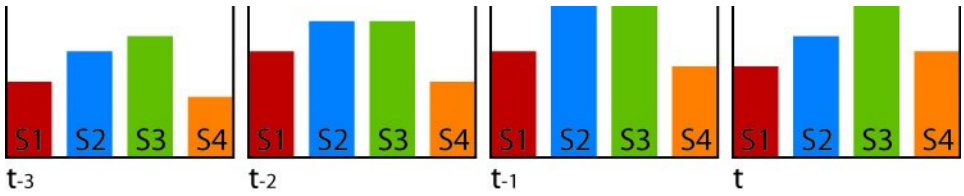

Figure 6.3: Series of histograms for the robosuitcase 015. Each histogram has 4 values, one for each sensor and represents the sensor state at the time t.

All these algorithms are based on some sort of vectors, what makes the calculation of steering commands an easy task based mainly on addition and multiplication. This is important, as the algorithm should later be programmed on the microcontroller which has only limited computational resources and therefore needs an easy to calculate algorithm.

An analysis of the algorithms showed, that the range histogram mentioned in [17] is the most promising. In the following the basics for vector based obstacle avoidance algorithms will be explained followed by an outline of the range histogram. Both were implemented in the Null15AIControl program and then tested.

6.2.1 Vector based Sensor Motor Coupling

To get into the subject of vector based algorithms a very simple algorithm has been implemented.

The idea behind this algorithm is to implement a very simple obstacle avoidance method for the first run, based on vector algebra. A vector usually is defined

[2]Forces that push the robot away.

[3]Forces that attract the robot.

[4]For a probability grid, space is divided in sections. Each section then receives a probability for an obstacle to be there.

through cartesian coordinates, two numbers representing the translation on the x-axis respectively on the y-axis of a coordinate system. An other method is to define it through its length and its direction (defined through its angle), the so called polar coordinates [29].

The ultrasonic sensors deliver the distance measured as a value between zero and 255 (see chapter 4). This value can be treated as the norm of a vector (*norm*) or in other words its length. The direction of the vector is given by the angle the sensor is mounted on the suitcase (α). These values give the polar coordinates of a vector, which can then be used to calculate a cartesian vector according to the following formula.

A very simple approach for an obstacle algorithm can be achieved by simply adding the the four vectors. As the addition of vectors is an addition of their directions weighted with their length, the direction of the resulting vector could then represent the steering angle and the resulting norm the speed. The following formula shows how the steering vector $v_{\vec{steer}}$ can be calculated:

$$\begin{aligned} v_{\vec{steer}} = \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} (x_1 + x_2 + x_3 + x_4) \\ (y_1 + y_2 + y_3 + y_4) \end{pmatrix} \\ speed &= \sqrt{x^2 \cdot y^2} \cdot a \\ direction &= \arcsin\left(\frac{x}{\sqrt{x^2 \cdot y^2}}\right) \cdot b \end{aligned}$$

In this formula a and b are constants to accommodate the steering and speed values to the values needed in the software. Figure 6.4 illustrates this algorithm.

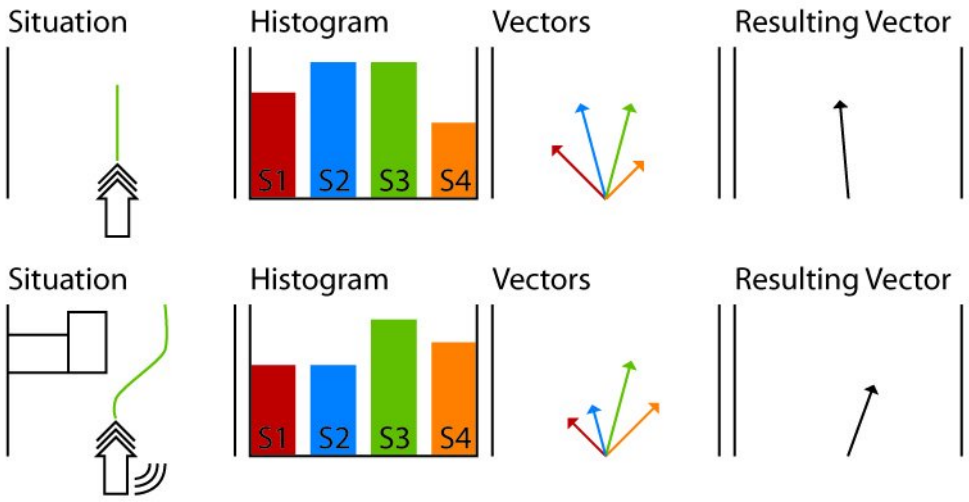

Figure 6.4: The elements of vector based algorithms for obstacle avoidance.

The major part of this calculation is addition and multiplication which is computationally very cheap. Replacing the sine and cosine functions through a lookup

table could further reduce complexity and cost. With these simplifications there is only a square root function left, that is of higher complexity.

This algorithm is implemented in the SMCVector class. The code and the code documentation can be found in the source folder, see appendix C.

Based on these relatively simple calculations a variety of algorithms exist, that take advantage of vectors as a means to determine a steering command. In the following section one of them, the range histogram algorithm, will be shortly presented, as it has been implemented too.

6.2.2 Range Histogram

The range histogram algorithm is based on the same data as the algorithm described before. Instead of calculating the vectors toward an obstacle it calculates a repulsive force from that obstacle [17]. Combined with an attractive force that is determined to be the vector of the robot's current motion this results in a new steering command. Originally the algorithm calculated the forces for each point that had an influence on the behavior of the robot. In the case of the 015 robot this means all the four wheels. For testing this formula has been simplified. The calculation for the repulsive force used in the project is based on the following formula:

$$\vec{F_{repulsive}} = \sum_{k=0}^{n} \frac{c_{repulsive}}{d_k^4} \left[\frac{d_k \cos \alpha_k}{d_k} \vec{x} - \frac{d_k \sin \alpha_k}{d_k} \vec{y} \right]$$

where $c_{repulsive}$ is a constant repulsive factor (which is not closer described by the authors), d_k is the distance measured by the k-th sensor with its angle α_k and $\vec{x}$ and $\vec{y}$ are the norm vectors. This calculation is made for every sensor.

The attractive force, as proposed by [17], implies that the speed and thereof the translation of the robot is known, however it is unknown in the case of the 015 project. To still be able to determine an attractive force the speed had to be estimated. Therefore two approaches were implemented. The first approach took the difference of two consecutive distance measurements and the second took the speed setting of the current steering command multiplied with a constant.

As one can imagine the first approach to determine the speed did not result in usable data. When the robot travels along a wall, or in completely free area no changes to the distance measured happen and therefore no speed could be determined.

The second approach, using the speed setting from the steering command was usable after some optimization. Still it has to be taken into account, that this can only be an approximation of the real speed, as the robot has significant differences in speed depending on the surface it rolls on. The attractive force for the determination of the steering command is then found through the following calculation (again in

a simplified version):

$$\vec{F_{attractive}} = \frac{c_{attractive}}{d_{ti}} \left[\frac{x_t - x_i}{d_{ti}} \vec{x} - \frac{y_t - y_i}{d_{ti}} \vec{y} \right]$$

where $k_{attractive}$ is an attraction constant (no further explanation by the authors), (x_t, y_t) describe the starting point of the movement (normally this would be $(0, 0)$), (x_i, y_i) is the point of the target position and d_{ti} is the distance from the starting point t to the target point i. Figure 6.5 illustrates the process of the range histogram algorithm.

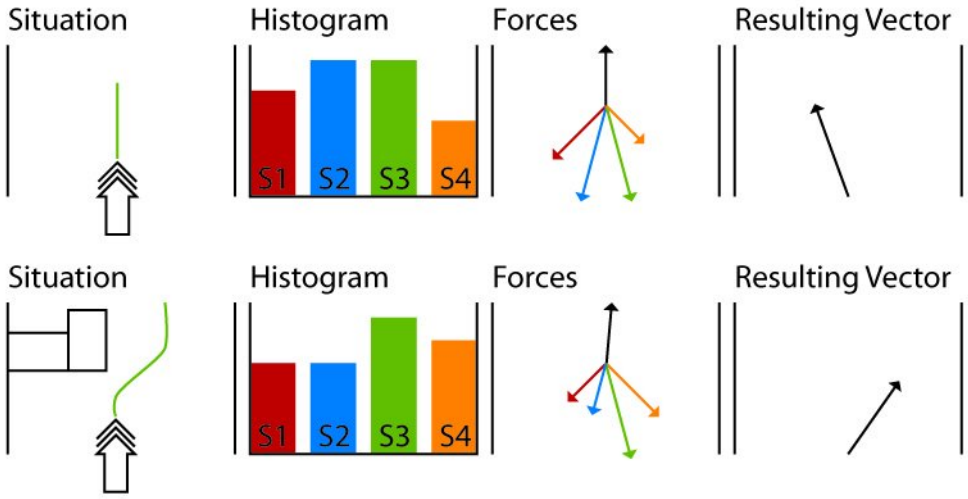

Figure 6.5: The range histogram algorithm combines repulsive forces with attractive forces to determine the resulting vector.

Chapter 7

Tests and Evolution

This chapter concentrates on the tests that have been performed with the 015 robotic suitcase and the evolution it has gone through. Therefore in the first section the test cases used to evaluate the robot's capabilities are described, followed by a description of the testing procedure and the steps taken to the final configuration of the robot. This final configuration will then be summarized with the results of the test runs.

7.1 Test Scenarios and Setup

To evaluate the suitcases capabilities in a comparable way test cases have to be defined. Taking into account the task of following an office floor there are basically three constraints that can be used in combination to define different test scenarios.

1. Walls
2. Obstacles
3. Passages

As the robot is built symmetrically along its length axis the amount of cases can be reduced, considering that an obstacle on the right is similar to the same obstacle on the left. This results in eight test cases as illustrated through the scenarios A to H in figure 7.1, which will be described in the following. More complex scenarios are then a combination of the tested scenarios and can therefore be omitted.

Scenario A – Following floor: the robot should follow a floor without touching the sidewalls. This scenario addresses one of the essential tasks within the thesis. Figure 7.2 shows the floor the suitcase is asked to follow.

Scenario B – Approaching floor end: the suitcase is supposed to stop when it comes to the end of a floor.

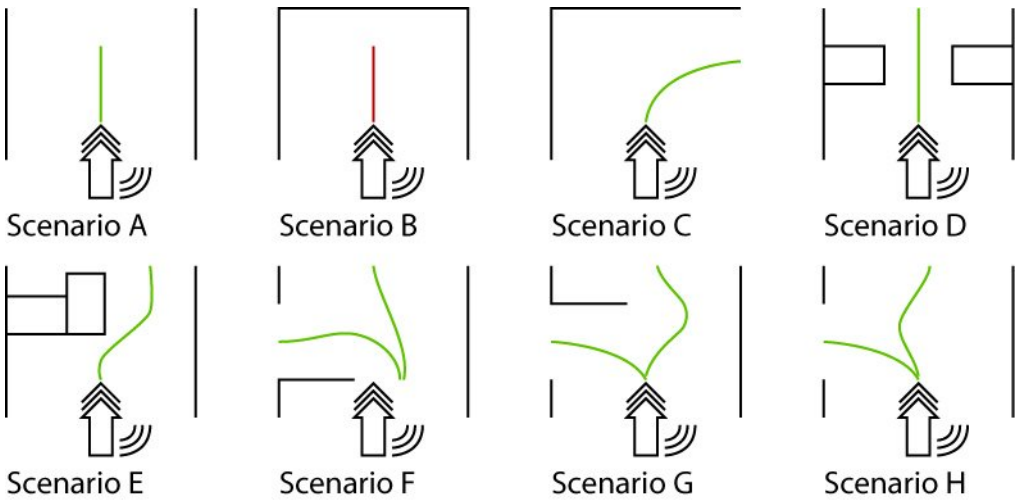

Figure 7.1: The proposed test scenarios A to H for the suitcase to pass. The lines illustrate the way the robot could take, where on *red* lines the suitcase should break and stop and on *green* lines the suitcase can continue driving.

Scenario C – Approaching corner: approaching a corner the suitcase should turn correctly and avoid getting stuck in the corner. If however it turns into the wrong direction it is supposed to stop.

Scenario D – Approaching two sided floor stricture: in the situation illustrated the suitcase is supposed to pass in between the two obstacles.

Scenario E – Approaching one sided floor stricture: this case addresses a scenario common for office floors. The suitcase should recognize the passage and take it.

Scenario F – Approaching open door (doorway behind the door): while this scenario is very similar to scenario E it has a passage immediately after the obstacle. The suitcase should omit the opened door and either continue driving on the floor or go through the passage.

Scenario G – Approaching open door (doorway in front of the door): an opened door can be approached from two sides. While scenario F addresses the doorway after the door in the way of the suitcase, this scenario has the doorway placed in front of the door. The robot therefore can directly go through the doorway or avoid the door and stay on the floor.

Scenario H – Door-like passage: Additionally to the two scenarios mentioned before a door can also be opened in two directions. While the cases F and G address a door opening toward the floor, this case covers the situation when a door opens toward a room.

The eight scenarios illustrated in figure 7.1 can be divided in three parts, according to their complexity. These parts are:

1. Basic behavior (A to C): these scenarios regard very basic behavioral aspects. In this part no obstacles are on the way of the robot. It addresses the detection of walls, the stopping and the turn behavior, as elementary tasks for the robot.

2. Artificial obstacles (D and E): using big plastic cases common scenarios can be constructed. Obstacles found in reality can be ideally recreated in a reproducible way. These scenarios are important in a scientific point of view.

3. Real obstacles (F to H): these scenarios address mainly doors, that are usual in office floors. As for the assignment the robot has to follow the floor of the AI Lab it is useful to test these scenarios separately.

Figure 7.2: The floor of the AI Lab at the University of Zurich. There are some tables and a lot of doors that can be opened to increase the difficulty for the 015 suitcase to pass.

The tests were all accomplished in the floor of the AI Lab. As the characteristics of this floor have an impact on the test results, the important measures are described in the following table 7.1, including the obstacles found there.

For each scenario at least five runs were performed with the following ranking:

- The test run was fulfilled (1 or +) when the suitcase took one of the ways proposed by the lines in figure 7.1.

Item	Measure
Floor width	$2.38m$
Door width	$0.80m$
Table width	$0.84m$
Part	**Material**
Floor	Carpet
Wall	Plaster

Table 7.1: The important measures of the AI Lab floor, where the test have been performed.

- The test run failed (-1 or –) when the suitcase touched an obstacle.
- Stopping in front of an obstacle without touching it was decided to be sufficient (0 or =).

For scenario C and D big plastic cases were positioned in the corresponding way. The rest of the scenarios was met by opening several doors according to the scenario. Fortunately there were doors that opened toward the floor as well as door opening toward the room.

Optimization of the algorithm was performed whenever a scenario could not be met. After an optimization of the algorithm was performed, the tests were restarted from Scenario A. The next section will provide an overview on the tests performed, with special respect to the last algorithm implemented and optimized. The summary of the most significant steps in this process is part of the following section.

7.2 Testing the Algorithms

The tests were started with the simple vector addition algorithm described in chapter 6.2.1. It showed a behavior, that was noticed in most of the linear algorithms. The steering angle never reached the full turn. It alway was very much centered and therefore the robot was not able to avoid obstacles of any sort. Even following a floor as illustrated in scenario A could not be achieved. As the robot never runs straight (see chapter 3.3) sooner or later it was approaching a wall, where it got stuck as it was not able to steer away from it.

By modifying the algorithm's steering behavior it was possible to make it avoid the sidewalls, but that resulted in a slalom like path, where the robot drove from one wall to the other, with very erratic changes in direction. This caused it to drive very slowly due to the constant proximity of the walls which is used to calculate the speed. The only scenario that could be met was scenario B, what made this algorithm not very promising for further development. For the detailed test results

see appendix A.1, where all the results and the changes made between the runs are listed.

This result has to do with the maximum angle the resulting vector can have. It is achieved, when only one of the outer sensors (sensor $S1$ or $S4$) had a length and the other sensors returned the minimal distance, what is not a realistic scenario.

The next series of tests was performed with the rang histogram algorithm described in chapter 6.2.2. Although the concept of repulsive forces did show slightly better results according to the erratic changes in direction, the over all behavior was not acceptable. Although the algorithm could be optimized to fulfill the criteria for all the basic scenarios without obstacles (see appendix A.2 for details on the results and changes made), the way it met the criteria was not promising.

After an analysis of the logged data it turned out, that the steering vector was very centered under most of the possible circumstances. Searching for a reason it was found, that while the creators of this algorithm used laser scanners to gather the distance data with a resolution of one degree over half a circle, the 015 project uses only four ultrasonic sensors that do not cover half a circle. In fact the rather complicated and costly range histogram algorithm did perform even worse than the simple vector addition algorithm, according to the over all performance.

Both algorithms basically suffered the same problem. The test runs showed that the steering angle did not use the full turning space. Therefore it was decided, that an algorithm appropriate for the robot could not be linear and the square repulsive forces algorithm described in the following section 7.3 has been developed.

7.3 Square Repulsive Force Algorithm

The algorithm developed for the 015 project is based on the idea of repulsive forces[1]. To give short distances more weight (as short distances signify a close object and therefore provide a bigger danger) the function could not be linear. A square function for repulsive forces seemed promising in the simulation[2]. As already stated in section 6.2.1 the vector for a sensor can be calculated according to its mounting angle α and its length $norm$.

$$\begin{aligned} \vec{v} &= \begin{pmatrix} v_x \\ v_y \end{pmatrix} \\ v_x &= \sin(\alpha) \cdot norm \\ v_y &= \cos(\alpha) \cdot norm \end{aligned}$$

[1] The idea of repulsive forces can be found in a variety of algorithms, for example [17, 5, 31].

[2] For simulation reasons an Excell sheet has been made, where the distances measured by the sensors could be entered and the resulting steering command is calculated. This makes it is more easy to compare different algorithms to another. See appendix C.

The repulsive force for a sensor is then calculated according to the following formula:

$$\begin{aligned} \vec{v_a} &= \begin{pmatrix} v_x \\ v_y \end{pmatrix} \\ v_x &= \frac{x_{max}}{x^2} \\ v_y &= \frac{y_{max}}{y^2} \end{aligned}$$

By summing up the four repulsive forces an over all repulsive vector $\vec{v_{repuls}}$ can then be calculated with

$$\vec{v_{repuls}} = \vec{v_1} + \vec{v_2} + \vec{v_3} + \vec{v_4}$$

As the alignment of the sensors is fixed, the costly sine and cosine functions can be exchanged with their value, what results in a simple multiplication. The resulting vector $\vec{v_{repuls}}$ is then directly used to calculate the steering command, where the angle of the resulting vector is used to determine the new direction and the norm is used to determine the speed (as seen in chapter 6.2.1).

While optimizing the algorithm and adapting it to the robot it turned out, that under some circumstances it did not perform very well. To improve the behavioral possibilities cases were introduced. The first case was the isolation of a wall approaching from the front followed by the isolation of a close wall on either side and a more general case where a very close object is detected. These cases provide the possibility to change the behavior of the suitcase in certain (predefined) situations. The situations mentioned above require a stronger repulsive force and a stronger turning of the steering, which can easily be implemented with the case distinction introduced. Above that, this design provides the possibility to change the architecture into a subsumption architecture by extracting separate classes from the cases.

7.3.1 Evolution of the Square Repulsive Force Algorithm

As this algorithm uses a square function to calculate the repulsive forces of an obstacle, near objects gain a much higher influence and therefore the range of values for the direction was suitably large. Figure 7.3 illustrates the process of this algorithm.

The first test runs with the scenario A still showed erratic changes in direction, which reduced the achieved speed significantly. An analysis of the logged information on the test runs showed, that the outer sensors were in a critical angle to the wall. That made them measure no wall at all, when the robot turned slightly away from

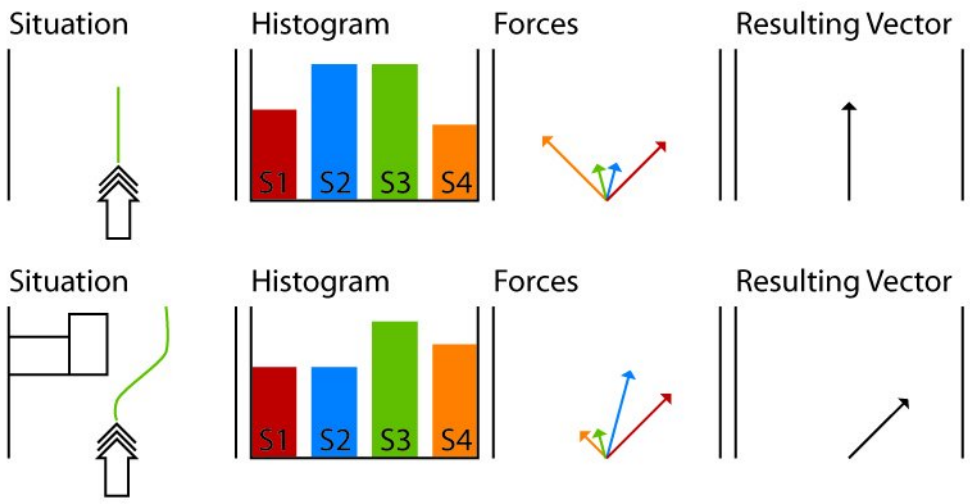

Figure 7.3: The square repulsive force algorithm.

it, suddenly changing to the correct distance when it turned slightly toward the wall. By changing the angle of the outer sensors to about 50° (-50° respectively) in relation to the length axis of the robosuitcase 015 this behavior changed completely. Now the robot was able to run along the floor with up to full speed and with very smooth changes in direction.

While the steering was smooth and the speed was fast, the stopping behavior as demanded by scenario B was a full flop. As the robot does not actively reduce the speed when approaching an obstacle the emergency situation was recognized far to late and it smashed into the wall with almost full speed. To ameliorate this very bad behavior, tests with an increased security distance were performed, which had an undesirable influence on later scenarios, such as scenario D and E, which were then emergency cases too. The solution to the problem was a case differentiation, which isolated a situation with a wall in front of the robot and a change in the behavior in this situation. This results in a subsumption architecture like hierarchical algorithm structure, where a higher structure can hold lower structures back [7]. In the so called wall case the robot reduces its speed and enforces the turning, to either stop before hitting the wall or turn away of it. The emergency distance could be reduced again without such bad accidents as experienced before. Still there were some situations where the robot touched an obstacle, namely when it approached the corner of an obstacle in a steep angle. By slightly increasing the security distance of the inner two sensors this problem was solved quickly.

To further improve the behavior in situations where the robot is very close to an obstacle and where it is near a side wall, two more cases were introduced. The close object case and the side wall case. When an object is very close the robot must steer with the maximum angle possible to have a chance to avoid the obstacle. The

second new case addressing walls on the side had to be introduced because there was a gap when the robot tried to turn because it detected a wall in front. When leaving the case the steering behavior changed to a smaller angle, what made it come to close to the wall and perform an emergency stop there. By adding the side wall case for this situation this behavior could be ameliorated. Figure 7.4 illustrates the resulting hierarchy. Now the robot is able to make the turn. Still this behavior is problematic, because scenario D is often recognized as a wall in front till the robot can make a distinction between the passage and the obstacles. Therefore a more secure wall avoidance results in a worse behavior in narrow passages. For the final adjustment it comes very close to walls, when trying to turn, what sometimes results in an emergency stop, but it can also handle reasonably small passages.

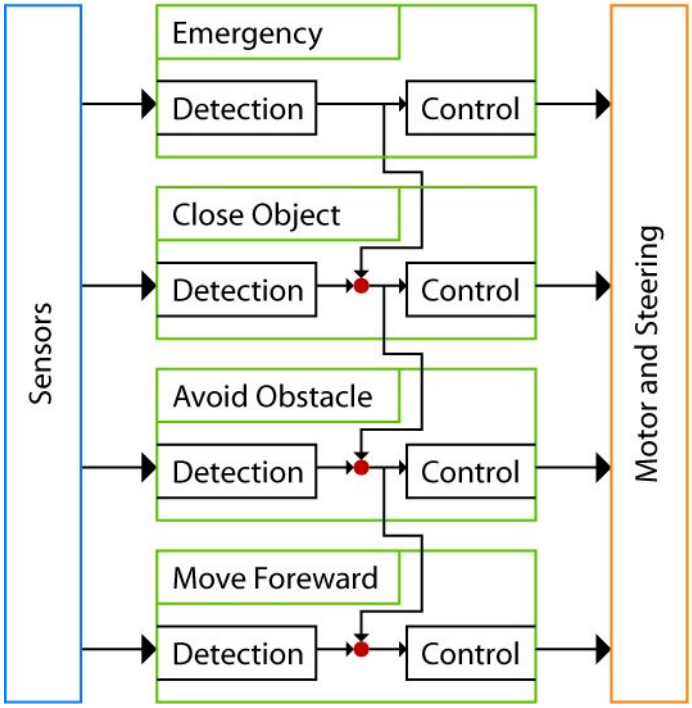

Figure 7.4: The hierarchical layers implemented in the square repulsive forces algorithm. Basically the robosuitcase 015 has to move forward. The layers above can disable the underlying layers. If an emergency occurs then all the lower layers are disabled. The close object layer addresses also the front and sidewall cases.

An analog problem occurred in scenario E. As the sensors take the distance of the closest obstacle in a range and these ranges cover the range of the neighboring senor at least partially an obstacle that is in the middle of the inner two sensors is detected as a wall in front. In a floor the robot has to get very close to an obstacle as in scenario E to make a difference between the passage and the closed side. Therefore it often fails to find the passage and has to make an emergency stop. However this problem does not occur when the obstacle is smaller and does not cover the middle of the floor. Opened doors for example were recognized and the passage could be

isolated. Therefore the behavior had not been changed for this scenario.

After these changes a complete test with all eight cases and five runs per case was performed which is summarized in the following section.

7.3.2 Final Test Results and Analysis

The tests showed, that the cases proposed are appropriate to evaluate the obstacle avoidance capabilities of the 015 suitcase especially with regard to development and optimization. They are easily reproducible and the ranking does not need special measuring techniques or equipment. That makes them fast and easy to use and the results with a given algorithm were reproducible.

The test results are categorized again into the three clusters good, sufficient and failed, where good indicates that at least three runs have passed with + and no test failed (–), sufficient indicates that less than three runs have passed with + and zero or one failed. Failed indicates that there have been two or more tests that failed.

While a lot of algorithms already failed with test A or B, it turned out, that scenario C is one of the most difficult to fulfill. Depending on the starting conditions the suitcase often turned too late and then had to stop in order not to touch the wall or it did not turn at all (see section 7.3.1). Though an emergency rule that turned into a random direction in such situations showed a better turn behavior, such a rule also had negative influence on other scenarios. Especially scenarios D and E failed regularly. Therefore it has been removed for the final tests which are summarized in table 7.2.

Scenario	Result	Comment
A	good	–
B	good	–
C	sufficient	Depending on the starting conditions the suitcase often turned too late and therefore had to stop.
D	good	–
E	good	–
F	sufficient	Under some circumstances the suitcase turned toward the center of the floor too early and then touched the door.
G	good	–
H	good	Though the suitcase did mostly turn toward the passage, it always stayed on the floor and never passed through the doorway.

Table 7.2: Results of the final obstacle avoidance algorithm implemented for the 015 suitcase. The detiled results for each run can be found in the appendix A.

Chapter 8

Conclusion

This assignment shows, that even with limited resources, a good dynamic behavior can be achieved. Only four distance sensors were necessary to enable a rich variety of obstacle avoidance behaviors, depending mainly on the morphology and the algorithms used, without the need of computationally expensive path finding algorithms or financially expensive sensors.

By an evaluation of various sensor types and techniques, first suitable sensors have been selected and attached to the microcontroller. Therefore a program had to be written, which implemented an activation strategy for the sensors. Four strategies were implemented and tested of which the serial approach has been selected.

The sensors then were integrated in a control program that implemented a behavior, based on the sensor data. The new program was designed to be easily modifiable and extensible. Then the morphological possibilities were analyzed and two different obstacle avoidance strategies taken from literature were implemented and added to the control program.

Through tests the drawbacks and disadvantages of these algorithms were shown. As they were not usable for the robosuitcase 015 a new algorithm was developed. This new algorithm, called square repulsive force algorithm, was then optimized and again tested.

The final test shows, that the obstacle avoidance algorithm proposed has a good overall behavior in the selected test cases. Further optimization could certainly improve the handling of the scenarios, that passed only with a sufficient grade.

The software developed within this assignment takes into account, that there will be further development on the robosuitcase 015 and through its strong encapsulation and easy to modify architecture it provides a good starting point for further improvement or even for other projects.

8.1 Outlook and Further Research

As this thesis has only taken care of the obstacle avoidance for the robosuitcase 015, there must be further development to achieve the aim of an autonomous suitcase that follows its owner.

To be able to follow a person through busy places, the algorithm surely must be improved to respect moving obstacles, as it was only tested in static environments. The next step should then be to find a solution for the problem of identifying the owner and determining his or her direction. Additionally the motor system has to be improved to work also on bumpy surfaces and to allow predictable backward motion.

During the thesis the idea came up, that it would probably be more feasible to build an autonomous suitcase carrier (where any suitcase can be placed in) rather than an autonomous suitcase, which would replace the luggage trolleys known from airports, what would provide a more independent solution. In this case it would also be possible that instead of the trolley following the owner of the luggage, the owner could follow the trolley which would lead him or her to the desired location.

This shows that there is still a lot of work to do, till the robosuitcase 015 reaches a state where it will be an everyday product, but as it is an interesting project with a lot of potential it sure is worth this effort.

Appendix A

Detailed Test Results

In this appendix the results of the tests performed with the 015 robotic suitcase are listed in detail. For each run the changes made are shortly described and the cases are listed in tables.

A.1 Results of the Vector Addition Algorithm Tests

The first test run is reported in table A.1.

For the second test run reported in table A.2, the following changes have been made:

Algorithm: Steering constants reduced (to increase turn behavior).

For the third test run reported in table A.3, the following changes have been made:

Morphology: Increased angle of the outer sensors (sensor 1 and 4).

Algorithm: Steering constants reduced (to increase turn behavior).

Algorithm: Increased security distance.

Algorithm: Introduced "braking" for emergency stop.

A.2 Results of the Range Histogram Algorithm Tests

The first test run is reported in table A.4.

For the second test run reported in table A.5, the following changes have been made:

Morphology: increased angle between all sensors (45° each).

Algorithm: steering constants reduced (increased turn behavior).

Scenario A		
Case	Result	Comment
1	-	Robot hit wall.
2	=	
3	-	Test aborted.
4	-	
5	-	

Table A.1: Results of the first test run with the vector algorithm.

Scenario A		
Case	Result	Comment
1	+	Very slow, due to erratic changes in direction.
2	=	Very close to walls.
3	=	
4	+	
5	+	
Scenario B		
Case	Result	Comment
1	-	No stop before hitting wall.
2	-	
3	#	Test aborted.
4	#	
5	#	

Table A.2: Results of the second test run with the vector algorithm.

Scenario A		
Case	Result	Comment
1	+	Still very close to walls.
2	+	Very slow.
3	+	Slalom
4	=	
5	+	
Scenario B		
Case	Result	Comment
1	+	Failure due to lag?
2	+	
3	+	
4	-	
5	+	
Scenario C		
Case	Result	Comment
1	=	No turn.
2	=	Approaching straight till security distance.
3	-	
4	=	
5	=	

Table A.3: Results of the third test run with the vector algorithm.

Scenario A		
Case	Result	Comment
1	+	Very slow, due constant wall proximity.
2	=	Erratic changes in direction.
3	=	
4	+	
5	+	
Scenario B		
Case	Result	Comment
1	-	No stop before hitting wall
2	-	
3	-	Test aborted.
4	#	
5	#	

Table A.4: Results of the first test run with the range histogram algorithm.

Scenario A		
Case	Result	Comment
1	+	Still close to walls.
2	+	
3	+	Very slow.
4	=	
5	+	
Scenario B		
Case	Result	Comment
1	-	Due to lag? Worked afterwards.
2	=	
3	=	
4	=	
5	=	
Scenario C		
Case	Result	Comment
1	-	No Turn.
2	=	
3	=	Approaching wall till security distance.
4	=	
5	=	

Table A.5: Results of the second test run with the range histogram algorithm.

A.3 Results of the Square Repulsive Force Algorithm Tests

The first test run is reported in table A.6.

For the second test run reported in table A.7, the following changes have been made:

Morphology: increased angle of the outer sensors (ca 50°).

Algorithm: steering constants optimized (improved turn behavior).

For the final test run reported in table A.8 and table A.9, the following changes have been made:

Algorithm: introduced case differentiation.

Algorithm: further optimization of the steering constants.

Scenario A		
Case	Result	Comment
1	+	Slalom path, slow
2	-	
3	=	Test aborted.
4	=	
5	+	

Table A.6: Results of the first test run with the square repulsive force algorithm.

Scenario A		
Case	Result	Comment
1	+	Some strange changes in direction.
2	+	
3	=	
4	+	
5	+	
Scenario B		
Case	Result	Comment
1	+	Failure due to lag?
2	+	
3	+	
4	-	
5	+	
Scenario C		
Case	Result	Comment
1	=	Behavior depending strongly on starting conditions.
2	-	
3	=	Test aborted
4	-	
5	+	

Table A.7: Results of the second test run with the square repulsive force algorithm.

Scenario A		
Case	Result	Comment
1	+	
2	+	
3	+	
4	+	
5	+	
Scenario B		
Case	Result	Comment
1	+	
2	+	
3	+	
4	+	
5	+	
Scenario C		
Case	Result	Comment
1	+	Depending on the starting conditions.
2	=	
3	=	
4	+	
5	+	
Scenario D		
Case	Result	Comment
1	=	
2	+	
3	+	
4	+	
5	+	

Table A.8: Results of the final test run with the square repulsive force algorithm, scenarios A to D.

Scenario E		
Case	Result	Comment
1	+	
2	+	
3	+	
4	+	
5	+	
Scenario F		
Case	Result	Comment
1	+	
2	+	
3	=	
4	+	
5	=	
Scenario G		
Case	Result	Comment
1	+	
2	+	
3	+	
4	=	
5	+	
Scenario H		
Case	Result	Comment
1	+	The suitcase sometimes seems to notice the open door.
2	+	(It moves toward it).
3	+	At the end it alway stayed on the floor.
4	+	
5	+	

Table A.9: Results of the final test run with the square repulsive force algorithm, scenarios E to H.

Appendix B

Software used in the Project

For the 015 robotic suitcase Project a variety of software has been used. As the project had to be runnable on a Apple computer, it sometimes was not that easy to find appropriate tools. For further development the software tools used are summarized in this appendix. All the tools are freeware mostly distributed under the terms of the GPL[1] [13]. This has two big advantages: first, there are no costs for the development tools, therefore more resources can be invested on the project itself and second, these applications exist for several platforms, what increases the portability of the project and makes it independent from the operating system.

B.1 Text Processing

As many tools in this section only exist for Mac OSX or Microsoft Windows the distinction between the two is made. The following list shows the tools to use with Mac OSX:

gwTEX: LATEXenvironment for Mac OSX.
`http://ii2.sourceforge.net/tex-index.html` (10.09.2006)

TEXShop: small and easy to use LATEX–editor.
`http://www.texniccenter.org/` (10.09.2006)

The following tools are for the use on computers with Microsoft Windows installed:

TEXnicCenter: very good free LATEX–editor.
`http://www.texniccenter.org/` (10.09.2006)

MiKTEX: LATEXcompiler for Windows.
`http://www.miktex.org/` (10.09.2006)

[1] GPL stands for the GNU General Public License. It can be read online at `http://www.gnu.org/copyleft/gpl.html` (30.12.2006).

Tools for both operating systems:

JabRef: free literature database for BibTEX–files, as they are used in LATEX.
`http://jabref.sourceforge.net/` (10.09.2006)

B.2 Web

firefox: comfortable web browser.
`http://www.mozilla.com/firefox/` (10.09.2006)

B.3 Programming

eclipse: extensive integrated Java development environment running on most platforms, including Mac OSX.
`http://www.eclipse.org/` (12.10.2006)

Subversive: SVN Plugin for eclipse (to download and edit code from subversion repositories).
`http://www.polarion.org/` (12.10.2006)

XCode: and the Apple development tools.
`www.apple.com` (12.10.2006)
This is a free set of tools mainly for C and C++ development of which XCode is the powerful editor. It comes with a C / C++ Compiler and other useful tools for software development.

B.4 PIC Programming

There are some freeware tools for Linux. Most also work on OSX, but they need to be compiled from source, what can be very exhausting. One set of tools that works on the Mac OSX operating system will be described here. To be able to compile and install the program sources the Apple development tools and XCode must be installed.

Most freeware tools are available through the fink package management system. Fink exists as disk image for the Mac and therefore is straight forward to install. It automatically gets the sources from the internet, compiles them and installs the packages.

fink: OSX package management system for GNU software
`http://fink.sourceforge.net/` (12.10.2006)
With fink packages can easily be installed by typing

```
fink -y install <package name>
```

The packages needed for this project will be listed in the following. For information on the packages see the corresponding link.

1. gputils – The GNU PIC utilities.
   `http://gputils.sourceforge.net/` (12.10.2006)

2. gpsim – Simulator for PIC microcontrollers.
   `http://gpsim.sourceforge.net/` (12.10.2006)

3. picp – pic burner for the PICSTART development programmer.
   `http://home.pacbell.net/theposts/picmicro/` (12.10.2006)

Not all the important packages are available through fink. To get picp working a serial-to-USB adapter is needed. In our case this adapter was a ATMEL UC-232A, which needs an additional driver to be installed.

PL2303 OS X driver: USB to serial dongle driver for many vendors.
`http://sourceforge.net/projects/osx-pl2303/` (12.10.2006)
This driver generates a new device called `/dev/tty.PL2303-3B1`. To make it more intuitive it can be linked to another name. To do so the following command must be typed:

```
ln -s /dev/tty.PL2303-3B1 /dev/ttyusb
```

Make sure the new name is not already used by some other device.

A very important package for the 015 project must be installed manually, the compiler for microcontroller C. The procedure will be described in the following description.

SDCC: Small Device C Compiler [10].
`http://sdcc.sourceforge.net/` (12.10.2006)
To compile this package download the source-code from the above address and do the following in the directory containing the unpacked sources:

```
./configure --prefix=/usr/local/
make
sudo make install
```

To enable the PIC16 port the microcontroller device libraries must be compiled separately. Therefore the GPUtils package is necessary. After installing it the following commands must be executed in the SDCC source directory:

```
cd device/lib/pic16
./configure
make
cd ..
make model-pic16
sudo make install
```

The headers must be installed to (again starting from the source directory):

```
cd device/include
sudo make install
```

The last step is to compile the I/O libraries. Therefore execute the following commands in the source directory:

```
cd device/lib/pic16/libio
make
sudo make install
```

If not all the I/O libraries are needed the file pics.build in the pic16 directory can be edited to only support the models required.

The prefix can be changed according the needs. The proposed way grants that the files are accessible in the shell without further configuration of the operating system.

With the proposed software setup PIC development on Mac OSX should be possible in an almost comfortable way.

Appendix C

Appendant CD Rom

This appendix is meant to give an overview on the contents of the appendant CD Rom.

In the root directory this text and an abstract in English and German as required from the university is located. Additionally there are two folders *Programs* and *Media*. These folders will be outlined in the following.

C.1 Programs

This folder is dedicated to the programs for the robosuitcase 015.

Null15AIControl This is the project directory of the Null15AIControl application. It can be opened directly in eclipse. It contains the runnable Java class files, the sources, configuration files and the source documentation.

The folder structure is as follows:

bin the Java class file directory.

conf configuration files for the Null15AIControl program.

doc documentation of the source code. This documentation is in the HTML format and can be viewed in any web browser.

images folder for the saved images from the camera.

lib additional libraries used for the Null15AIControl program (at the moment this is only the log4j library).

log the log files from the Null15AIControl program.

src the Java source files.

For those who always want the most recent version of the source code, I recommend the online SVN[1] repository at `http://code.google.com/p/project015/`.

[1]Subversion (SVN) is an open source application for revision control.

With eclipse and the Suversive plugin (see appendix B) the code can easily be downloaded.

Null15ControlCenter This is the project directory of the Null15ControlCenter application written by Jonas Bösch. It is an eclipse project too.

PICPrograms This folder contains the C sources for the two PIC microcontrollers.

C.2 Media

In this folder different media that was made through the assignment is collected.

Illustrator Adobe Illustrator files of the graphics in the assignment.

Other Files that did not fit in an other folder.

Pictures A selection of pictures that have been made from the suitcase.

Technical Documentation Technical documentation of the electronics used for the robosuitcase 015.

Thesis Sources The LaTeXsource code for the text of the thesis.

Videos Videos made from test runs.

List of Figures

List of Tables

Bibliography

[1] R. Basler. Koffer 015 design dokumentation. Unpublished, Feb. 2005.

[2] R. Basler. Koffer version 4 design dokumentation, erweiterung mit io-board für sensorsignale. Unpublished, Sept. 2006.

[3] J. Borenstein and Y. Koren. Obstacle avoidance with ultrasonic sensors. *Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation]*, 4(2):213–218, 1988.

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

[5] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, 1991.

[6] Brockhaus. *Brockhaus Naturwissenschaften und Technik*. F. A. Brockhaus, Wiesbaden, 1983.

[7] R. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988]*, 2(1):14–23, 1986.

[8] D. Caveney and J.K. Hedrick. Single versus tandem radar sensor target tracking in the adaptive cruise control environment. In *American Control Conference, 2002. Proceedings of the 2002*, volume 1, pages 292–297 vol.1, 2002.

[9] J. Crowley. Dynamic world modeling for an intelligent mobile robot using a rotating ultra-sonic ranging device. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 128–135, 1985.

[10] S. Dutta. *SDCC Compiler User Guide*, 4436 edition, 10 2006.

[11] A. Fattouh and S. Nader. Preliminary results on dynamic obstacle avoidance for powered wheelchair. In *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, volume 1, pages 849–853, 2006.

[12] Alex Foessel. Radar sensor model for three-dimensional map building. In *Proc. SPIE, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, volume 4195. SPIE, November 2000.

[13] FSF. Gnu general public license. `http://www.gnu.org/copyleft/gpl.html`, 2006.

[14] S. Ghoshray. Optimal path determination for a robot in a 2d workspace using quadtree modeling. In *Intelligent Engineering Systems, 1997. INES '97. Proceedings., 1997 IEEE International Conference on*, pages 175–181, 1997.

[15] S. Ghoshray and K.K. Yen. A comprehensive robot collision avoidance scheme by two-dimensional geometric modeling. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1087–1092 vol.2, 1996.

[16] F. Iida. *Cheap Design and Behavioral Diversity for Autonomous Adaptive Roobots*. PhD thesis, Unviersität Zürich, 2005.

[17] N. Katevas, S. Voliotis, T. Zahariadis, and K. Hrissagis. Range histogram: an environment representation method for indoor navigation of mobile robots. In *Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium*, pages 353–358, 2004.

[18] R. Kurozumi and T. Yamamoto. Implementation of an obstacle avoidance support system using adaptive and learning schemes on electric wheelchairs. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1108–1113, 2005.

[19] Y.D. Kwon and J.S. Lee. An obstacle avoidance algorithm for mobile robot: the improved weighted safety vector field method. In *Intelligent Control, 1995., Proceedings of the 1995 IEEE International Symposium on*, pages 441–446, 1995.

[20] J. McCarthey, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence. 1955.

[21] Microchip. *PIC16F877 Development Board V1.0 Manual*, 2001.

[22] Microchip. *PIC18FXX2 Data Sheet*. Microchip Technology Incorporated, 2002.

[23] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*. September

1980. Available as Stanford AIM-340, CS-80-813 and republished as a Carnegie Mellon University Robotics Institue Technical Report to increase availability.

[24] D. Oprisan and H. Rohling. Tracking systems for automotive radar networks. In *RADAR 2002*, pages 339–343, 2002.

[25] R. Pfeifer and F. Iida. Morphological computation: Connecting body, brain and environment. *Japanese Scientific Monthly*, 58, No. 2:48–54, 2005.

[26] R. Pfeifer and Chr. Scheier. *Understanding Intelligence*. MIT Press, Cambridge Massachusetts, USA, 1999.

[27] Sharp. *Sharp GP2D120 General Purpose Type Distance Measuring Sensors*, 2003.

[28] SRF05. Srf05 - ultra-sonic ranger technical specification, 2006.

[29] H. Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren*. Verlag Harri Deutsch, Frankfurt am Main, 1999.

[30] M. Toens, R. Doerfler, M.-M. Meinecke, and M.A. Obojski. Radar sensors and sensor platform used for pedestrian protection in the ec-funded project save-u. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 813–818, 2004.

[31] I. Ulrich and J. Borenstein. Vfh[*]: local obstacle avoidance with look-ahead verification. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2505–2511 vol.3, 2000.