ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/312529574

Integration of sensors on a mobile robot

Thesis · June 2001

CITATION
1

READS
1,194

1 author:

Geert De Cubber
Royal Military Academy
89 PUBLICATIONS 406 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

PhD: abnormal events' detection using UAVs View project

groundwater issue including exploration View project

All content following this page was uploaded by Geert De Cubber on 19 January 2017.

The user has requested enhancement of the downloaded file.

FREE UNIVERSITY OF BRUSSELS
FACULTY OF APPLIED SCIENCES
DEPARTMENT OF MECHANICS

INTEGRATION OF SENSORS ON A MOBILE ROBOT

-

INTEGRATIE VAN SENSOREN OP EEN MOBIELE ROBOT

Geert De Cubber

Promotor: Prof. Dr. Ir. P. Kool
Co-promotor: Prof. Dr. Ir. H. Sahli
Academic year 2000 - 2001

This work is submitted
to obtain the degree of
Mechanical – electro - technical civil engineer

Acknowledgements

This final term project couldn't have been terminated successfully or the results would have looked quite different without the kind contributions of many people whom I cannot thank enough for helping me solve the numerous problems on the path of me and the robot towards reaching our goals.

First of all, I have to be grateful to my promoters Patrick Kool and Hichem Sahli and to the original "father" of the robot, Ronald Van Ham, for conceiving this attractive project outside of the initially planned thesis proposals and by doing so, providing me with the opportunity to work on the very interesting subject of integrating mechanics and electronics. Their guidance and their effort to discuss and review intermediary results has kept me on the right track during the year.

Next, I'd like to thank Francis Decroos for the hours he spent with me debugging code, getting the camera to work and rewriting the target tracking program. Without his efforts, the camera sensor could never have been integrated and the final program would have lacked a lot of the reusability it contains now.

Many thanks go also out to a number of people providing essential information or support:

- Ping Hong of the Royal Military Academy for providing his target tracking algorithm and for his efforts in explaining and helping to adjust it to meet the requirements for our camera.

- Dirk Lefeber for his interest in my work, for his idea to solve the local minima problem and his information about a different implementation approach.

- Lixin Yang, Jan Ramboer and Bamdad Lessani for thinking about ways to speed up the calculations for the map-building process.

- The technicians Andre Plasschaert, Walter Van Praet, Jean-Paul Schepens and Thierry Lenoir for providing all kinds of necessary tools for the project and for helping me with some practical modifications of the robot.

- The departments Electronics and Mechanics for placing their infrastructure and material at my disposal.

- My colleague students for providing a pleasant working atmosphere inside and outside the laboratory.

Integration of sensors on a mobile robot

Abstract

The final goal of this project is to add some sort of “intelligence” to an existing pneumatic mobile robot and by doing this, making the robot capable of walking towards a certain designated target in a complex and unknown environment with multiple obstacles and this without any user interaction.

To realise this desired goal, some sensory equipment was added to the robot, in particular 2 ultrasonic sensors and a camera. This camera has the specific task of following the target object and returning its position, whereas the ultrasonic sensors have the more general task of retrieving environmental information. This information, coming from the different sensors, is brought together and fused in an intelligent way by a sensor fusion procedure based upon the principles of fuzzy logic. In order to be able to navigate in its environment, the robot makes use of the acquired sensory data to build a map – more specifically a potential field map – as a means of representing its surroundings. This map is used to plan the path to be followed and the actions to be undertaken.

A control program was written in order to gather and to coordinate all these different functions, making the robot capable of reaching the goals set up initially.

Integratie van sensoren op een mobiele robot

Samenvatting

Het doel van dit thesisproject bestond erin van een zekere vorm van intelligentie toe te voegen aan een bestaande mobiele pneumatische robot. Als gevolg hiervan moest de robot in staat worden om autonoom naar een bepaald doelobject toe te bewegen en dit in een voor hem totaal onbekende omgeving.

Om deze doelstelling te realiseren werden een aantal sensoren toegevoegd aan de robot, meerbepaald 2 ultrasoonsensoren en een camera. De camera heeft als specifieke opdracht het doelobject te volgen en te lokaliseren, terwijl de ultrasoonsensoren meer algemeen gebruikt worden om informatie uit de omgeving te halen. De informatie komende van deze sensoren wordt op een intelligente manier samengebracht door een sensorfusie - procedure gestoeld op de principes van de vage logica. Om succesvol te kunnen navigeren in zijn omgeving, bouwt de robot aan de hand van de sensorgegevens een map – meerbepaald een potentiaalmap - op als voorstelling van de omgeving, waarop hij dan de padplanning baseert.

Er werd een omvangrijk controleprogramma geschreven om al deze verschillende functies te bundelen en te coördineren. Aan de hand van dit computerprogramma is de robot in staat de gestelde doelstellingen te verwezenlijken.

Intégration de capteurs sur un robot mobile

Résumé

L'objectif de ce projet de fin d'études était de rendre un robot mobile existant un peu plus intelligent en tant qu'il serait capable de se diriger autonome vers un certain objet cible et ceci dans un environnement complexe et inconnu.

Pour atteindre cet objectif, quelques capteurs ont été ajouté au robot, en particulier 2 capteurs à ultrasons et une caméra. Cette caméra a comme tâche spécifique de suivre l'objet cible et de retourner ça position. Les capteurs à ultrasons ont la tâche plus générale de dériver de l'information sur les environs du robot. L'information de tous ces senseurs doit être mis ensemble d'une façon intelligente, ce qui est fait par une procédure de fusion de senseurs, basé sur les principes de la logique floue. Pour naviguer avec succès dans son environnement, le robot construit, à partir de les données des différents capteurs, une mappe – plus spécifique une mappe de champ de potentiel - représentant l'environnement. Cette mappe est alors utilisé pour planifier la route à suivre planning et pour déterminer les actions.

Un programme de contrôle a été construit pour réunir et pour coordonner toutes les différentes fonctions du robot. Sur la base de ce programme informatique, le robot est capable de réaliser les objectifs proposés.

Index

Chapter 1: Introduction

Objectives

The final goal of this project is to add some sort of "intelligence" to an existing pneumatic climbing robot and by doing this, making the robot capable of walking to a certain designated target in a complex environment with multiple obstacles and this without any user interaction. The robot has no prior knowledge whatsoever about its environment, so all the information it needs to move towards the target must be autonomously gathered during operation. The target to be reached was decided to be a red ball; in practice, a common basketball was painted red and used. Originally, it was the intention to work with clearly defined standard obstacles; in practice, any kind of obstacle will do, as long as they are not too high, too small, or too soft.

An important factor to keep in mind during the implementation of this project is the reusability of all the different components. This project is a symbiotic cooperation between the departments Mechanics and Electronics, but it can be expected that after the completion the robot will be somewhat disassembled to seize new projects. Thus, it is essential that the different departments can further take use of the components they are interested in, so the work isn't lost.

Material and techniques used

- The robot was originally built by Ronald Van Ham for his final term project [1]. The robot itself and some design adjustments are further described in the next chapter.
- In order to gain information about the environment, some sensory equipment is added to the robot:
    - A CCD Camera (Sony EVI-D31) continually tracks the target and retrieves its position.
    - Two ultrasonic sensors (Polaroid US 6500) detect obstacles in front of the robot.

  These sensors are discussed in detail in chapter 4.
- The sensor information has to be fused in an intelligent way. The robot makes use of a hybrid fuzzy logic based sensor fusion procedure to perform this task, which is discussed in chapter 5.
- The robot has to keep a map of the environment to find its way to the target. In this case, the map more specifically represents a potential field of the environment. Why this option was chosen regarding this subject can be read in chapter 6.
- As the robot must be able to advance in a complex environment, the path-planning task is not trivial. The potential field is used to determine the optimal move that can be made. There is no clear distinction between path planning and motion planning, as explained in chapter 7.

Chapter 2: Presentation of the construction

The robot itself

The robot was originally designed to be a climbing robot: it could climb walls by sucking itself to the surface with its suckers. Because the valves for the suckers are not controlled, this wall climbing capability is no longer present. The robot is able to take discrete steps of 23cm and turns of 16° by lifting its feet and turning the slider body.

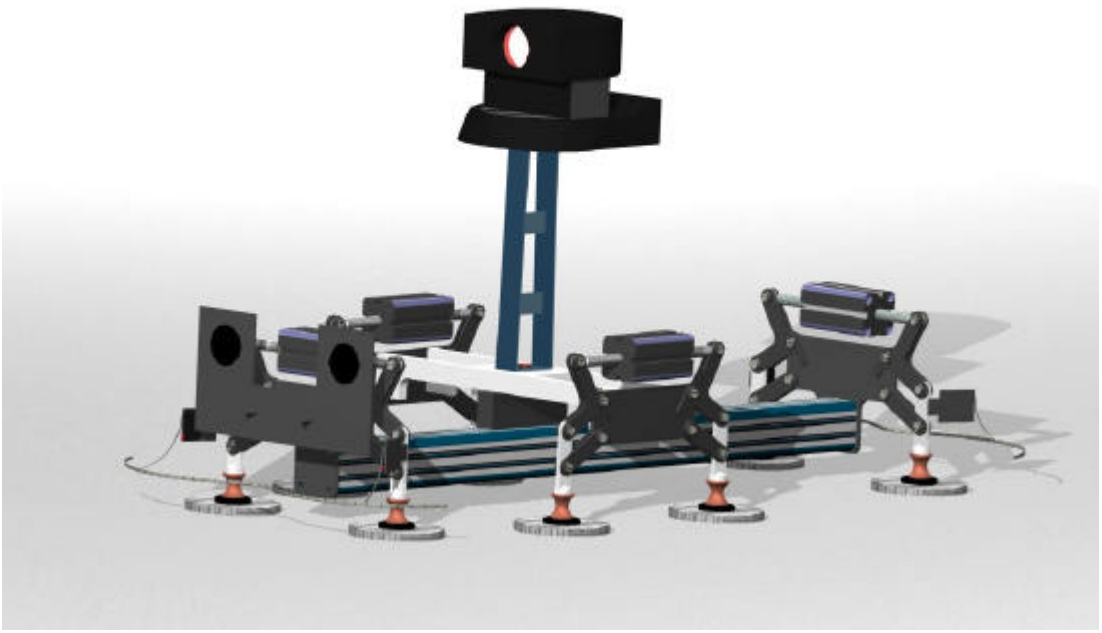

Figure 1 : Presentation of the robot

As no shorter distances can be travelled or angles can be made, this means that the spatial resolution the robot can reach is very coarse, which is an important factor concerning the path planning. The following graph shows the different points reachable within 6 steps beginning from the central position.

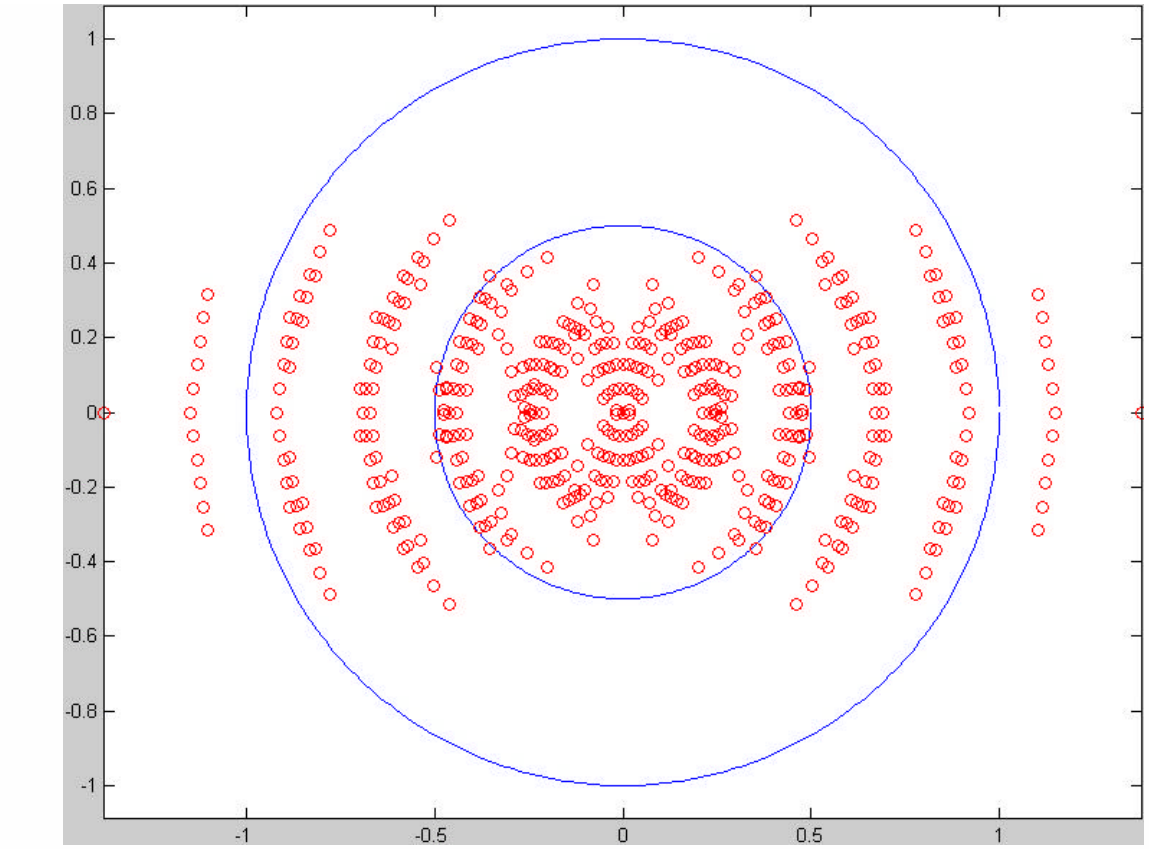

Figure 2 : Matlab graph of the points reachable within 6 steps

It is clear the robot lacks positioning precision for reaching sharp defined target points, which will limit it in its applications.

The turning speed of the slider body was brought down by stifling the appropriate valves because the robot had the tendency to slip through after the actual move. This slipping, which was not present when the suckers were used of course, caused intolerable positioning errors.

On-board electronics

As the control for the robot was completely brought over from microcontroller to PC, the existing microcontroller was removed in order not to damage it. With all the new control and electrical supply lines necessary for the ultrasonic sensors, camera and valves, it was to be feared that the robot would drag with it a whole bunch of wires. To counter this, it was decided to combine as much wiring as possible into one 25 – channel cable. Only for the S-Video signal from the camera, a different cable was needed.

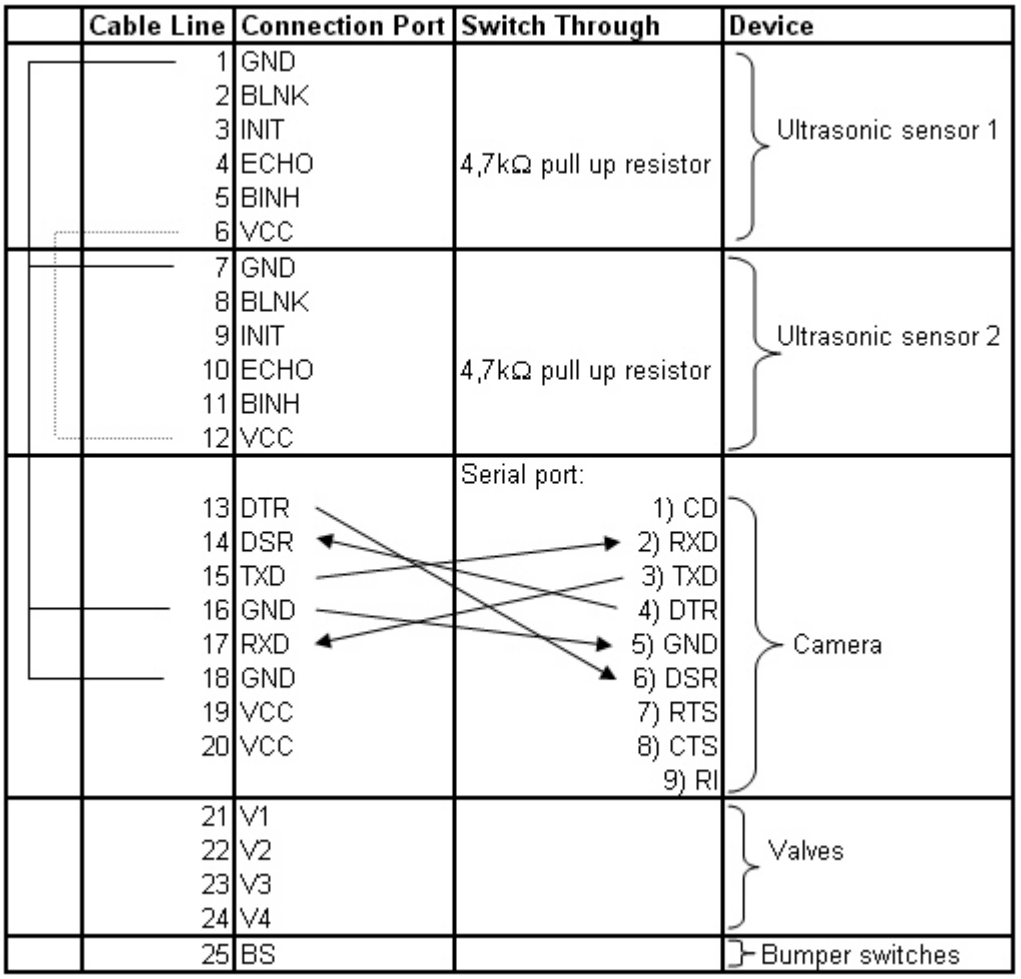

Figure 3 : Cable diagram

Sensors

The camera was set on an existing support, originally made for securing the robot. This stand is about 20cm tall, which is just what was needed for the camera. With a lower support, the field of view of the camera would be too limited, as obstacles and the robot itself would have blocked it. A higher stand would have resulted in excessive forces of inertia on the camera body.

The ultrasonic sensors were embedded, together with their ranging modules, in a plastic front panel. Aluminium was not used here, as for the rest of the robot, for reasons of electrical conductivity.

The bumper switches for direct collision detection were already present and they are still not unnecessary, even with the presence of the other sensors. They were just extended on the front side, to keep them in front of the board of the ultrasonic sensors. To save connection lines, all four signal lines from the bumper switches were combined into one signal.

On the 2 following sketches, some general changes made to the robot are visible:

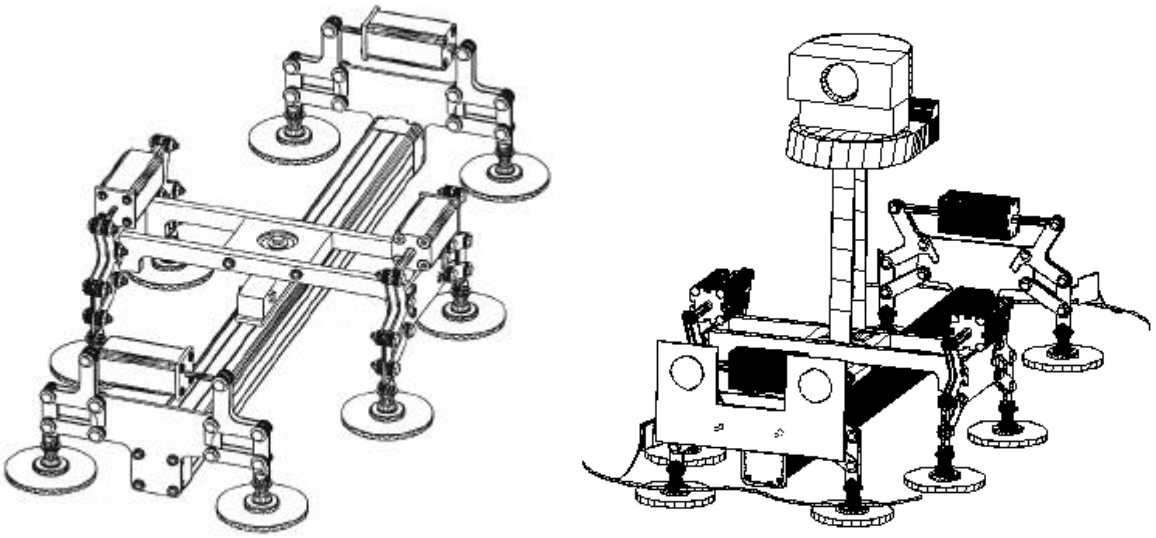

Figure 4 : The robot before and after the project

The external electronic system

The robot must be completely computer – controlled. The used computer is an Intel 266MHz PC with 128MB 66MHz SDRAM and the Windows2000 operating system. All these numbers may seem a bit superfluous, but they are important for the performance analysis of the controlling software.
The I/O operations for ultrasonic sensors, pneumatic valves and bumper switches are controlled by the Nidaq PCI 6025E digital acquisition board. This board has as most important features 32 I/O channels and an internal timer at 20MHz, which is needed for the measurement with the ultrasonic sensors.
The camera is controlled through a serial port RS232 interface, following the Sony VISCA protocol. The VISCA command reference can be found in appendix C. The camera video signal, respecting the S-Video standard, is processed by the Winnov Videum video capture board.

Chapter 3: Design of the control architecture

In theory

Introduction

In order for the robot to perform a non-trivial task, this task must be decomposed into a number of primitive actions of the actuators - in this case the valves and the different sensors - of the robot. This transformation of the task description from a high to a low level must occur during operation, because this is the sole possibility to make the robot capable of anticipating to changing environmental conditions and new sensor information. The idea is to draw up a detailed plan of the primitive actions and to execute these actions by primitive components that are controlled by a planning process on a higher level. The difficulty consists of making sure the right information is present at the right time, for the execution of the different sub – goals isn't that difficult, it is combining them that can lead to problems.

The use of abstract sensors and multi-layer data fusion – discussed in the chapters 4 and 5 - fits perfectly into the context of this approach. These concepts are extended as an even higher level of abstraction is used: the Logical Sensor / Actuator (LSA) [3]. An LSA does not only refer to a specific sensor or actuator, but to the whole well defined task performed by this sensor or actuator. This decomposition of the control procedure into subtasks is also very well suited for the object oriented programming approach, which is used in order to preserve the reusability of the different components. Moreover, it facilitates debugging and provides a logic building up of the program.

The control architecture defines how the different modules will be organised to come to a general plan-of-action, so the robot is able to operate fully autonomous. A number of existing classic approaches for designing such architectures are briefly discussed below:

SMPA - architecture

The "Sense-Model-Plan-Act" (SMPA) control architecture is a minimal architecture to connect the three basic skills of an autonomous mobile vehicle: mobility, perception and intelligence [2].

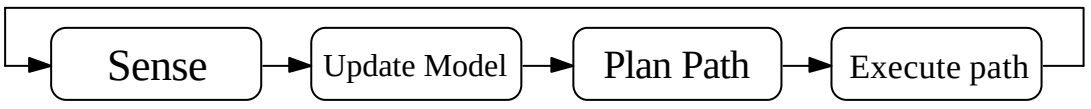

Figure 5 : Sketch of SMPA control architecture

The sensor system is used to collect data about the current environment (perception), the model (map) is updated, the path for this cycle is planned (intelligence), and finally this path is executed (mobility).

Blackboard architecture

In a blackboard architecture several agents or modules that are each responsible for a specific task, e.g. sensing, path planning, motion control, run in parallel and use a blackboard as communication medium.

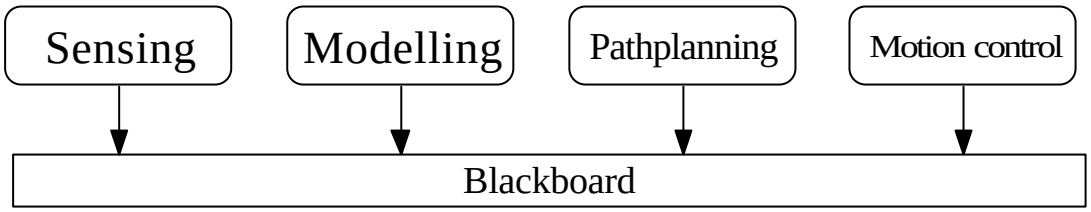

Figure 6: Sketch of the blackboard control architecture

This architecture, due to its parallel nature, is limited to multi-tasking or multi-processor systems. Basis of successful implementation is the definition of an efficient communication format. Another important factor is the level of decomposition of tasks for the single modules, which determines the intensity of communication.

Subsumption architecture

In the subsumption architecture, behavioural modules are defined as layers of decreasing priority. Each layer represents a single behaviour and higher levels can override (subsume) lower levels.

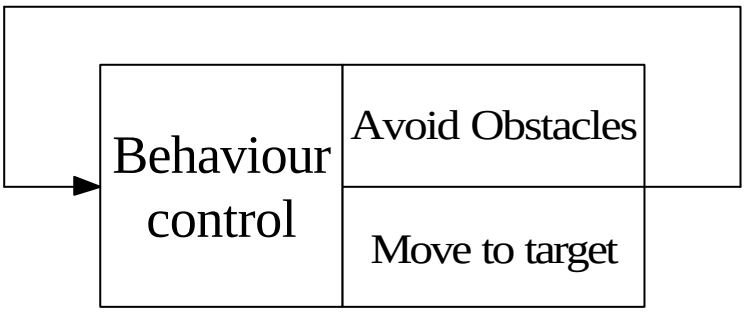

Figure 7: Sketch of subsumption control architecture

This type of architecture is used to implement behaviour-based navigation consisting of many behaviours with different priorities. In practice, one such a conduct needn't to completely override the other ones, but a sort of fuzzy intermediate behaviour can be adopted [7][10][12].

Hybrid architecture

Hybrid architectures are any combination of elements of the architecture types described above. They seek to combine the strengths of blackboard, SMPA and/or subsumption architecture and are of higher complexity and individuality. A general structure can therefore not be given.

Used architecture

The used architecture will be a hybrid one, consisting basically of an SMPA architecture due to its logical and straightforward reasoning. The blackboard architecture is incorporated for integrating the camera target tracking process with the simpler SMPA architecture, enabling the camera to continuously track the target while the robot is moving, sensing or thinking. This means there are two modules addressing the blackboard in this case: the camera target tracking process and the rest of the robot control program. In addition, some sort of subsumption architecture was brought into the controlling sequence of the robot. The robot is given 2 path-planning behaviours: one based on an extensive iterative calculation, and one extremely simple path planning behaviour when the robot notices that it is heading straight towards the target and that there are no obstacles blocking the way. It is clear that in this last case, it is not very "intelligent" to perform time-consuming map calculations, as all the robot needs to do is to step forward.

Note that these different architectures which are incorporated aren't actually treated equally as the SMPA approach is clearly dominating the control concept and the other structures are in fact only used at lower levels. In practice, the robot is capable of adopting different kinds of control architectures depending on the commands given, yet describing every single working mode is beyond the scope of this text.

In practice

Explaining the used control architecture

The general control flow – chart is given below:

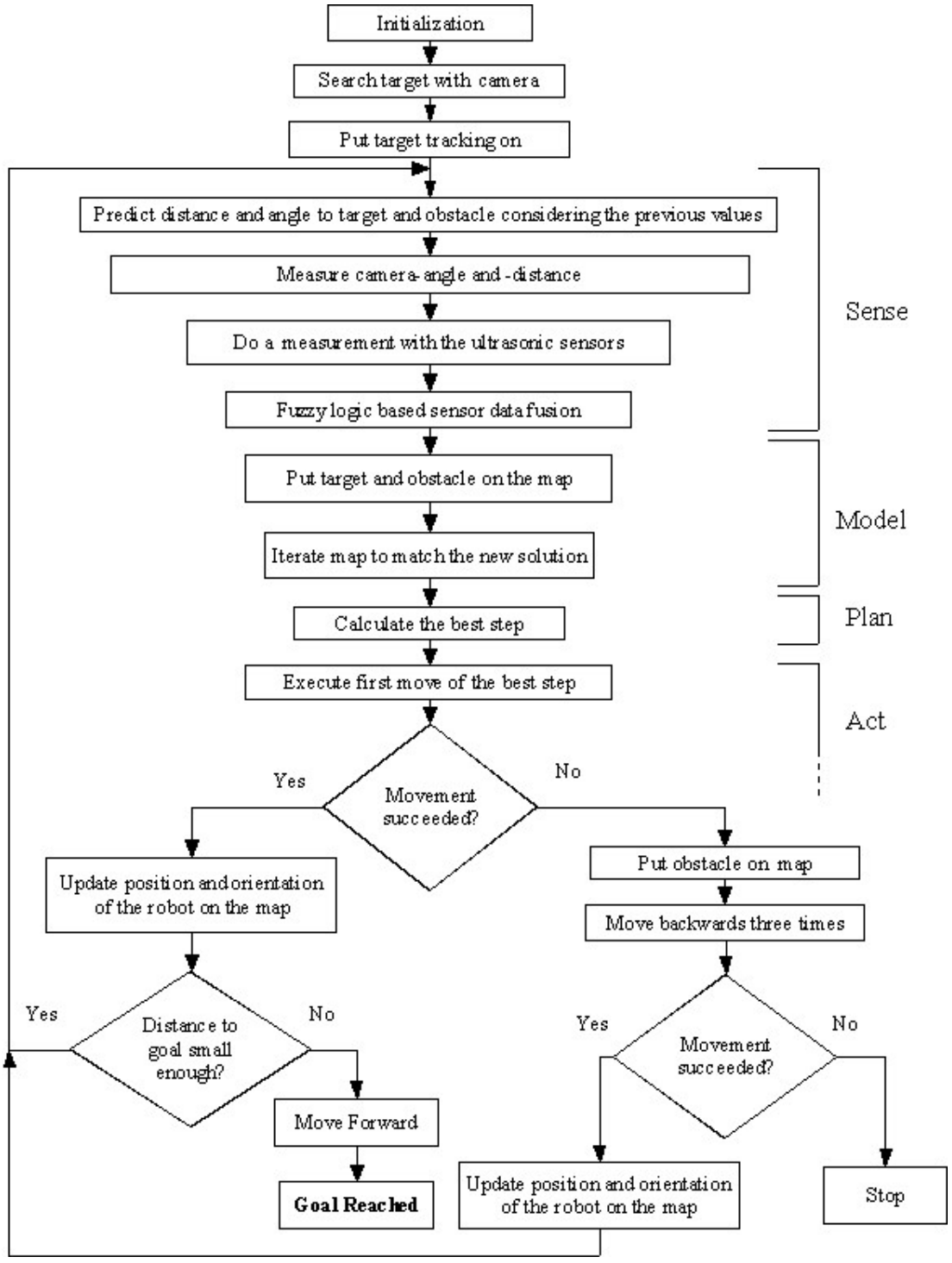

Figure 8: Used control architecture do not put obstacle on map here ^

As can be seen on the preceding figure, an SMPA architecture is used in general. The subsumption behaviour – based control is somewhat hidden in the path planning procedure. The camera target tracking procedure, which is switched on in the beginning, must be seen as a process running in parallel with the given program flow, realising as such a blackboard control architecture. The different modules are discussed more in detail in the respective chapters, but here is a short overview of the different tasks to be performed:

- The initialisation procedure sets up the digital acquisition board and the COM-port for the serial communication. It also puts the robot in its initial position with the slider body in front of the robot and all the legs down, as this is the most stable configuration for doing measurements.
- The search target procedure scans the area: it aims the camera into different positions and uses the target tracking routine to decide whether a target is present in the received image.
- The prediction procedure calculates where an obstacle or the target should be detected taking into account their former position and the knowledge of the step which was made in the mean time.
- The distance to the target, measured by the camera is read. As this camera is supposed to centre the target object in its image plane, the pan angle is also the angle towards the target. As these readings require the sharing of the variable constructions holding the needed parameters, the read and write operations to these variables must be carefully managed to avoid access faults.
- The ultrasonic sensors do their measurement
- The fuzzy logic based sensor fusion procedure retrieves the useful information out of the different sensor readings to come to unambiguous data.
- The internal map the robot keeps of its environment is updated with the newly acquired information, after which the map has to be iterated to represent the new solution. This is the modelling part of the control architecture.
- Based on this new information, an optimal step is calculated in the path planning procedure.
- Finally, the first move of the optimal step is executed. If the best step was for example to turn right and then go forward, the robot will only turn right and redo the measurement process from this new position. This approach is necessary to counter the extremely limited field of view of the available sensors.

The reactive nervous system

What is described above is actually the reflexive nervous system of the robot. It takes use of the ultrasonic and camera sensors to gain knowledge about the environment and plan its acts based on this information. The robot also has a sort of reactive nervous system, which takes as sensor input the signals from the bumper switches. When such a switch detects a collision, the movement is

immediately reversed. This capability is programmed within the movement sequence itself, is therefore instantaneous, and cannot be undone. This swift action is necessary in order for the robot not to damage itself. Only after performing the pull back movement, the robot will start thinking about the new actions that should be carried out. These actions are dependent on the specific situation:

- When the robot notices that a real collision was not possible, for example during the movement of the slider body, it will try to redo the undone movement.
- When a real collision has taken place, the robot will perform three steps backwards. Since the robot makes steps of 23cm and the ultrasonic sensors can only measure distances above 50cm, the obstacle should now be in view of the ultrasonic sensors if it is in front of the robot. If there are any problems during the execution of these three steps the robot will come to a full stop, otherwise the general control algorithm can continue from the new position.

The distinction made between control over the bumper switches and the other sensors can be compared to the human nervous system: humans also use a reflexive nervous system controlled by the brain with actions proceeding from reflection, and on the other hand, a reactive nervous system controlled by the spinal marrow, with reflex – actions.

Programming issues

The modular approach used during programming has as a result that just about everything is reusable. The different modules are reflected in the separate files included in the project source code. For the cooperation between the camera and the rest of the robot control, this division was even further carried through as two separate static libraries were implemented. One of those libraries takes care of the camera control and can therefore only be used for cameras responsive to the Sony VISCA protocol. The other library consists of the target-tracking algorithm and uses the first control library, but is fully camera-independent itself. Finally, the robot control program makes use of these static libraries to operate using the camera sensor. This approach has as a great advantage for later use that the camera libraries can run without using the robot, or that only the camera control library should be changed for example if another camera were to be put on the robot. The same is valid for the robot itself: if the camera or the camera control library is not used, all the other functions of the robot control program will stay available.

The different modules of the robot control program are:

- RobotGeneral.cpp: this module holds all the functions needed for the steering of the robot, as well as the initialisation and prediction routines.
- Ultrasonic_Sensors.cpp: this module completely controls the measurement with the ultrasonic sensors as well as all the statistical processing of the returned data from these

sensors. The module can run separately from the rest of the robot, so it can be reused by anyone who wants to do measurements with (2) ultrasonic sensors in the future.

- Fuzzy_Logic_Fusion.cpp: this module performs the fuzzy logic based sensor fusion process, but can actually be used as a programming tool to conceive all kinds of fuzzy logic controllers. User-friendly functions are made to compose membership functions, to aggregate rules, to fuzzify and to defuzzify variables. For new projects, the rules will have to be changed of course, and the membership functions will need to be edited, but this can happen fast and without a concern about the actual internal fuzzy logic programming background.
- PathPlanning.cpp: this module controls the complete map building and path planning process. As for the fuzzy logic module, it can run separate from the rest of the programming infrastructure. It initialises the map, puts obstacles and targets on the map, calculates potential fields, determines the optimal move and can even be used to calculate a start – to – finish path when used without the robot.
- PneuRobDlg.cpp: this is the actual main module combining and controlling the different submodules. This module takes also care of the frame grabbing process in order to completely separate the image acquisition and target tracking process. The final program provides a user-friendly dialog based interface showing the state of the most important system parameters during run-time, so the eventual operator can have an idea of what the robot is sensing and "thinking".

This general control program makes use of two static libraries for accessing the camera functions.
A first library "Huetracker" consists of the target-tracking algorithm. Most of the names of its different modules explain themselves; they are just mentioned to give the reader an overview of the different techniques and approaches used:

- Huetracker.cpp is the main library module controlling the actual target tracking process. It takes as an input the image grabbed by the general robot control program and provides as output camera actions to be performed by the camera control library and a distance estimation for the target.
- AdaptiveFilter.cpp
- FullStateFeedbackController.cpp
- KalmanFilter.cpp
- LowPassFilter.cpp
- PIDController.cpp
- SystemIdentification.cpp

The secondary library gathers all the camera control functions and is responsible for all I/O traffic with the camera. It is the only part of the program actually communicating with the camera, so it will be the only part to be changed when another camera should be installed.

Chapter 4: Sensors

Introduction

A sensor is a device that tries to make the link between the complex and chaotic environment and a well – structured computer, which can only work with formatted and limited data and this by quantifying one ore more environmental variables. Sensors can be subdivided into active and passive sensors. If a sensor merely records information that is present in the environment, it is called a passive sensor. On the other hand, if a sensor records information resulting from a certain sensor action, it is called an active sensor. In our case, the camera is a passive sensor, whereas the ultrasonic sensors are active sensors, because they emit an ultrasonic wave used for the distance measurement and change the environment by doing so.

Multiple sensors = multiple problems?

Information that can be derived from measurements with only one sensor – or one type of sensor - is always very limited. An "intelligent robot", defined by the JIRA as "a robot with the ability to comprehend its surroundings and to successfully accomplish a certain task within changing environmental conditions", wanting to autonomously find its way in a given environment, will therefore need information from different sensors to gain some sort of consciousness about this environment. The great advantage of using multiple sensors is redundancy. Indeed, though this creates difficulties for the processing of the information, it offers the opportunity of a drastic error reduction amongst some other advantages [5]:

- Multiple imprecise sensors may cost less than a few accurate ones.
- The reliability of the sensor can be increased.
- The efficiency and the performance of the sensor can be improved.
- Sensors may calibrate themselves
- The sensor architecture can be made more flexible, thus being better adapted to changing environmental conditions.
- In complicated environments, the information of multiple types of sensors is required to gain a correct image of the surroundings.

On the other hand, some difficulties and problems have to be taken care of:

- The fusion problem of al this sensor information is a very complex task and is still a field of much research.

- Too much information leads to an information overload, which is a problem, because the sensor fusion procedure is not only expected to produce a correct answer, it is also expected to do this in a time interval enabling real-time processing.
- When the system is extended with more and more sensors, the reliability of the system as a whole is also becoming more and more an issue, since this reliability will generally be decreased by adding components. On this topic, one has to make a compromise between the reliability of the sensor and the reliability of the system. Regarding this subject, no actual reliability study was made, because the robot works with very few sensors and it must be expected that eventually it will fail in its operation once one of these sensors fails. This does not mean the occurrence of a sensor returning totally wrong information mustn't be taken into account; on the contrary: this is an important aspect in the sensor fusion process.
- The advantages of self-calibration and flexible sensor architecture mentioned above are research domains in full development, but the results are not completely satisfying thus far.

Abstract sensors

Working with multiple sensors can be facilitated by introducing the terms abstract and concrete sensors, also called logical and physical sensors by some authors. Abstract sensors are not directly related to the physical reality, they return a certain measurement, but how this measurement happened is not important. This can be realised by using only one concrete, physical or real sensor; in this case, the sensor is called a simple sensor. When the measurement is performed by the combination of information from multiple sensors, this is called a sensor network. The camera and the ultrasonic sensors are all concrete sensors, but if the measurement from the ultrasonic sensors is further referenced, actually the combined reading from the 2 ultrasonic sensors is meant and this is an abstract sensor. The localisation predictions for target and obstacle that can be made are purely mathematical abstract sensors.

The type of sensor is closely related to the level of data fusion preceding the sensor reading. When talking about multi-sensor fusion, the first thing that comes to mind is the process of integrating measurements from different sensors to gain an unambiguous and correct image of the environment. Yet, this is actually only the highest step of data fusion, commonly called the decision step. The lower levels of data fusion are most often not recognised as such, as they are more system specific.

In this text, three levels of data fusion will be considered; although this classification certainly is not universal, other authors [6] [19] often use more levels such as the pixel, feature, symbol and the behaviour level.

- The lowest step of fusion usually takes place inside the sensor itself. Just think about a simple flow measurement where a time and a mass measurement are fused in an extremely simplistic manner. Similar low level fusion operations are performed within the ultrasonic

sensors, with a timer and a sort of pressure measurement explained later, and the camera, with the different pixels being brought together to form an image.

- A clear example of the medium level of data fusion is the abstract ultrasonic sensor, which consists actually of two different sensors. Through a fusion process, the resolution of these sensors can be increased and new information - such as an angle measurement – can be retrieved.
- Finally, the "measurement" of the different abstract sensors is presented to the decision step of the sensor fusion process.

It's only this latest process that is handled in chapter 5 about sensor fusion; the preceding two levels are specific for each of the abstract sensors, and are therefore discussed in the following paragraphs explaining more in detail these different abstract sensors.

Camera

In theory

The sole objective for the camera is to retrieve positioning information about the target to be reached, so the camera is not used to gain any extra knowledge of the environment. The used approach is to follow the goal with a target tracking procedure, written originally by Ping Hong of the RMA [4]. As stated earlier, this target tracking process runs in parallel with the rest of the robot control program, implementing a blackboard control architecture as such. The algorithm aims the camera towards the target continuously and returns a distance estimation for this target object. In the following paragraph, this target-tracking algorithm is further described.

Target recognition and tracking

The used algorithm was originally designed to track moving objects with a static camera as explained in [4]. In its new application, this situation is reversed as the target is standing still –though this is not really necessary– and the camera is moving with the robot. The target-tracking problem is in fact a camera control problem, as the camera must orientate itself in such a way that the objective stays in the centre of the image. The algorithm can be generally subdivided into the following stages:

- Colour classification

  In order to be able to work under changing lighting conditions, a learning phase is implemented during which a hue interval used for pixel classification and corresponding to the colour of the target is estimated. The hue-value is an angle representing a colour in the

HSI (Hue – Saturation – Intensity) colour space. This colour scheme uses as a basis the following circle:

- The hue value is simply an angle on the circle representing a colour. For example, at an angle of 0°, the colour is red.
- The saturation S is the distance from the centre of the colour triangle, represented at a ratio with respect to the edge of the colour triangle. A grey pixel will be given a saturation of zero, a red one a saturation of one.

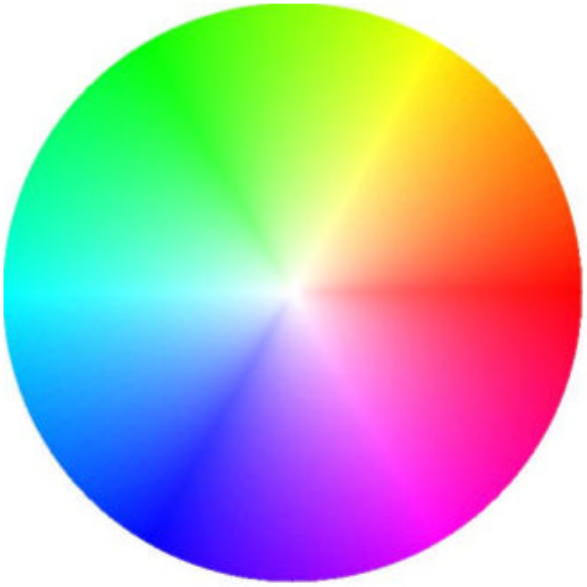

Figure 9 : Hue colour space

- The intensity I is the mean of the RGB colour values: $I = \frac{R + G + B}{3}$

- Shape detection

  In a following stage, a noise reduction is performed and a morphology filter is used to do image segmentation. During this operation, the image pixels are classified as belonging to the target or not belonging to the target. The morphology-filtered image is used to calculate a new hue region in order to preserve detection under changing lighting conditions.

- Camera control

  The camera pan (rotation around the vertical axis) and tilt (up-down rotation around the across horizontal axis) have to be controlled in such a way that the target is centred in the image plane. To be able to adapt to different dynamic behaviours of the target, an adaptive PI regulator is used based upon a Kalman filter [13].

- Window estimation

  To increase the signal – to – noise ratio and the sampling speed, the image segmentation process is not performed on the whole image, but only on a relatively small window surrounding the target. Because of this approach, the dimensions of this window must be recalculated after every step in the control process.

- Target position estimation

  The target position is written as a function of the pan angle, the tilt angle and a distance measurement. The angle measurements do not pose a problem; since the objective was

already centred in the image plane by the target tracking process, only these angles have to be read. The distance measurement is performed by comparing the target image size with the real target size taking into account the camera focal length. In practice, this measurement must be carefully calibrated to come to serious results.

In practice

The used camera is a Sony EVI-D31 CCD camera, which is controlled by the PC using an RS-232C serial control link following the VISCA protocol. In fact, this camera has a built-in tracking function, which was not used however, since this algorithm is not available and since the possible distance measurement that can be made using this algorithm is way too imprecise.

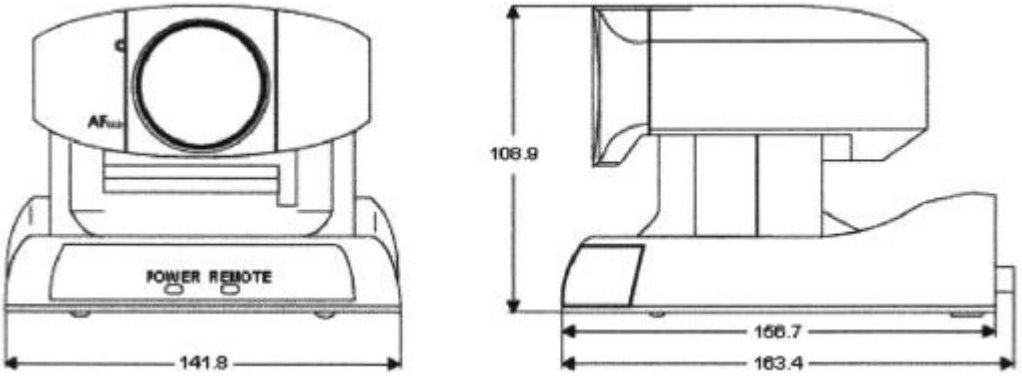

Figure 10: The Sony EVI-D31 camera

The camera was fitted on a support in such a way that the field of view of the camera could be maximised without exposing the device to extreme inertial forces due to the impulsive robot movements. This stand places the camera about 40cm above the ground level, leaving it a clear enough view when using its panning and tilting motors. The pan and tilt range is depicted on the following figure.

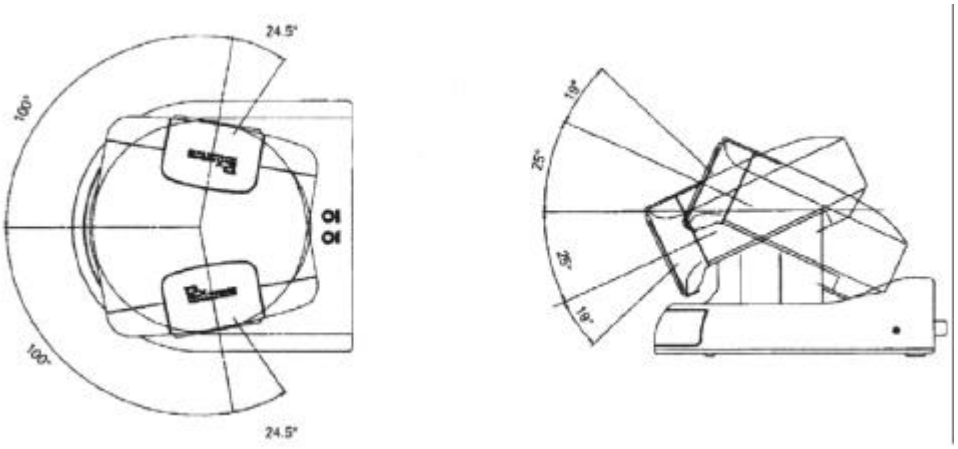

Figure 11: Pan and tilt range

The implementation of the serial communication link between the Windows 2000 based computer and the camera posed some problems at first, but after doing some tests linking two computers these growing pains were eliminated.

As mentioned above, an earlier developed target-tracking algorithm was used, which doesn't mean it needn't to be altered. A first problem was that the algorithm was written for a different video capture card, so the frame grabbing routines had to be rewritten. In order to avoid such problems in the future and to preserve the portability of the software, these routines were not just adapted to fit the video capture card which was used for this project. Instead, a frame grabber on the basis of Video-for-Windows was implemented, so the algorithm can now run from any computer using a Windows-based operating system, independently from the video capture board. Another problem was that the original target-tracking program was implemented as a single document interface, whereas the robot control program features a dialog-based interface. This means that the different object classes had to be rewritten to be able to insert the target-tracking algorithm into the main program. There were also some bugs left in the tracking program, mainly to blame to some inappropriate system constants and the tilting action of the camera control needed to be reviewed as it acted quite randomly. The used approach was to train the target tracking for the ball for a certain time. When the behaviour of the algorithm was considered appropriate, the system constants controlling the adaptive filter were stored. These values are set up during the initialisation procedure, so the target tracking does not need the extensive training in the beginning of the process anymore. This does not mean however, that this training does not further take place throughout the operation of the target tracking algorithm to improve the initial system parameters.

Experimental results

The camera distance measurement is performed by scaling the target object in the image plane to the known dimensions, so it is crucial this target object is fully recognised. Of course, this measurement needs to be calibrated before operation. A common source for errors is the lighting condition as the recognition of the ball will be more ore less optimal dependent on these different situations. The following graph shows a comparison of some calibration runs under different lighting conditions; however, since the tests were performed in Belgium, sadly no calibration runs could be carried out under a sunny atmosphere.

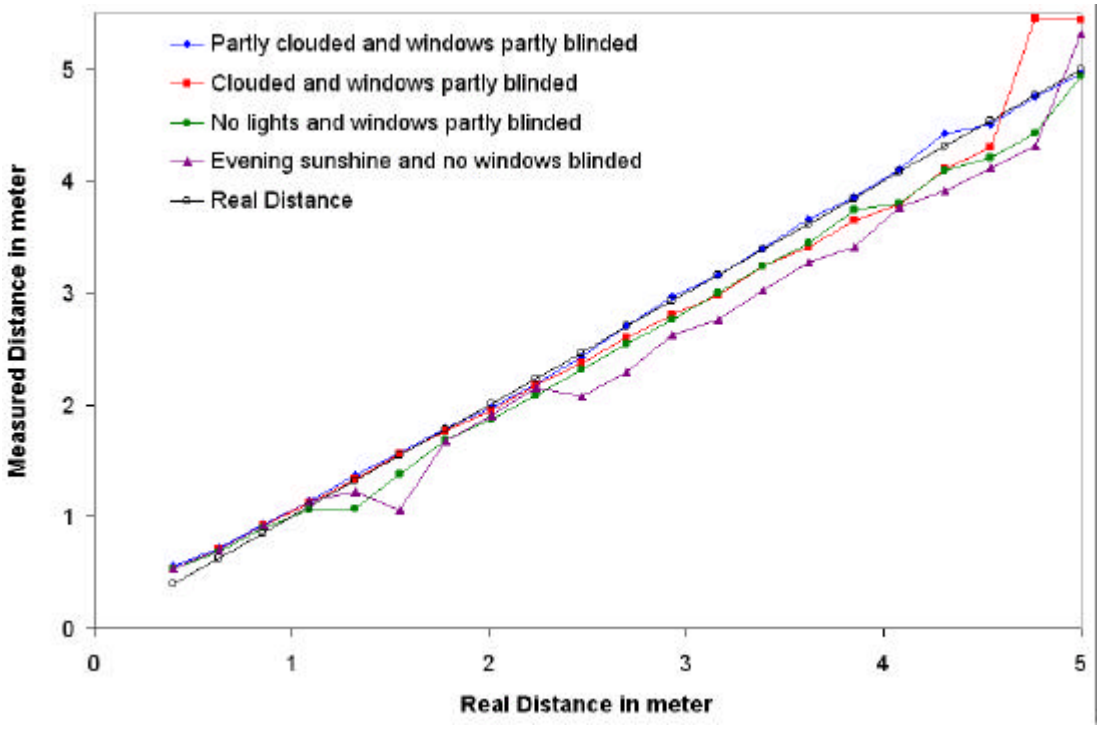

Figure 12: Camera calibration

The accuracy of the different sensors is an extremely important parameter for the sensor fusion and the map building process. Therefore, the error on the different abstract sensor readings is analysed, as shown here below for the error on the camera distance measurement.

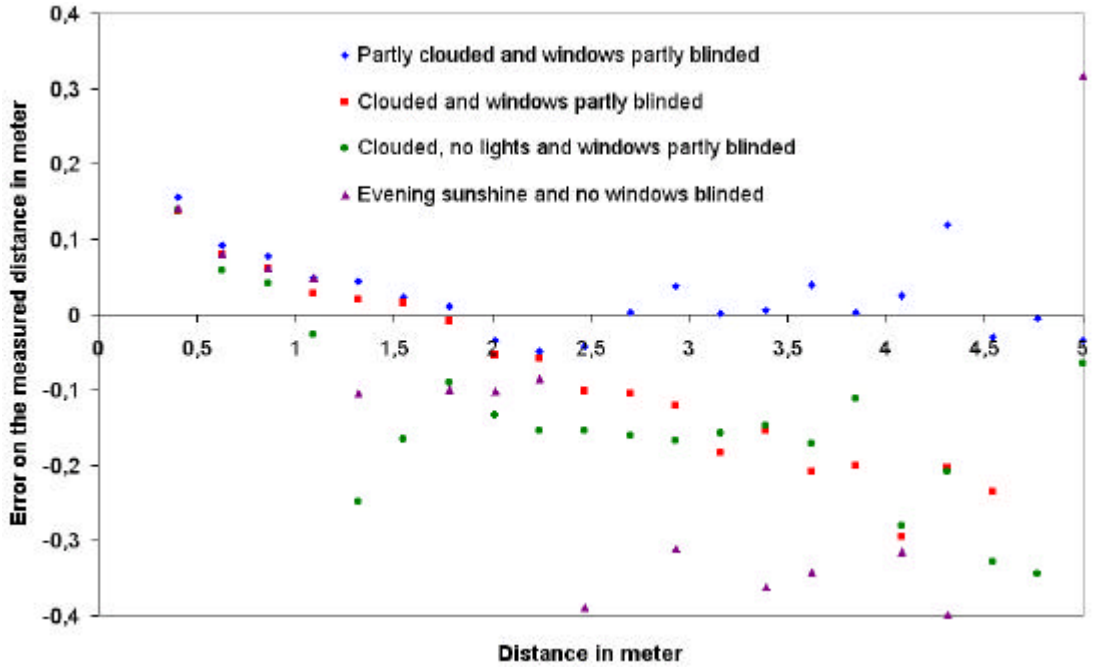

Figure 13: Error made with the camera distance measurement

One might note the slight overestimation at short distances, which can be explained easily by pointing out that at these short distances the camera view is becoming more and more blocked by other parts of the robot and shading effects become more and more important, hampering a full recognition of the ball. As the target object is not fully recognised, all these effects lead to an overestimation of the distance, which can be easily corrected mathematically however. At greater distances and with "darker" lighting conditions, the distance measurement tends to be an underestimation, which is caused by an erroneous recognition of the brown floor as part of the red target object, due to the lack of illumination. Eliminating the blinds of some windows behind the ball clearly is not a good idea, as can be seen by analysing the last data series in purple, which shows great and unpredictable errors due to the increased glare on the ball. When avoiding these difficult illumination circumstances, the distance measurement is capable of reaching an accuracy of about 10cm, which is very good looking at other experiments with only one fairly simple camera as is used here. The camera distance measurement is more accurate at shorter distances and less at longer ranges, which is a behaviour that fits the robot fine, because for the robot it is not that important to know the position of the target at great distances very precisely, it is only at shorter ranges that this becomes more imperative.
Nevertheless, one must take into account the random error on the distance measurement as an important source of error regarding the returned data from the camera. Other sources of errors are changing lighting conditions and imprecise calibration. It is always possible that at some stage, the target is lost; for example, when it goes out of range of the panning area of the camera or when the tracking algorithm simply fails. In these conditions, the robot will not be able to advance, as it has no renewed target information, so this situation has to be avoided in every case.

Ultrasonic sensors

In theory

Physical working principles

An ultrasonic sensor emits a series of short ultrasonic pulses and measures the time till a returned echo is detected. As the speed of sound is known, this time reading is actually a distance measurement to the object responsible for reflecting the acoustic wave. The ultrasonic wave is emitted by a membrane, which is excited by an alternating current. The same membrane is used to detect a returned wave as it gets excited by this echo.

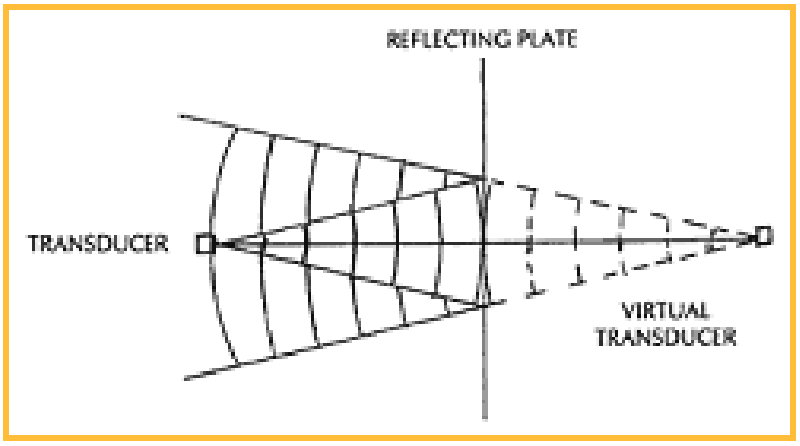

Figure 14: Ultrasonic distance measurement

Since a certain transition time is necessary to switch from emitting to receiving function, there exists a minimal detectable distance for each ultrasonic sensor. This minimal distance is about 50cm, meaning that an object closer to the sensor than half a meter will nevertheless lead to a 50cm distance reading. This is the main reason why the bumper switches are still useful for detecting obstacles in the near surroundings of the robot.

The situation on the figure above where an ultrasonic wave is represented as a line does not really correspond to the physical reality. In fact, the wave front propagates at a first Fresnel stage in the form of a cylinder, whereas in a second Fraunhofer stage, the wave diverges and a more or less conical form is adopted.

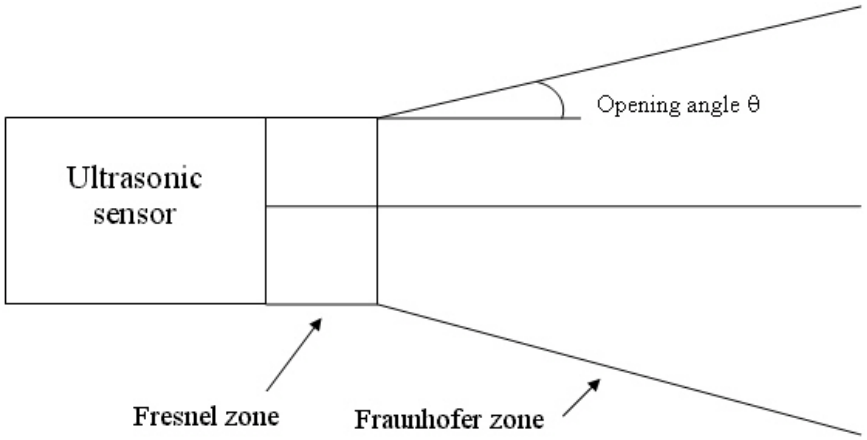

Figure 15: Fresnel and Fraunhofer propagation stages

Only the Fraunhofer zone is really important for doing ultrasonic distance measurements, the wave pattern adopted during this stage is called “the sonic bow” and has the following typical form:

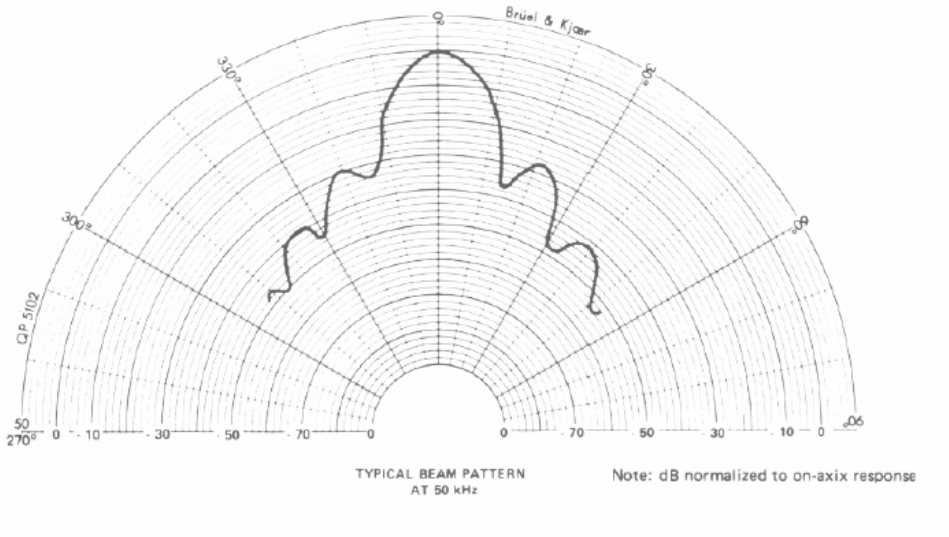

Figure 16: Wave pattern for the Polaroid US6500 Ultrasonic Sensor

Medium level sensor fusion

The above wave pattern must be taken into account when the distance to a certain object is to be estimated correctly. If only one ultrasonic sensor is present, a detected object can in fact be located anywhere on the sonic bow, so there is an intolerable uncertainty on the angle measurement. By implementing a medium level fusion of the readings of different ultrasonic sensors, this problem can be evaded, but first let us take a closer look at the problems arising when using the classical model of separate ultrasonic sensors. In this case, the best estimation concerning the location of the detected object is in the middle point of the bow, not because this point has a higher probability than the others – experiments have shown a uniform probability distribution – but just because the maximum error is minimized by assuming this. Yet, by using this simple assumption, an error is made which isn't negligible, as is made clear on the following figure:

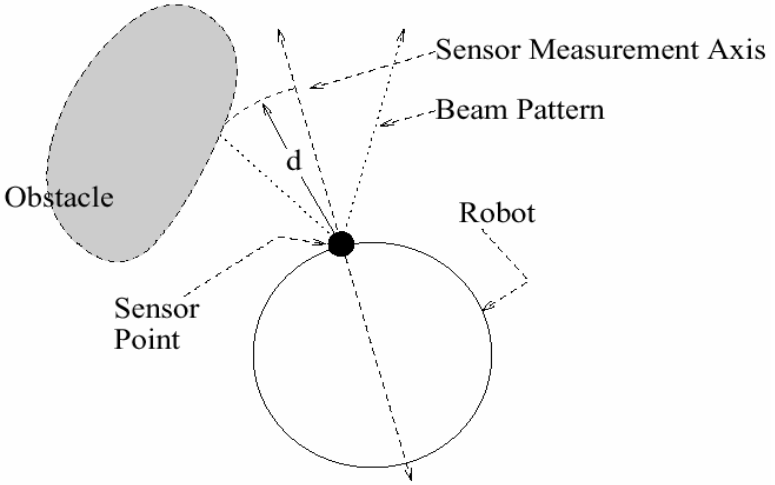

Figure 17: Error made by choosing the middle point as the object position

When using the ultrasonic sensor measurements for the detection of doorway passings, this approach fails in correctly perceiving the environment, as is shown on the following figures:

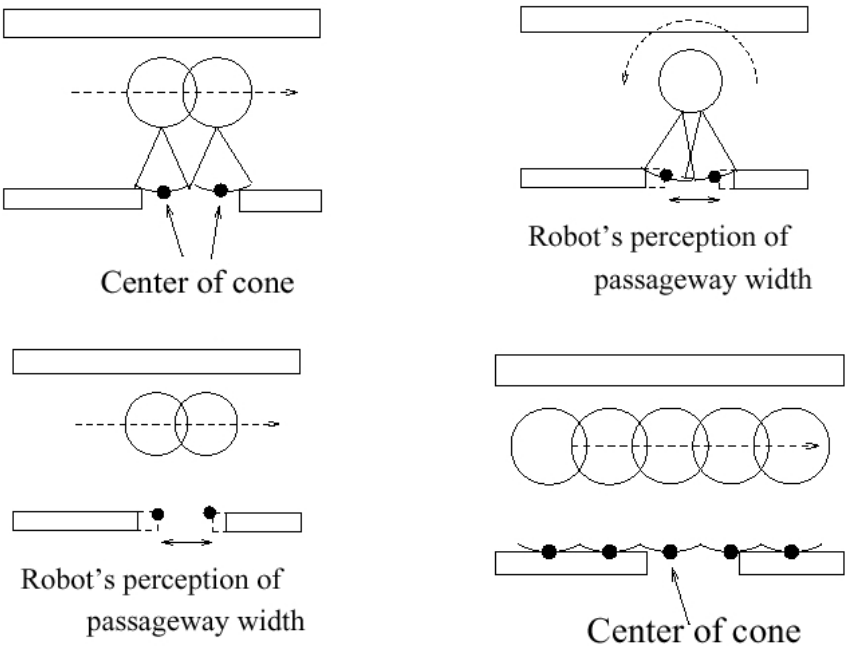

Figure 18: Errors made by choosing the middle point as the object position

By choosing the middle points of the sonic bows as the object location, the aperture seems smaller or even worse: it seems there is no opening.

A first method to deal with the fusion process is implementing a simple triangulation of sensor readings [11]. This approach is explained below using only the two ultrasonic sensors that are available on the robot. The following situation occurs:

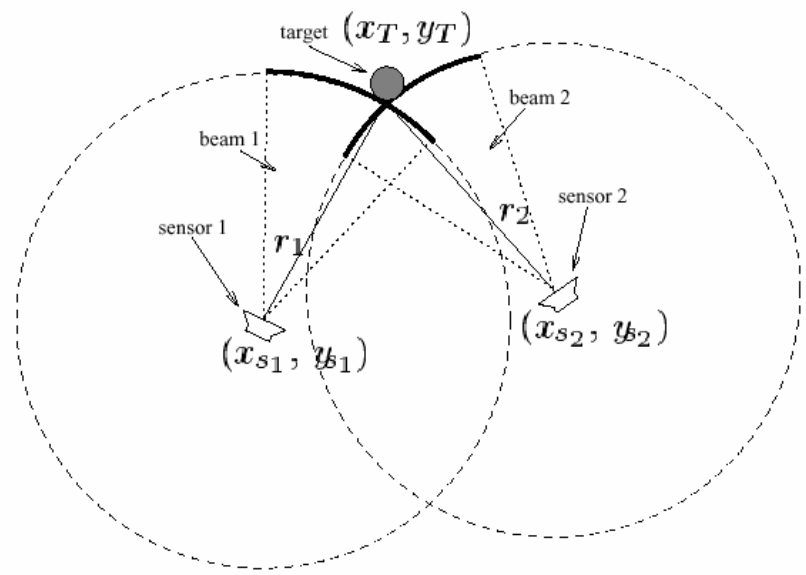

Figure 19: Triangulation of ultrasonic sensors

The point of intersection of the two arcs defines the location of the object. Since the sonic bows aren't really arcs of a circle, an error is made here too, but considering the small opening angles of the ultrasonic sensors, this error stays within acceptable limits.
This point of intersection can be calculated by expressing that the target with coordinates (x_T,y_T) belongs to the two arcs using the following system:

$$(x_T - x_{s1})^2 + (y_T - y_{s1})^2 = r_1^2$$
$$(x_T - x_{s2})^2 + (y_T - y_{s2})^2 = r_2^2$$

The solution of this system can be written as:

$$y_T = y_{s1} + \frac{1}{c^2}(b.d \pm |a|.\sqrt{r_1^2.c^2 - d^2})$$
$$x_T = x_{s1} \pm \sqrt{r_1^2 - (y_T - y_{s1})^2}$$

with:

$$a = x_{s1} - x_{s2}$$
$$b = y_{s1} - y_{s2}$$
$$c = \sqrt{a^2 + b^2}$$
$$d = \frac{r_2^2 - r_1^2 - c^2}{2}$$

Enabling us to calculate x_T and y_T.
As relative coordinates are used to designate the positions of the ultrasonic sensors, the coordinates for the target are relative too, but this does not pose a problem since they are used to calculate a relative angle and distance:

Suppose $y_{s1} = y_{s2} = 0$
Suppose $x_{s2} = -x_{s1} = e$ (sensors are aligned with the origin in the middle; e = 8cm)

$$\text{Distance} = \sqrt{(x_T - 0)^2 + (y_T - 0)^2}$$

$$\text{Angle} = atan(\frac{y_T}{x_T})$$

This triangulation method returns the wanted parameters, yet the reliability of this "measurement" must be questioned since the doorway detection problem is not solved completely as is shown in the following figure:

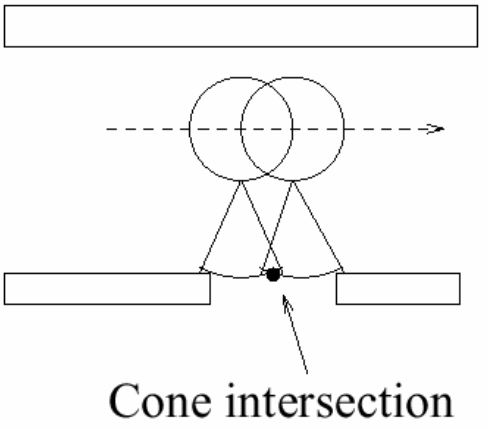

Figure 20: Error when using the triangulation method

The two cones do intersect, but the intersection point is not a good representation of the position of the object and thus of the environment. A remaining possibility to improve the sensor readings is implementing the Arc Transversal Median or ATM method [17]. In this approach, the location of an object is determined by intersecting one arc with other arcs whose angle of intersection exceeds a threshold and then taking the median of the intersection. If multiple arcs intersect in the same point, the presence of an object in this point becomes more and more probable, yet this approach assumes more than two sensors of course. Taking the median of those intersection points assures a drastic noise reduction and therefore the robustness of the result. The introduction of a threshold level for the angle of intersection between two arcs is done in order to eliminate unstable intersections. Two cones stably or transversally intersect if their intersection does not significantly change after one of the sets is slightly perturbed. The reason for this limitation is that no accurate conclusions can be derived in the unstable case. The following figures explain the difference between stable and unstable intersections:

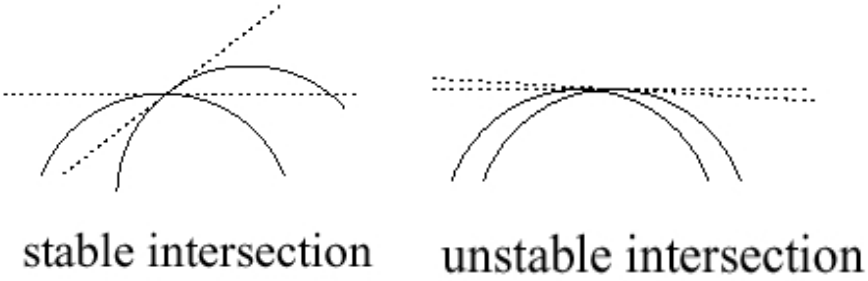

Figure 21: Stable and unstable intersections

In general, a threshold angle of 30° is used when implementing this technique. The robot does not make use of this method, because if it did, simply all measurements would be eliminated since they would all be considered as unstable intersections. This observation marks an important limiting factor regarding the reliability of the ultrasonic sensor measurement. The cause for this problem is that the two used sensors are positioned too close to each other, but there was no real other option for placing them since the dimensions of the robot had to be taken into account.

In practice

At a first glimpse, the ultrasonic distance measurement seems extremely simple to process. A plain time measurement is performed and the required distance can be calculated immediately:

$$d = \frac{1}{2}.c.t$$

With c being the speed of sound and t the measured time.
Their product has to be divided by two since the acoustic wave travels twice the distance to the detected object. However, the ultrasonic distance measurement has to cope with some additional problems limiting the accuracy of the readings.

Influencing factors

A first influencing factor is the temperature as the speed of sound is a function of this parameter:

$$c = \sqrt{g.R.T}$$

The robot is for the moment only used in inside laboratorial conditions, so the temperature variations aren't important, but when the robot should be put in the outside world, the ultrasonic distance measurement should be corrected according to the following formula:

$$d_{corrected} = d_{measured}\sqrt{\frac{T_{environment}}{T_{reference}}}$$

$T_{reference}$ being the temperature of 293K used for calibration.
The most important factor limiting the range of view of the ultrasonic sensors is the attenuation of the acoustic wave in the propagation medium, in this case the air. This attenuation can be formulated as a decrease of the wave intensity as a function of the distance travelled R [23]:

$$I = I_0.\frac{e^{-a.R}}{4p.R^2}$$

With I_0 the initial intensity and α the attenuation coefficient of the propagation medium.

To counter this effect and preserve the perceptibility of objects at greater distances, the sensors use a gain control, which is an amplification of the incoming signals as a function of the time passed since the start of the measurement.

The frequency of the acoustic wave plays an important role and defines the capabilities of the used sensor. A higher frequency leads to a more directive wave, a shorter range due to more attenuation and a shorter wavelength permitting to detect smaller objects, so the used frequency of 49.5 kHz must be seen as a compromise between range and resolution.

A last factor influencing the ultrasonic distance measurement is the reflection characteristic of the detected object. As an acoustic wave reaches a certain surface, a part of the energy is absorbed and another part is reflected in a diffuse or specular manner. Assuming specular reflection, the intensity of the returned echo will be:

$$I = K_r . I_0 . \frac{e^{-2a.R}}{16p^2 . R^4}$$

With Kr the coefficient of reflection of the surface, depending on the surface characteristics of the object. In practice, the manner of reflection isn't purely specular, nor purely diffuse, but the specular characteristic clearly takes the upper hand as it can be shown that oblique objects aren't well detected. This was by the way an important factor for the decision to work with a ball as a target object as a wave front will always collide with the surface of a sphere normally. Another advantage of using a sphere is that multiple chances of detection are possible. Indeed, tests comparing detection of a cylindrical and a spherical object showed that the detection of the sphere was better, except at long distances where the more diffuse characteristic of the spherical surface presented problems. A simple explanation for this better detection at short distances is that with the sphere the possibility of a reflection to the ground is possible as shown in the next figure:

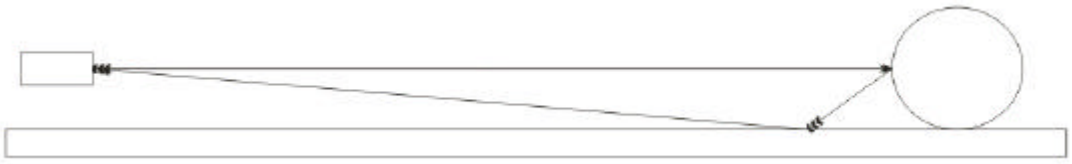

Figure 22: Multiple echoes when using a spherical object

When the directly returned wave is missed for some reason, the sensor gets a second chance when the wave reflected by the ground reaches the membrane. Measuring this second wave will lead to an overestimation which is negligible in practice, so the reading is still very useful.

Sensor control and operation

The Polaroid 6500 ultrasonic sensors are equipped with a ranging module to provide a digitalized user interface to control the sensors.

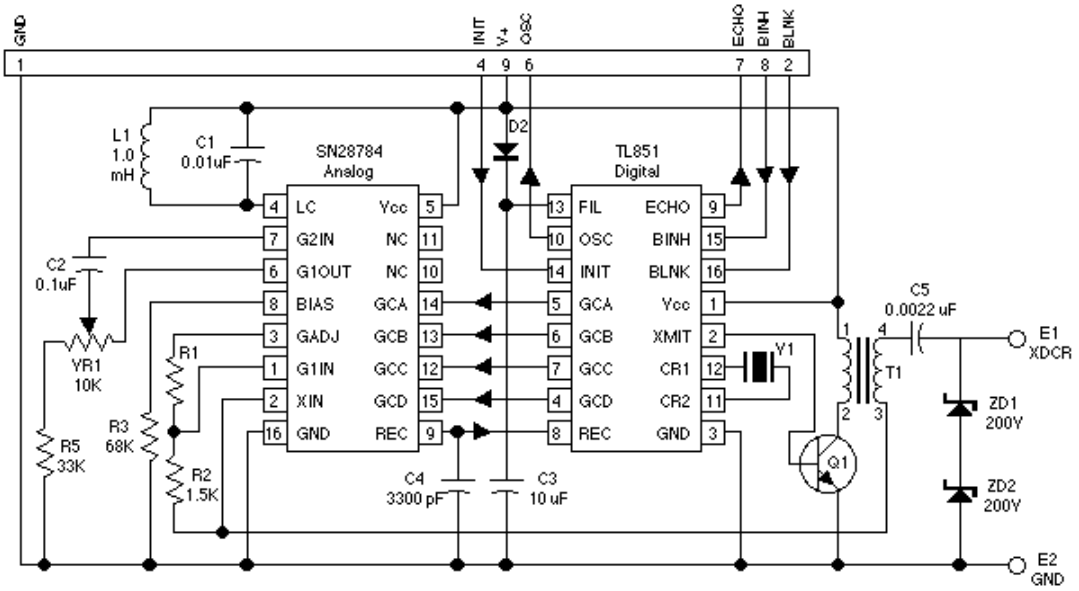

Figure 23: Polaroid US6500 control board

The control sequence to perform a single echo measurement is explained below on the basis of the following figure:

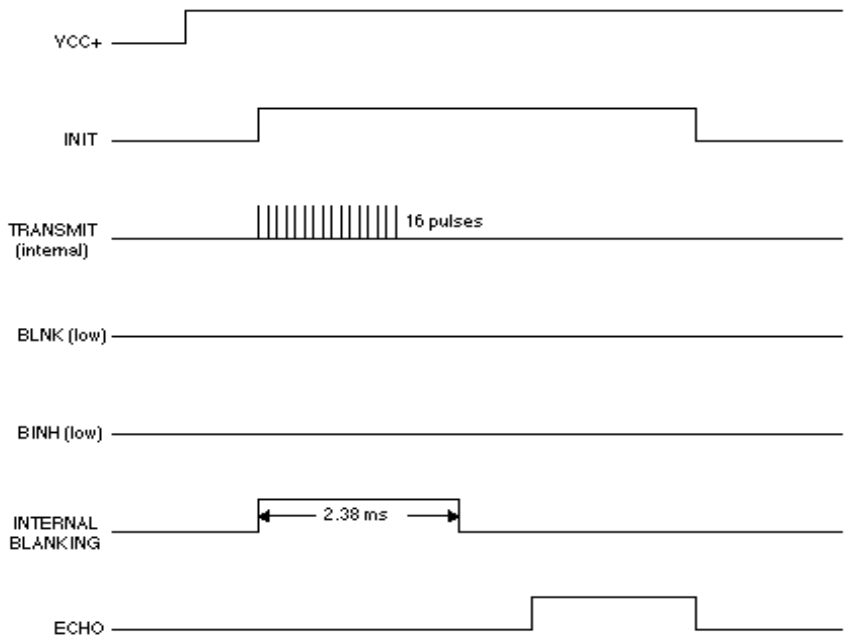

Figure 24: US 6500 Control sequence in single-echo mode

After applying power (V_{CC}) a minimum of 5 milliseconds must elapse before the INIT input can be taken high. During this time, all internal circuitry is reset and the internal oscillator stabilizes. When INIT is taken high, drive to the transducer output (XDCR) occurs. Sixteen pulses at 49.4 kilohertz with 400-volt amplitude will excite the transducer as transmission occurs. In order to eliminate ringing of the transducer from being detected as a return signal, the receive (REC) input of the ranging control IC is inhibited by internal blanking for 2.38 milliseconds after the initiate signal. In the single-echo mode of operation, all that must be done next is to wait for the return of the transmitted signal. The returning signal is amplified and appears as a high-logic-level echo output. The time between INIT going high and the Echo (ECHO) output going high is proportional to the distance of the target from the transducer. If desired, the cycle can now be repeated by returning INIT to a low logic level and then taking it high when the next transmission is desired, but it a certain delay between measurements has to be respected, because otherwise secondary echoes of the former transmission will be detected. This delay time needed can be estimated easily by calculating the time for the ultrasonic pulse train sent out to be attenuated enough:

$$t = \frac{2.d_{\max}}{c} = 63ms$$

With d_{max} the maximum detectable distance of 10.7m according to the constructor. To stay on the safe side, an actual delay time of 50ms was implemented.

The BLNK signal can be manipulated if needed to make the detection of multiple echoes possible. The BINH signal is used to shorten the internal blanking time interval, enabling the sensors to detect objects at a closer distance than the 40cm corresponding to a detection delay of 2.38ms, at a cost of an increased unreliability of the sensor. These extra capabilities were not used in order not to complicate things, as the sensors already posed reliability problems in single echo mode.

Realisation

The ultrasonic sensors were placed on the front of the robot to achieve a maximum visibility range. Since the sensors have a very limited field of view and only two sensors are at our disposal, this installation location is critical as it determines the image of the environment the robot will be able to percept.

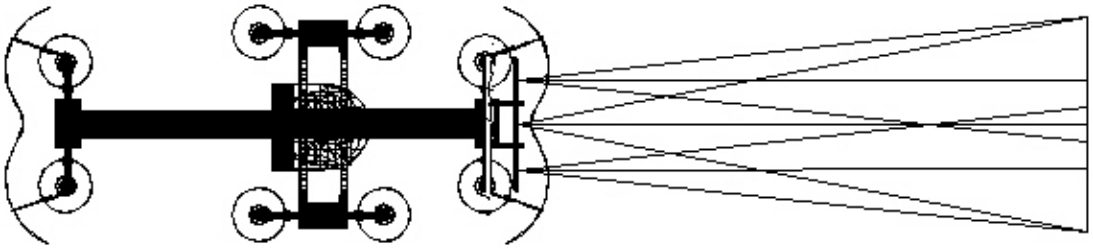

Figure 25: Angular range of the ultrasonic sensors (to scale)

To reach a maximum precision during triangulation of sensor readings, both sensors must be placed at a reasonable distance to each other. On the other hand, increasing this mutual distance leads to other fusion problems, as more often a situation will arise where the two sensors are measuring different objects and a correct fusion is impossible. Concrete, the sensors were installed at mutual distance of 16cm and at a height of 10 cm above the ground level. This height of installation was also thought about, since the sensors were so sensitive they detected the joints on the floor. This behaviour could not be tackled by altering the height of the sensors however, so the joints had to be taped.
The ultrasonic sensors require a considerable supply current of 2A, so the original idea of using the electrical supply of the computer through the acquisition card had to be abandoned and a separate supply was set up.
The actual time measurement is an operation that is not performed by the sensor itself. In order to reach a maximum resolution, the general-purpose counter / timer of the digital acquisition board at a frequency of 20MHz was used instead of the internal clock of the PC.

Programming issues

The general program flow of the module controlling the ultrasonic sensors is presented below:

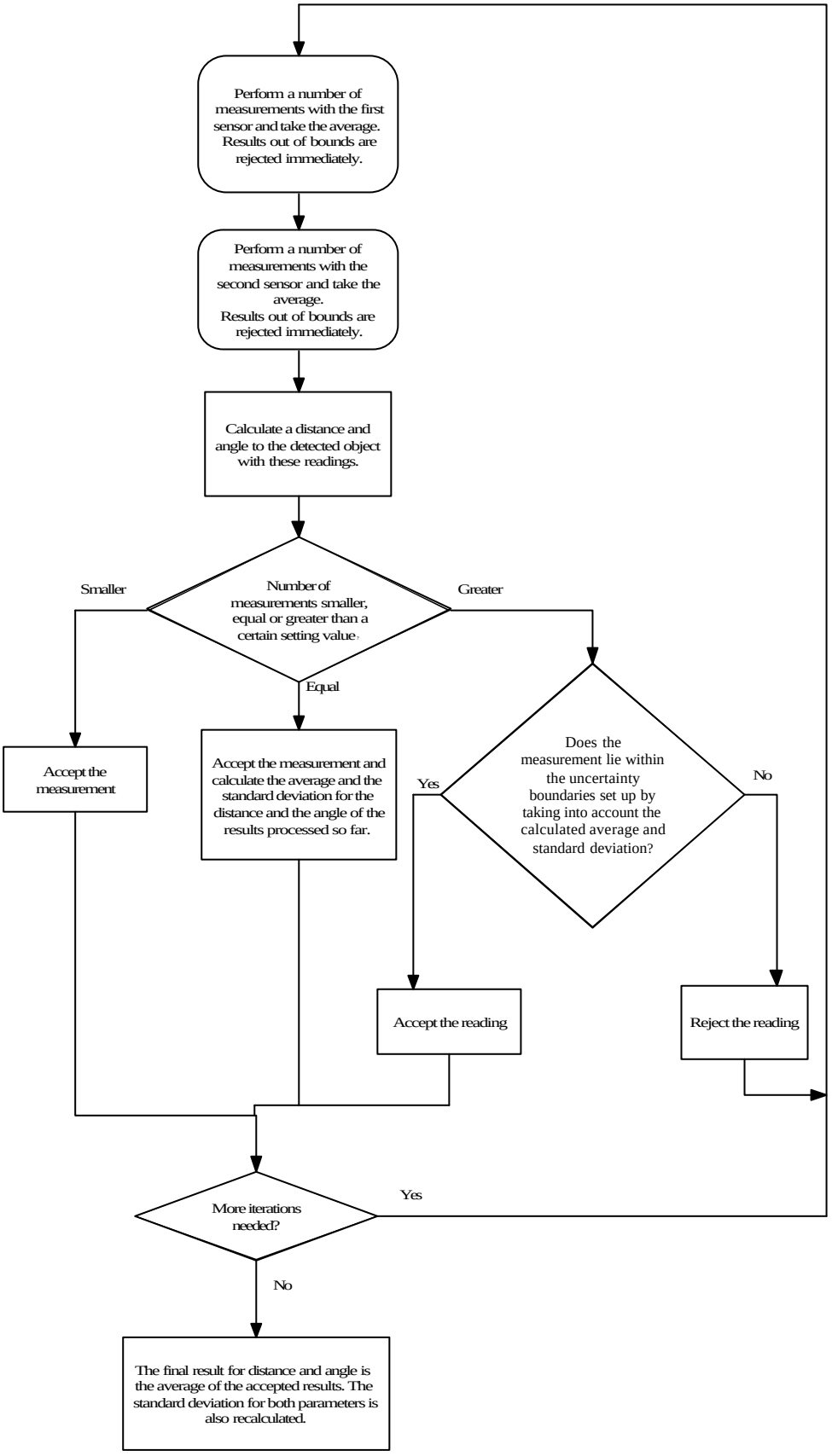

Figure 26: Flow chart of Ultrasonic distance and angle measurement

- First, the right sensor will execute a series of measurements. These readings are only accepted if they fall within the range from 40cm to 10m. To finish this first stage, the mean of these measurements is calculated.
- The same procedure is done for the left sensor.
- Using these averages, the distance and angle to the target is calculated using the triangulation method.
- These three steps are iterated a number of times, after which a mean distance and angle is calculated, and - even more important - the standard deviation on these values is computed.
- The program continues iterating the first four steps, with this difference that new values for distance and angle are henceforth only accepted if they fall in the $[x - 3 . \sigma , x + 3 . \sigma]$ interval of the parameter considered.
- Finally, all results are averaged and the standard deviation of the whole set of values is calculated.

The extensive statistical processing used may seem a little overkill, yet this approach will show its results when the reliability of the measurement is regarded. To increase the speed of the ultrasonic measurements, the amount of measurement points was brought down from the testing stage to the actual robot operation. This does not have much effect on the distance and angle readings, but it has as a side effect that the calculated standard deviations are more unreliable.
If the two sensors are measuring different objects, which is a very common situation, the triangulation procedure will not be able to deduce any angle information about the detected objects. Instead, it will return the distance to the closest object and an angle of plus or minus four degrees depending on which sensor detected this object. This arbitrary value of ±4° was set up after some experiments in order to minimize the maximum error.

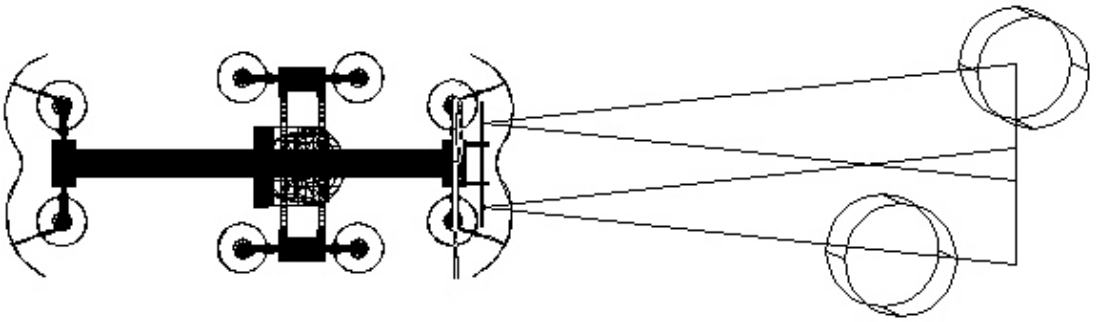

Figure 27: Sensors measuring different objects

Experimental results

Procedure:

- Distances are measured from the middle point of the line connecting the two sensors till the foremost point of the measured ball.
- Angles are considered negative to the right of the longitudinal robot axis and positive to the left

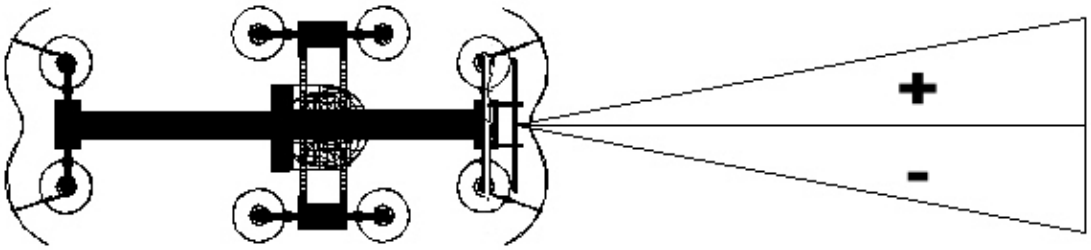

Figure 28: Robot angle convention

- The distance measurements were calibrated at a distance of 3 meters. This calibration consists of small adaptations being made to the speed of sound to acclimatize the robot to the ambient temperature and to incorporate the reflection characteristics of the detected object.
- The object to be detected, the basketball, is put on a series of measurement points. An error of about 1cm can be expected on this positioning.
- The processing of the readings takes place according to the statistical method explained above. This means that one measurement consists in reality to a multitude of physical readings.

Distance measurement:

- A first important and unexpected observation is that measurements under 3 meters always lead to overestimations of the measured distance, whereas readings above 3 meters tend to produce an underestimation, which is shown by the first data series (in brown) on the next chart.

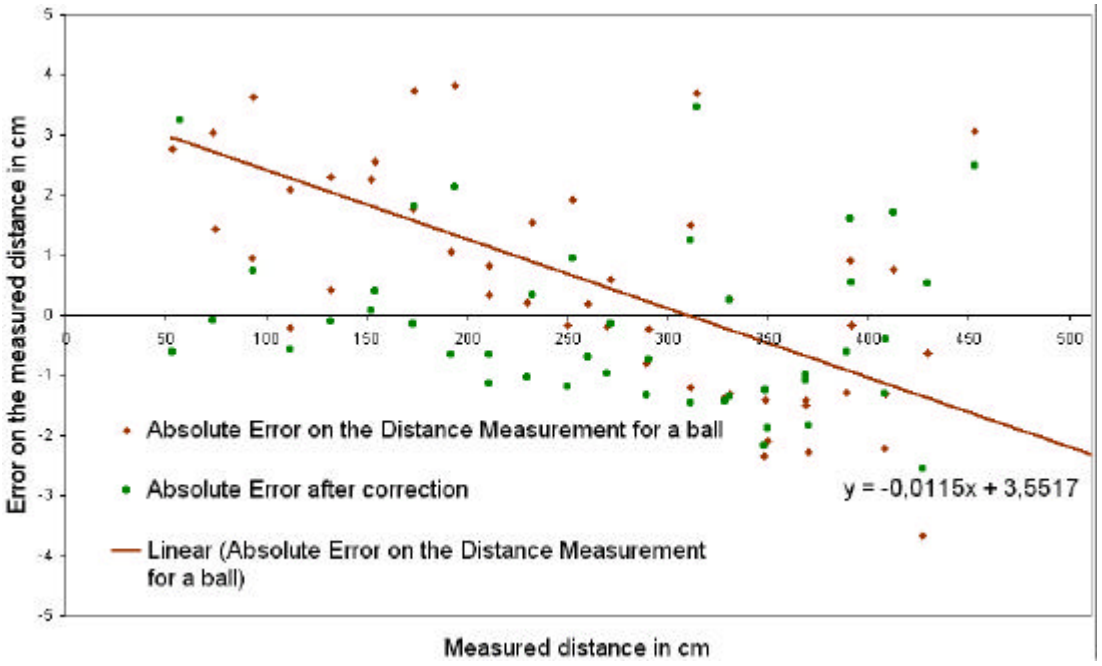

Figure 29: Distance overestimation and underestimation and the correction

An explanation for this erroneous behaviour lies in an overestimation made in the time measurement. Such an overestimation will have a relatively smaller effect for readings at a greater distance, but will have more and more effect at shorter ranges where the overestimation will become important. Because the Nidaq general purpose timer / counter isn't able to simply count from the rising edge of one channel to the rising edge of another - which is needed for the distance reading, these channels being respectively the INIT and the ECHO signal – a small programming artifice was used to get round this difficulty. This produces a small overestimation; yet testing showed that the amount of this overestimation could not explain the observed error. Thus, it must be concluded that there is a small time delay of the order of 0.1ms present in the ultrasonic sensors ranging module itself. However, regarding the very systematic nature of this overestimation, this really does not pose a problem since it can easily be countered by implementing a linear correction, which produces the green data points on the above chart.

After correction, most of the errors are located in the [-2cm , +2cm] interval, which is very acceptable for our purposes.

- A second observation made is that large errors are in most cases overestimations, which is quite logic since these are the situations where the directly returned wave was missed and a later echo was measured.
- In order to correctly fuse sensor data in the decision step, we need a means to track down erroneous measurements. Therefore, the following graph presenting the error on the distance measurement as a function of the standard deviation is drawn.

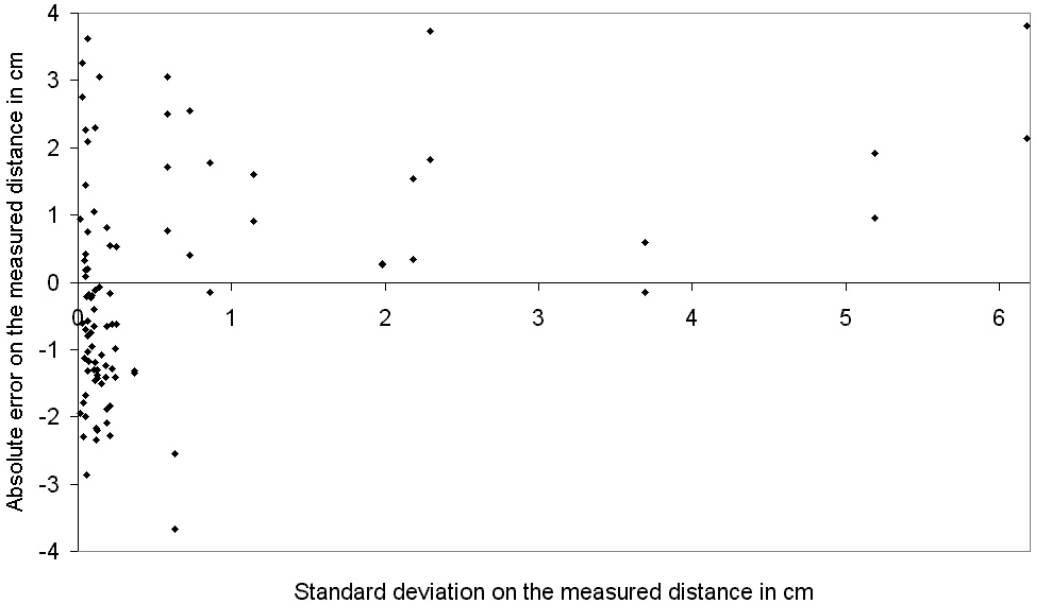

Figure 30: Error on the distance measurement as a function of the standard deviation

A conclusion that can be made is that if the standard deviation grows large, presumably an overestimation of the distance to the detected object is made.

- The constructor has indicated a range of 10 m for the specific sensors after which the acoustic wave would be too attenuated to be detected. This seems to be an optimistic vision as the basketball is not detected anymore at a distance of more than 4.5 m.

Angle measurement:

- Ultrasonic sensors are notorious for their lack of spatial resolution and indeed, the angle measurement isn't very accurate. The reason is that this angle must be calculated by the triangulation procedure making use of unstable intersections of arcs as explained above.

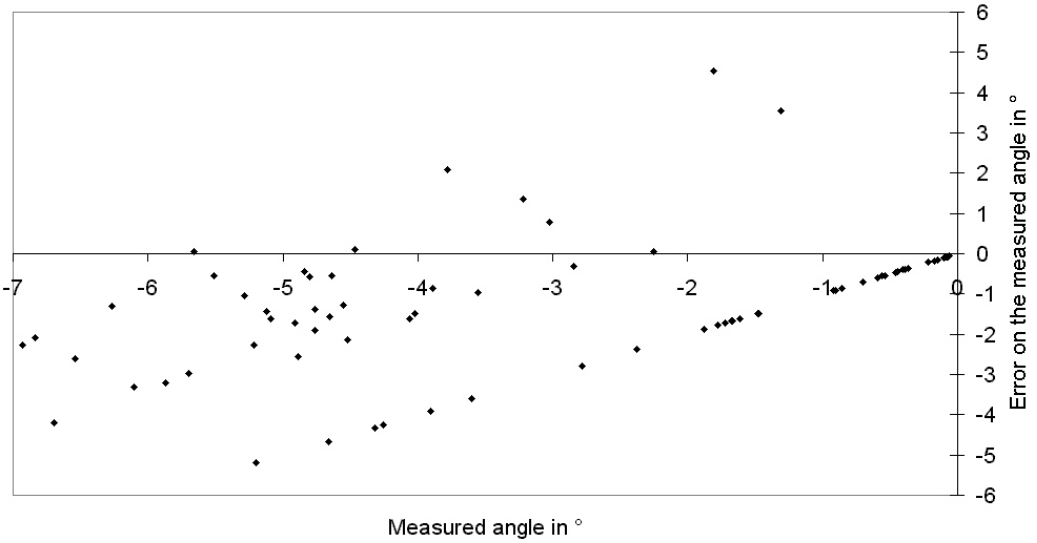

Figure 31: Error on the angle measurement as a function of the real angle

- Though a mean error of about 0° can be observed, the readings are very inaccurate and it can be noted that greater angle measurements are attended by greater errors. Now, the important errors are mostly underestimations, which can be explained by stating that only measurements at negative angles were performed as prior experiments showed a symmetrical relationship and double work could be avoided by doing so. When the object to be detected is located at a considerable negative angle, the left sensor will have difficulties in detecting this object and will most likely return an overestimation for the distance, leading to an overrated negative angle measurement.
- Any angle measurement above 7° in absolute values must be mistrusted, as this is not really possible taking into account the opening angle of the two sensors and the distance they are apart.
- Another rule for determining the reliability of the angle reading can be derived by looking at the following chart plotting the error on the angle measurement as a function of the calculated standard deviation on the angle measurement.

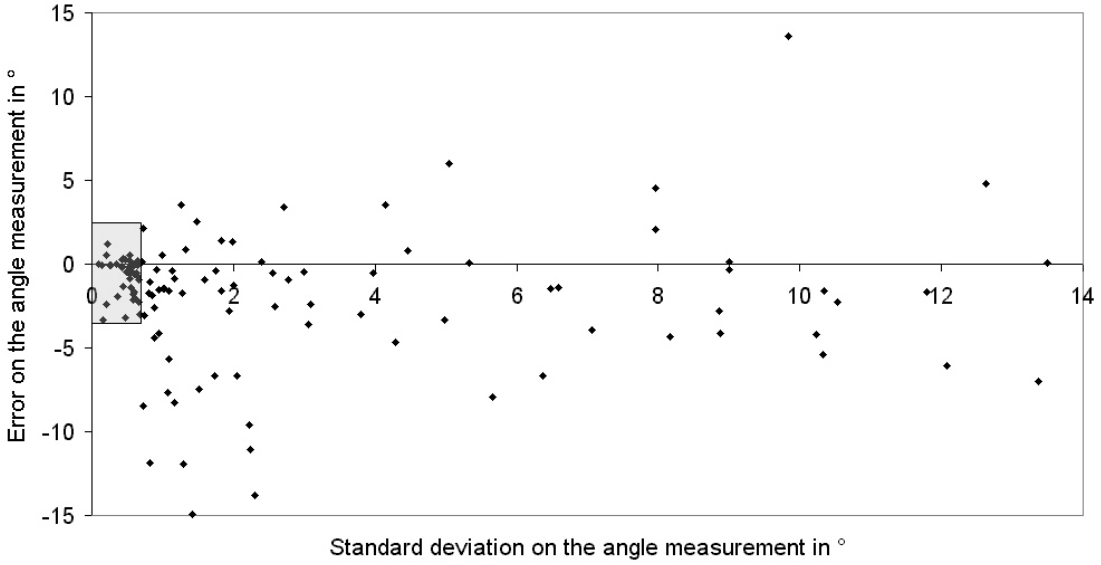

Figure 32: Error on the angle measurement as a function of the standard deviation

The graph seems useless, but it can be noted that only angle measurements with a small standard deviation below 0.7 can be regarded as quite reliable, as is accentuated by the shaded rectangle. These sorts of conclusions are very important for determining the behaviour of the sensor fusion procedure and more in particular for the construction of the membership functions of the fuzzy logic based controller.

Mathematical sensors

In theory

When the robot knew the position of an object relative to itself at a certain stage and it knows what moves it performed since then, it is possible to calculate the new relative position where this object should be detected. This operation is performed for a detected obstacle as well as for the target. These prediction – sensors have therefore no physical basis but are purely mathematical sensors. This can be compared to the principle of dead reckoning used in robot path planning by which mobile robots can be directed to a certain target without the use of any (physical) sensor, just by very accurately recording the consecutive movements and therefore also the position of the robot. This approach isn't new, nor limited to robotics, as many sailors who didn't want to use a compass or sextant used it in ancient days to find their way at sea, simply by estimating the ships movement speed and recording the changes of the rudder position. The error made by using this technique depends on the precision by which the robot movement is known and of course the error on the initial position information. In general, an elliptical region of uncertainty is assumed when using dead reckoning as a positioning tool. In order not to complicate things, a simple circular region of uncertainty is assumed by the robot.

In practice

Predicting the new position of an object on the basis of a former location and the knowledge of the step made in the mean time, is in fact a very simple calculation, since there are only 4 different moves possible:

- *Robot turns right*

  Calculation:
  The robot rotation angle (16°) is added to the angle to the object.

  The same angle definition as with the ultrasonic sensors is used.

  $d_{new} = d_{old}$

  $\alpha_{new} = \alpha_{old} + \sigma$

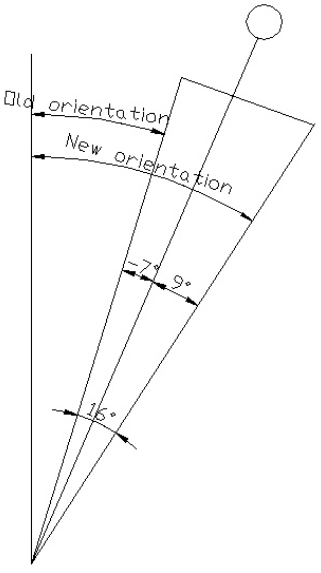

Figure 33: Robot turning left

- *Robot turns left*

  Calculation:

  The robot rotation angle (16°) is subtracted from the angle to the object.

  $d_{new} = d_{old}$

  $\alpha_{new} = \alpha_{old} - \sigma$

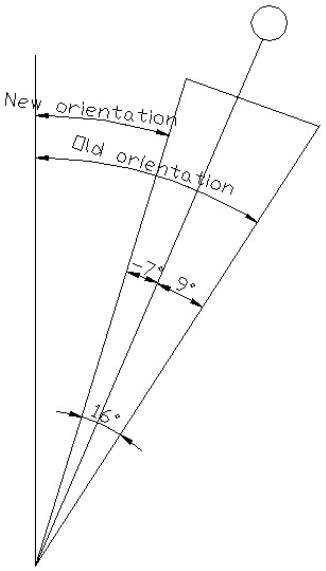

Figure 34: Robot turning left

- *Robot moves forward*

  Calculation:

  The cosine rule is used to calculate the new distance and angle.

$$d_{new} = \sqrt{d_{old}{}^2 + stepsize^2 - 2.d_{old}.stepsize.\cos\left(a_{old}\right)}$$

$$a_{new} = p - \text{acos}\left(\frac{d_{new}{}^2 + stepsize^2 - d_{old}{}^2}{2.d_{new}.stepsize}\right)$$

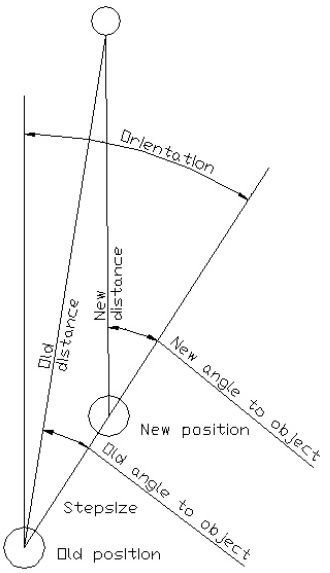

Figure 35: Robot moving forward

- *Robot moves backward*

  Calculation:

  The cosine rule is used to calculate the new distance and angle.

$$d_{new} = \sqrt{d_{old}{}^2 + stepsize^2 - 2.d_{old}.stepsize.\cos\left(p - a_{old}\right)}$$

$$a_{new} = \text{acos}\left(\frac{d_{new}{}^2 + stepsize^2 - d_{old}{}^2}{2.d_{new}.stepsize}\right)$$

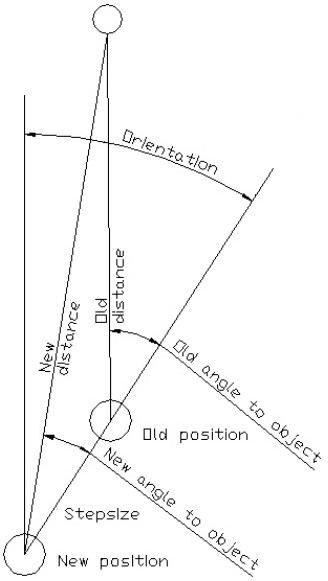

Figure 36: Robot moving backward

The error or standard deviation on this sensor needs to be set arbitrarily. At the beginning of the testing, the dead reckoning errors were considerable due to the slipping of the robot caused by the impulsive movements. By stifling the valves controlling the turning movements, this behaviour could be corrected, so an overall value of 2cm for the standard deviation on the mathematical sensors could be used.

Conclusions

The presented sensor equipment the robot has at its disposal is very limited and by no means sufficient for gaining a complete image of the environment. Looking only at the angle of view of the ultrasonic sensors, it is clear that the robot will perceive only a narrow section of the surroundings after every step, so the following data processing modules will have a defiant task to fulfil to make the robot able to navigate with such incomplete information.

It is important for the decision fusion to succeed that the used abstract sensors are independent. Indeed, the fusion process may be able to detect that one sensor returned erroneous data, but it will not succeed in detecting such errors if multiple sensors report them. Therefore, it is important to compare the various sources of errors for the different sensors.

For the camera, we know the readings are influenced by the lighting conditions and the calibration. The ultrasonic sensors are subject to a number of external influences such as temperature, calibration, nature of the reflecting object and acoustic climate.
Some common failure situations for both sensors are summarized in the following table:

Situation	*Camera*	*Ultrasonic sensor*
Close objects (<0.5m)	*OK / Overestimation*	*Fails*
Far objects (>5m)	*Fails*	*Fails*
Object at small angle	*OK*	*OK*
Object at large angle	*OK (< 100°)*	*Fails*
No external light source	*Fails*	*OK*
Oblique objects	*OK*	*Fails*
High obstacles	*Fails*	*OK*
Red environment	*Fails*	*OK*

Table 1 : Sensor failure situations

Luckily, there does not seem to exist a certain parameter affecting both sensors within the regarded working range, but one has to keep in mind that both sensors will fail at greater distances
This demand for independence will pose problems however, if we would use one sensor to calibrate the other as was proposed as one of the advantages of using multiple sensors. In order not to complicate things, these techniques were therefore not used.

Chapter 5: Sensor fusion

Introduction / need

Sensor fusion can be defined as any process where there is an actual combination or fusion of different sets of sensory data into one representational format [6]. As already mentioned, this chapter only deals with the final step of sensor fusion, the so-called decision step, where the readings of the different abstract sensors are combined, yet this is also the most complicated and interesting stage of the fusion process.

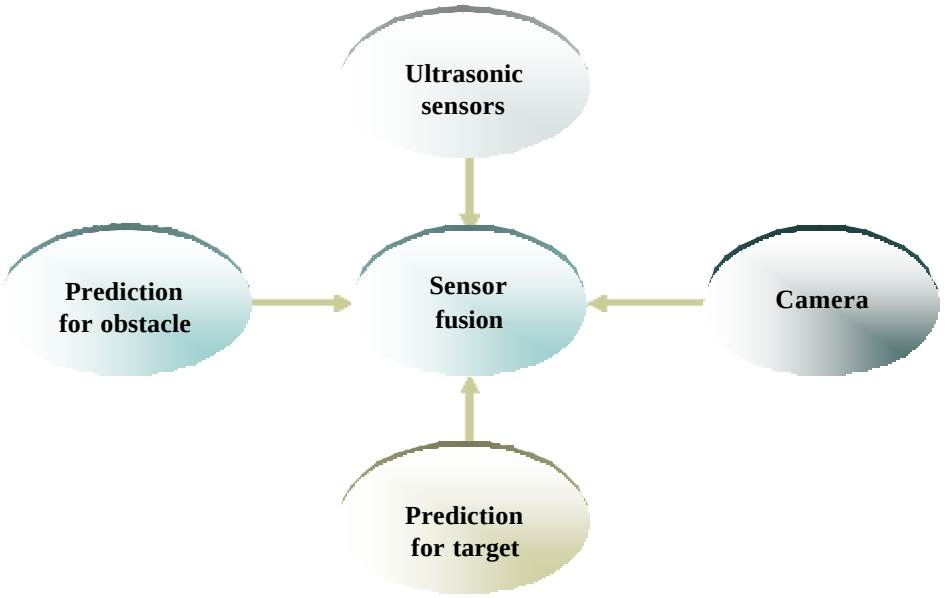

Figure 37: Decision fusion

The image human beings make themselves of their surroundings is constructed by the intelligent fusion of information from all their sense organs. A clear example is the process of eating where not only the sense of taste, but also the senses of smell, vision and touch are involved. It may be stated that one of the main reasons for the fact that the current generation of robots still falls short of what humans are capable of, is the lack of sensors and the possibility to process the massive complex information stream these sensors provide in an intelligent manner. A robot cannot be made clever just by adding sensors; it is only by the manner of processing of the presented data that a robot can be given a certain amount of "intelligence".

This makes clear that the sensor fusion module plays a crucial role for the working of the whole system. Nevertheless, it is not common practice to provide a separate fusion module for dealing with this task, as in many applications sensor data is fused during the map building process itself. This approach was not followed to preserve the reusability of the different components and because it was the idea to implement something new.

In theory

Problem statement

A common difficulty in implementing multi sensor data fusion is to find a suited data structure to combine the often incompatible sensor readings. This problem has been evaded elegantly by the introduction of the abstract sensors and the medium level fusion processes. All these abstract sensors return the same type of data, namely a distance, an angle and the respective standard deviations on these parameters.
The problem of sensor fusion can be subdivided into a number of categories, depending on the contents of the returned information. Regarding this subject, an important distinction is to be made between complementary, competitive and cooperative sensors, which are discussed below:

- Complementary sensors are completely independent from each other, but they can be combined to give a more complete image of the environment. A typical example would be four radar stations placed into the following configuration:

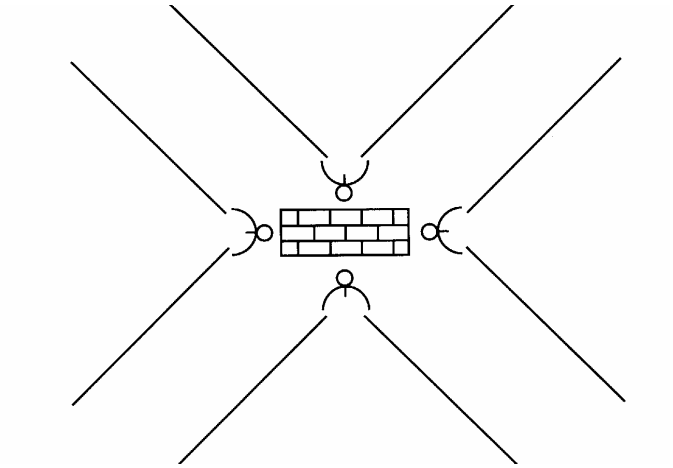

Figure 38: Radar stations working as complementary sensors

The different radar stations all measure the same type of information - although this is not a requirement - but they are each covering a different region. Merging their data leads to an equivalent radar station covering the entire region. The fusing process for complementary sensors is not difficult, as their data merely needs to be appended and there is no real interaction between this information.

- Competitive sensors provide independent measurements of the same information, so the returned information should be equivalent. Since they provide what should be identical data, the sensors are in competition as to which reading will be believed by the system in the case of discrepancies.

A competitive network of radar stations would for example have the following configuration:

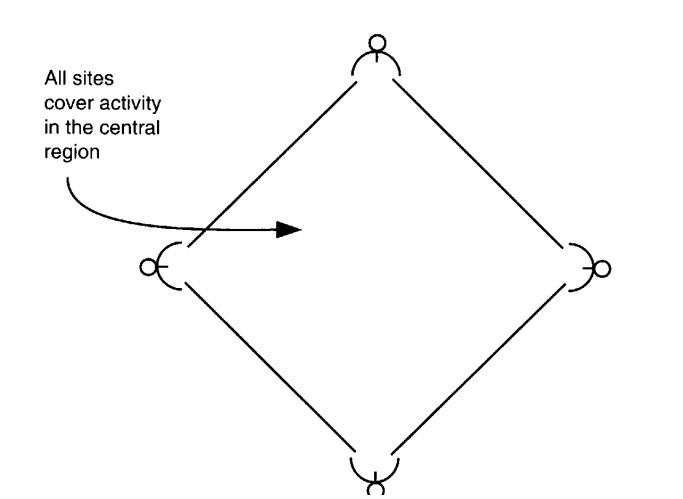

Figure 39: Radar stations working as competitive sensors

This technique of redundant sensors is often used in mission critical components, since the reliability can be increased and, in principle, a single sensor failure can be coped with. Data fusion of competitive sensors is not that simple because often, conflicting sensor information needs to be interpreted

- Cooperative sensor networks combine data from independent sensors to derive information that would be unavailable from the individual sensors. A typical example of a cooperative sensor is a stereo camera: none of the both cameras used can extract depth information, yet by combining the two images, 3D information can be retrieved out of the 2D sources. Data fusion of cooperative sensors is in general a difficult operation and highly system specific and will therefore not be handled any further.

A key aspect of the sensor fusion problem in the presented application is that the used sensors are sometimes working as complementary sensors and other times as competitive sensors. If the ultrasonic sensors are for example measuring an obstacle, the readings from these sensors and these from the camera are complementary. Yet, if the ultrasonic sensors measure the target object, their returned measurement is in competition with the camera reading. The same remarks are valid for the prediction sensors, as it must for example be detected whether the robot is still measuring the same obstacle so the prediction made is valid, or whether a new obstacle is being detected so the prediction is useless. This unclear and changing working mode of the different sensors poses an extra difficulty for the sensor fusion process and as a result different fusion techniques had to be used together to come to satisfactory results.

The use of competitive sensors implicates that one has to find a means to fuse inconsistent sensory data, since there is always a good possibility that one of the sensors is returning erroneous information. The general purpose of applying a redundant system is to preserve the system functions

in the case where one of the components fails. As stated earlier, this capability to meet with defective sensors isn't really a reasonable goal, since only very few sensors were used. Yet, redundancy is used here to improve measurements and to detect an incorrect measurement of a sensor working correctly.
To introduce the problems of sensor fusion, a simple time measurement is discussed. A philosopher once stated that is better to have only one clock than two, for an observer with only one watch doesn't have to doubt his time measurement, whereas someone with two watches can never be sure of the exact time:

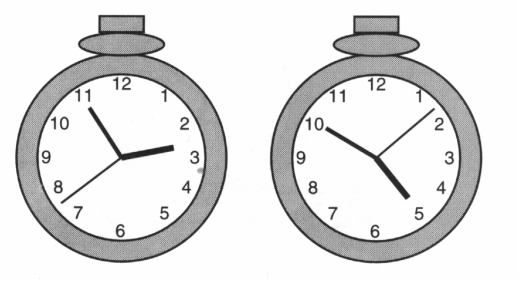

Figure 40: Time measurement using two clocks

A more fortunate observer using three clocks can be relatively certain of the correct time, even if one watch fails:

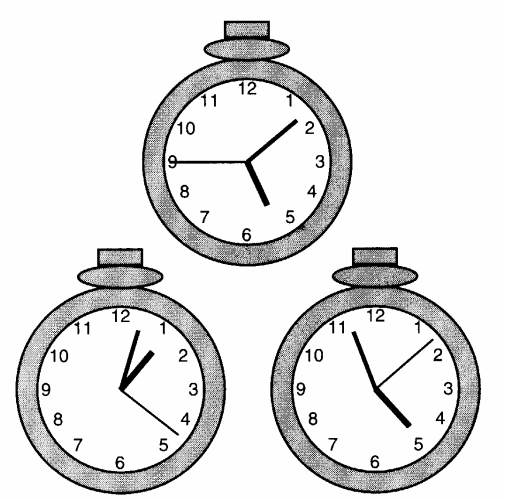

Figure 41: Time measurement using three clocks

The situation on the above figure makes also clear that a naive approach to sensor fusion that could be proposed, namely taking a weighted average of the sensor readings, may lead to totally incorrect results. In order to come to unambiguous results, the uncertainty on the measurement needs to be taken into account in some way. The uncertainty on a measurement is an essential factor in the data fusion process and this is why such an effort was made to represent this uncertainty on the readings from the different abstract sensors using the standard deviation. Indeed, fusing data from perfect sensors isn't difficult, it is the uncertainty that makes the process more complicated and also more interesting.

Fusion Tools

To deal with uncertainty, numerous techniques stand at our disposal. Because of the complicated nature of the sensor fusion problem for the robot, different tools will need to be used, thus realizing a hybrid fusion process. Some of the common fusion techniques are described in the following sections.

Explicit accuracy bounds

The assumption made by this technique is that a sensor does not return a single exact value but a certain range that contains the correct value. The uncertainty area can be estimated by making use of statistical methods as is done by the abstract ultrasonic sensor for example. In the great advantage of this technique is that it is a deterministic approach. On the other hand, this can also be seen as a disadvantage, as the explicit well-defined accuracy bounds do not correspond to the physical reality and are unable to represent the real probabilistic error distribution.

A practical approach for implementing this method is to make use of so-called d-rectangles and d-circles. This more-dimensional technique comes in the presented one-dimensional case to find the line segment where the uncertainty areas of at least two sensors overlap. This situation with seven sensor readings is sketched on the following figure:

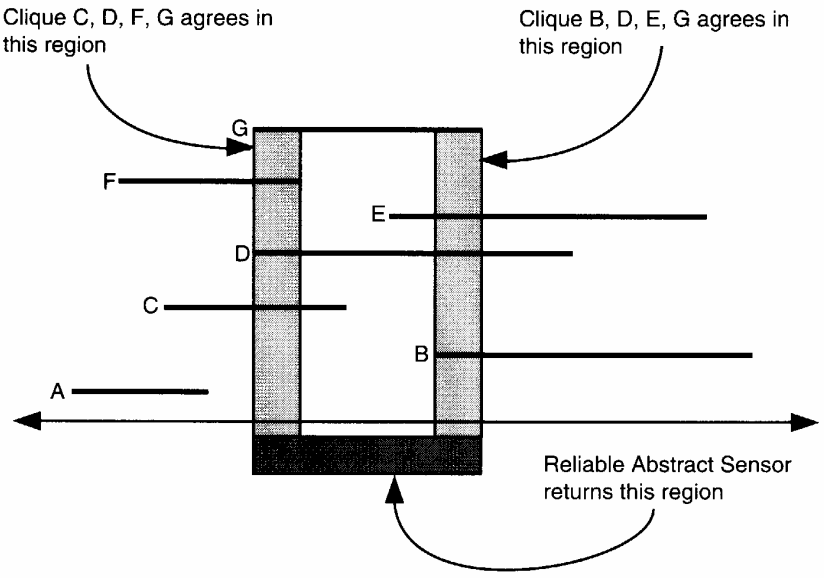

Figure 42: Fusion of 7 1-dimensional readings

As every measurement is surrounded by an uncertainty area, the reading can be represented as a finite line segment along the axis with possible measurements. By superposing these line segments, the segments can be found where, in general, more than $\frac{N}{2.D}$ segments overlap, N being the number of

sensors and D being the number of dimensions, which is one in the robots' case. The number of line segments needed therefore depends on the number of sensors involved in the reading.
Of course, it is still possible for a sensor to return a complete erroneous measurement, which is a situation that cannot be handled in an efficient way using this method. This technique also does not permit to take into account the specific nature of every different sensor; all readings are handled equally, which is not a very intelligent approach.

Probability and Dempster – Shafer methods

Probability methods for dealing with sensor fusion problems, most often rely on the use of Bayes' rule, which quantifies the probability of a certain event Y, given that event X has already occurred:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

In order to fuse measurements using Bayes' rule, it is necessary to know or estimate the conditional probability distributions $P(X \mid Y)$ for the sensing actions, that is, the probabilities that the properties are observed provided the different hypotheses are true. These distributions can be found by performing a large number of sensing actions and observe how often the different sensing actions identify the features of the different objects. The method also implicates that an a priori probability distribution $P(Y)$ has to be found in some way. As the robot has to deal with at the beginning complete unknown and arbitrary environments, these limitations make the method useless in all but laboratory conditions.
Dempster – Shafer reasoning tries to deal with the problem of the need for a priori environmental knowledge. The sensors now assign probability masses to propositions, i.e. sets of hypotheses, which the particular sensor is unable to distinguish between. For the frame of discernment [5], θ, which contains every single hypothesis, there exists 2^{θ} such propositions. This immediately makes clear the main problem of the Dempster-Shafer technique as 2^{θ} will rapidly grow out of proportion when the number of hypothesises is large, which is the case for the robots' sensor readings.

Statistical methods

In contrast to the probability approach, the possibility theory keeps track of the information presented, thereby providing a measure of the amount of information unavailable to the system. This means that no prior knowledge is necessary, and that the system will improve itself in its capabilities over time. A popular and powerful technique for implementing a statistical method is making use of the Kalman filter. The robot does not use this method for fusing the sensor information in the decision step, yet the Kalman filter is used by the target-tracking algorithm to improve the tracking capabilities.

Fuzzy logic

Fuzzy logic is a quite new technology, which fuzzy nature makes it an excellent tool to deal with the uncertainty in the sensor fusion process. Fuzzy logic is a multi-valued logic capable of simulating a "human" kind of thinking through the use of a set of if-then rules, implemented by the programmer on a basis of experience data. The relation to the human mind can be made as the fuzzy logic controller does not make use of raw figures as other systems do, but uses fuzzy words to depict a fuzzy variable such as small, medium, high,… An interesting possibility in this context is that it is perfectly legal for a fuzzy variable to be part of multiple fuzzy subsets. This capability is produced by the introduction of non-deterministic membership functions, or functions expressing to which extent fuzzy variables belong to a certain fuzzy subset.

Used Tools

It has always been an objective during the implementation of this final term project to reproduce as much as possible a sort of human kind of thinking for the robot. As the human mind may be modelled as a fuzzy neural network, it is clear that in this case, fuzzy logic presents the most interesting framework for achieving an intelligent sensor fusion. Nevertheless, a sort of hybrid structure was implemented in practice, incorporating next to the backbone fuzzy logic controller also the explicit accuracy bounds method. This technique was used in order to come to a general fusion controller using the best of two worlds to achieve better results. The explicit accuracy bounds method will merely be used in order to discriminate between complementary and competitive working mode of the abstract sensors and a fuzzy logic method to actually fuse the data of the competitive sensors. Why no pure fuzzy logic controller is used to handle the whole fusion task, is explained later in the text when the problems this approach would lead to can be shown more clearly.

In practice

Defining the sensor fusion algorithm

To shape the form of the sensor fusion procedure, it is very important to correctly define its task at first. The sensor fusion module takes as input the data provided by the four abstract sensors and must deliver ready for use information for the map-building module, this data being a clear positioning for the target and an eventual obstacle.

The sensor fusion procedure has as result no less than 16 input variables:

- A distance
- A standard deviation on this distance
- An angle
- A standard deviation on this angle

And this for all of the four abstract sensors.

There are “only” 8 output variables:

- The distance to the target
- The standard deviation on the distance to the target
- The angle to the target
- The standard deviation on the angle to the target
- The distance to an obstacle
- The standard deviation on the distance to an obstacle
- The angle to an obstacle
- The standard deviation on the angle to an obstacle

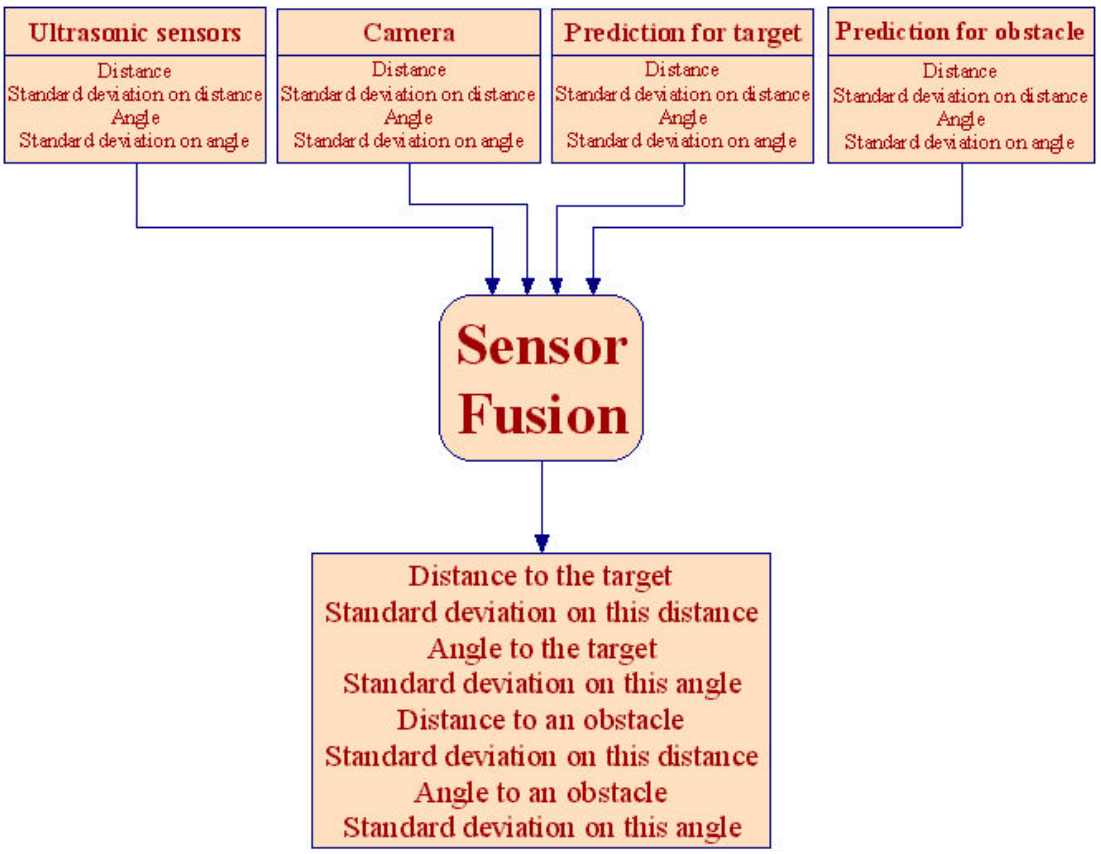

Figure 43: Input and output for the sensor fusion module

According to the general theorems of fuzzy logic, a controller with n input variables and m membership functions would make use of m^n rules. As a minimum of three membership functions is really required to adequately tune the controller, it may be clear that following this approach would lead to an excessive amount of over 40 million rules to be set up. Luckily, fuzzy logic theory is very flexible and this abundance of rules can be avoided by following a different tactics, which will be explained now.

The idea is to calculate every output variable as a weighted average of the input variables; the fuzzy logic based controller determines the weights accorded to these input variables. For the input variables, a number of membership functions are defined:

- For the distance:

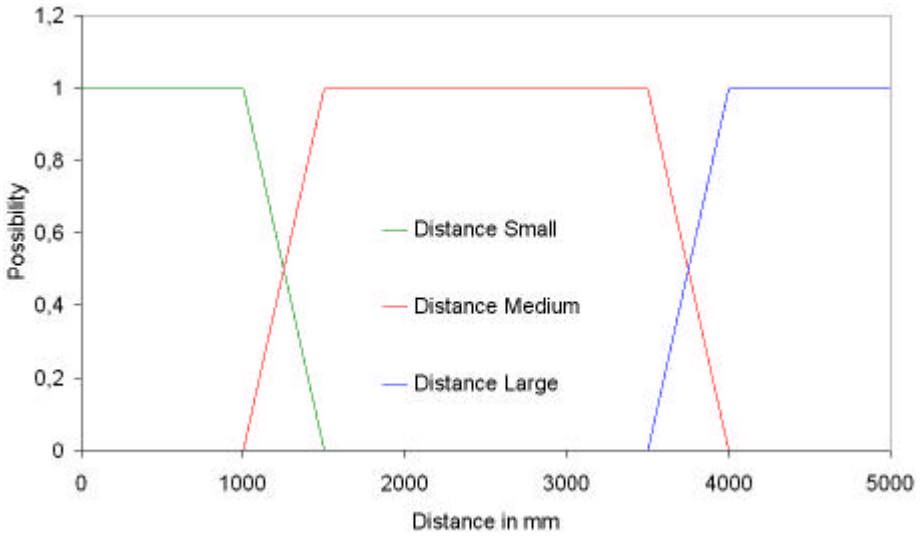

Figure 44: Membership functions for the measured distance

- For the standard deviation on this distance:

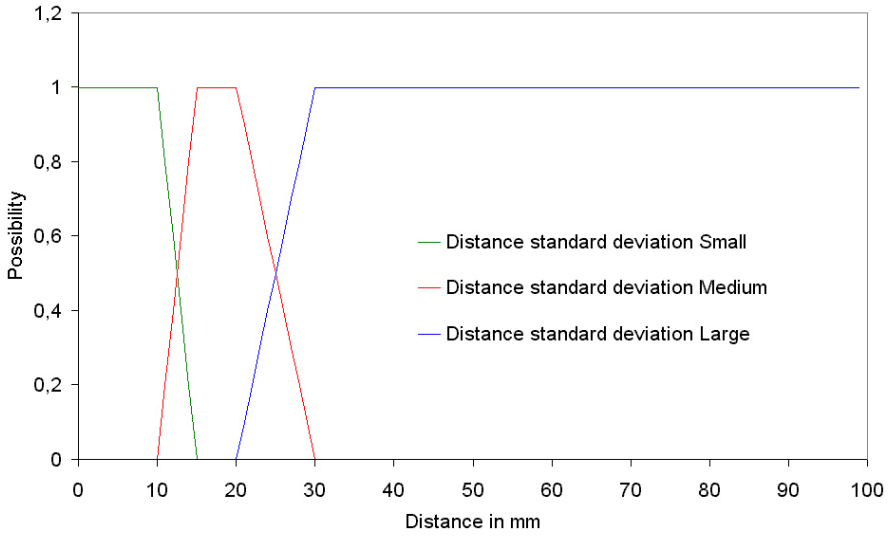

Figure 45: Membership functions for the standard deviation on the measured distance

- For the angle:

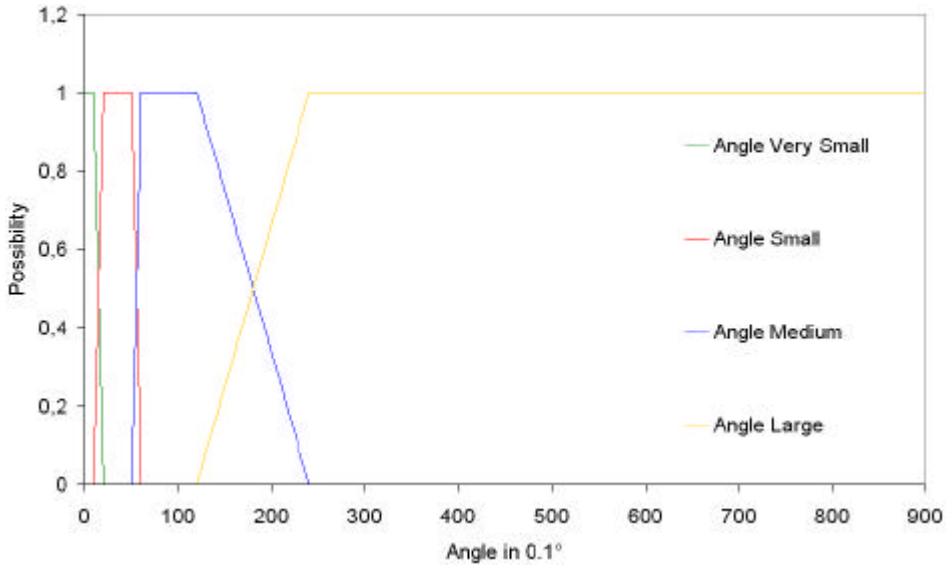

Figure 46: Membership functions for the measured angle

- For the standard deviation on this angle:

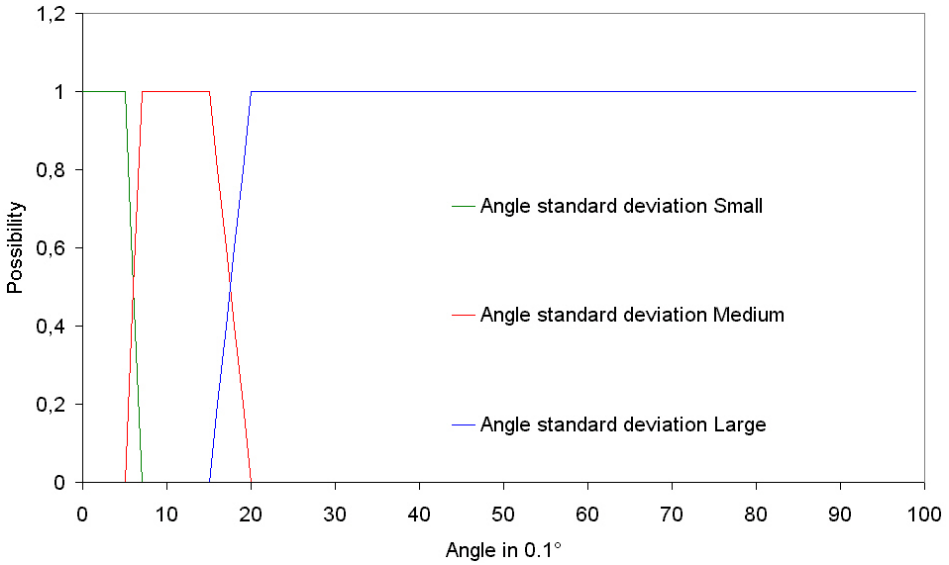

Figure 47: Membership functions for the standard deviation on the measured angle

These membership functions can be used for the inputs of all four abstract sensors, thus reducing the number of functions to be defined drastically. Only for the angle measurement, an extra membership function needs to be foreseen, as the ultrasonic sensors and camera have a very different range for this reading, the ultrasonic sensors being able to detect objects at a maximum angle at about 7°, whereas this maximum angle is 100° for the camera.

The fuzzy logic controller needs to output the weights for the different input variables. The membership functions for the output variables are defined as:

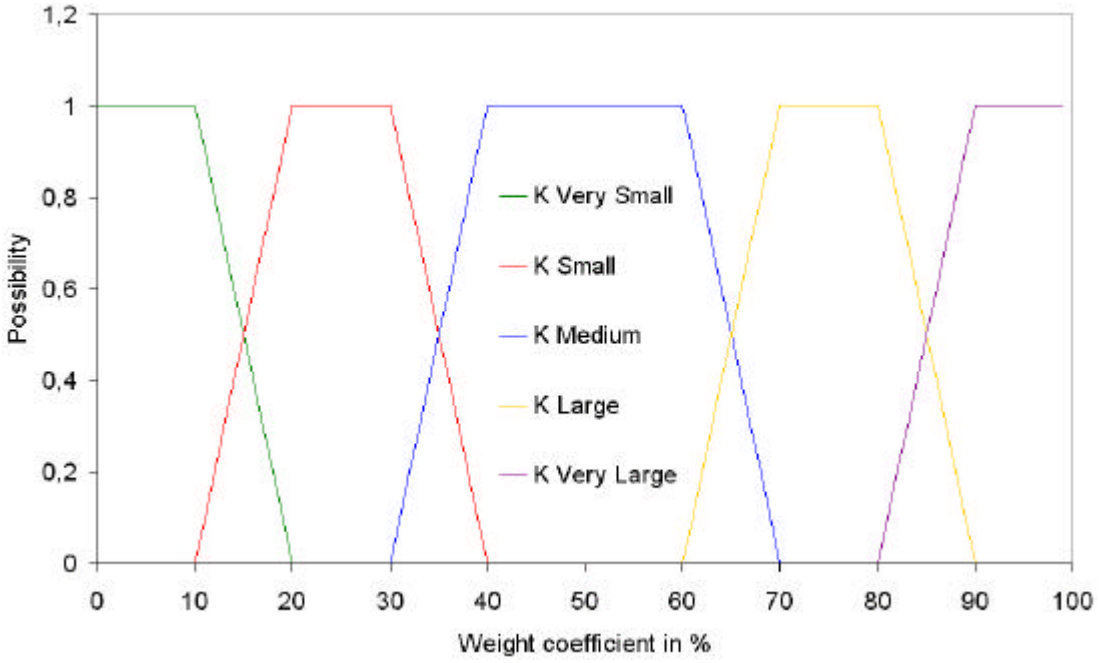

Figure 48: Membership functions for the output variables

A full explanation of the fuzzy logic theory and all the techniques and rules used during the implementation cannot be completed within the scope of this text; as a result, only the used sensor fusion approach to come to an unambiguous distance to the target reading is further explained more in detail. This distance is calculated using the following formula:

$$\text{Distance\_To\_Target} = \frac{K\_US * Distance\_US + K\_CAM * Distance\_CAM + K\_PT * Distance\_PT}{K\_US + K\_CAM + K\_PT}$$

With:

- K_US: Weight coefficient for the Ultrasonic Sensor measurement
- K_CAM: Weight coefficient for the Camera measurement
- K_PT: Weight coefficient for the Prediction of the Target location
- Distance_US: Distance measured by the abstract Ultrasonic Sensor
- Distance_CAM: Distance measured by the Camera
- Distance_PT: Distance measured by the Prediction of the Target location

It is clear that the abstract sensor that was not mentioned, namely the prediction for the position of the obstacle, is not involved in this process of deducing a distance measurement for the target.

The rulebase

These are the rules controlling the distance measurement for the target:

1. *If US_Distance = Distance_Small then K_US = K_Large*
2. *If US_Distance = Distance_Medium then K_US = K_VeryLarge*
3. *If US_Distance = Distance_Large then K_US = K_Large*
4. *If US_Distance_Sigma = Distance_Sigma_Small then K_US = K_VeryLarge*
5. *If US_Distance_Sigma = Distance_Sigma_Medium then K_US = K_Large*
6. *If US_Distance_Sigma = Distance_Sigma_Large then K_US = K_Medium*
7. *If US_Angle = Angle_Large then K_US = K_VerySmall*
8. *If US_Angle = Angle_Medium then K_US = K_Large*
9. *If US_Angle = Angle_Small then K_US = K_VeryLarge*
10. *If US_Angle = Angle_VerySmall then K_US = K_VeryLarge*
11. *If CAM_Distance = Distance_Small then K_CAM = K_Medium*
12. *If CAM_Distance = Distance_Medium then K_CAM = K_Small*
13. *If CAM_Distance = Distance_Large then K_CAM = K_VerySmall*
14. *If PT_Distance = Distance_Small then K_PT = K_Small*
15. *If PT_Distance = Distance_Medium then K_PT = K_Medium*
16. *If PT_Distance = Distance_Large then K_PT = K_Large*

These rules reflect some specific characteristics of the different sensors and some observations derived out of prior experiments, as is explained below for the different rules:

Rules 1 to 3: The ultrasonic sensors deliver an excellent distance measurement with an accuracy slightly dependent on the distance.
Rules 4 to 6: Measurements with a higher standard deviation must be more mistrusted.
Rules 7 to 10: The ultrasonic sensors are less accurate when the object is at a greater angle. If this angle is really large, an error must have occurred as this is physically not possible.
Rules 11 to 13: The camera distance measurement isn't as reliable, certainly at greater distances.
Rules 14 to 16: The prediction for the distance to the target is less reliable than the ultrasonic measurement, but more accurate than the camera measurement.

The whole sensor fusion module has about four times as many rules as it also needs to produce the angle to the target, the distance to an obstacle and an angle to an obstacle. In order not to further complicate things for the sensor fusion process, the standard deviation for a certain parameter is calculated using the same weight coefficients as for the respective parameter.

The benefit of using explicit accuracy bounds

The rules and the theory enunciated above ignore an important aspect of the fusion process as they assume the different abstract sensors are all working as competitive sensors. This is where the explicit

accuracy bounds method comes in. For example, the ultrasonic sensors are capable of producing very accurate distance measurements, but as they are not aware of what they are actually measuring, it is not sure whether this detected object is the target or an obstacle. To deal with this difficulty, it is checked whether the camera measurement and the ultrasonic measurement have overlapping uncertainty areas. If this is not the case, the weight coefficient for the ultrasonic sensors for the target distance reading is automatically set to zero as it must be concluded that the ultrasonic sensors are actually measuring an obstacle. This check must be made not only for the ultrasonic sensors of course; also the prediction abstract sensors are subject to these kinds of operations. To define the explicit accuracy bounds, the $[x - 3 . \sigma, x + 3 . \sigma]$ interval is applied, which is why it was so important to retrieve a notice of the standard deviation on the different readings.

If this functionality were to be implemented using purely fuzzy logic theory, this would have implicated defining extra input variables such as for example (US_Distance – CAM_Distance), plus extra membership functions, thus unnecessary complicating the sensor fusion controller. Another drawback would be that as fuzzy controllers are not well suited to return extreme results such as zero because they perform a certain averaging during the defuzzification process, the weight coefficient would be probably still non-zero.

Implementation

Now the rules and the membership functions are defined, the actual fusion process can take place. The first step is the fuzzification where the input value is blurred and written linguistically, which is done by writing the input value as a fuzzy input set. As the standard deviations on distance and angle measurements are known, these are taken into account and a Gauss function with the appropriate standard deviation is drawn to represent these parameters.

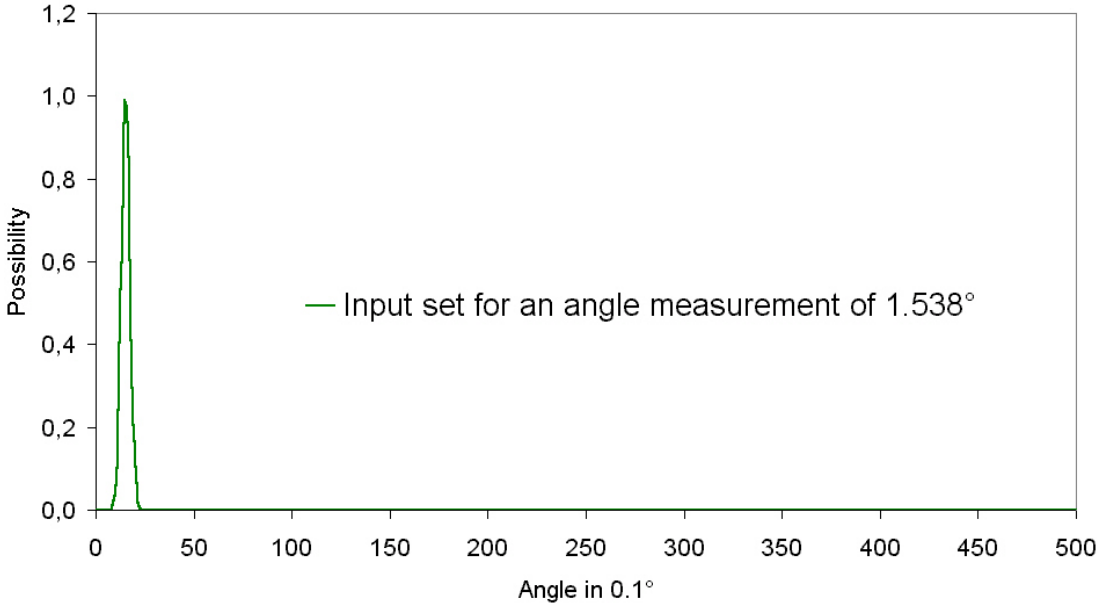

Figure 49: Input set for an ultrasonic angle measurement

Since no higher moments are calculated, this error distribution information is not present for the standard deviation input values themselves, so they are simply represented by a singleton.

Next, it needs to be stated what type of technique was used to implement the fuzzy logic controller. On this subject, the final choice went to a Max-Min aggregation of rules, implicating the following formula to calculate the degree of firing for each rule is used:

$$t_i = Sup_x \left[I(x) \wedge A_i(x) \right]$$

With I(x) being the input set and $A_i(x)$ the membership function needed to be tested for this specific rule i. These and other relations are worked out using the t-norm Min and t-conorm Max.

This number τ_i actually represents to what degree the investigated rule is applicable to the given fuzzy input variable. The result of this validation process is shown on the next figure:

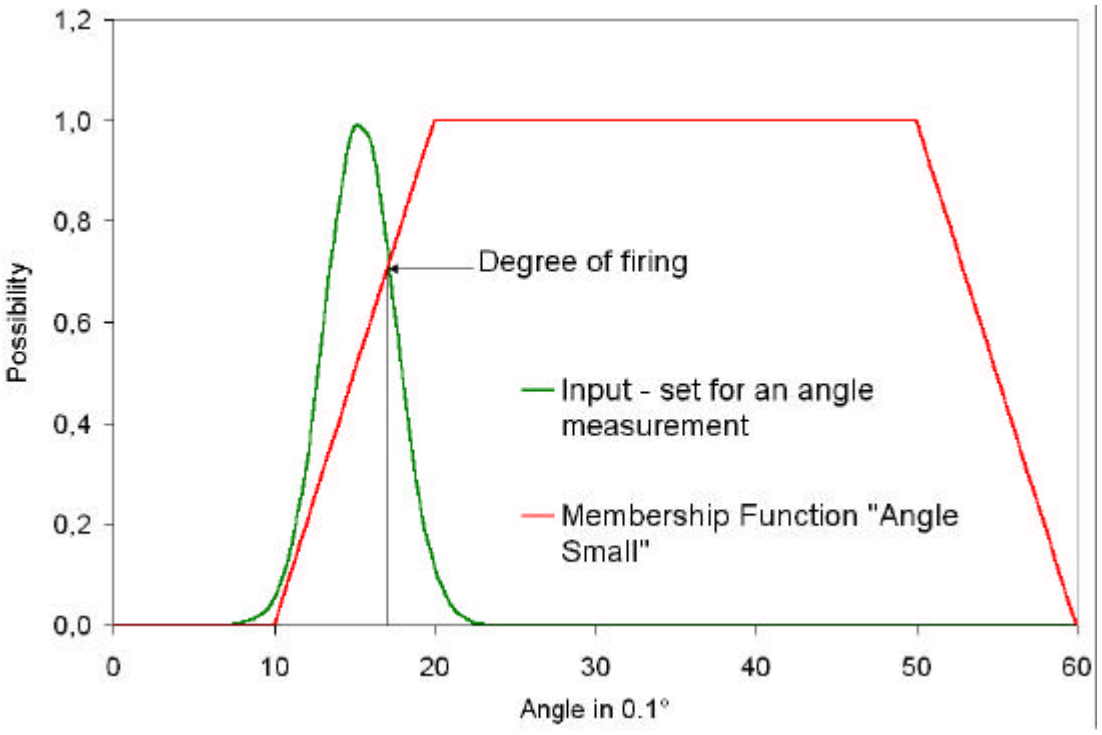

Figure 50: Calculating the degree of firing

Next, the actual rule is checked by performing a simple Min-operation. The result for rule 9 is shown on the next chart as an example:

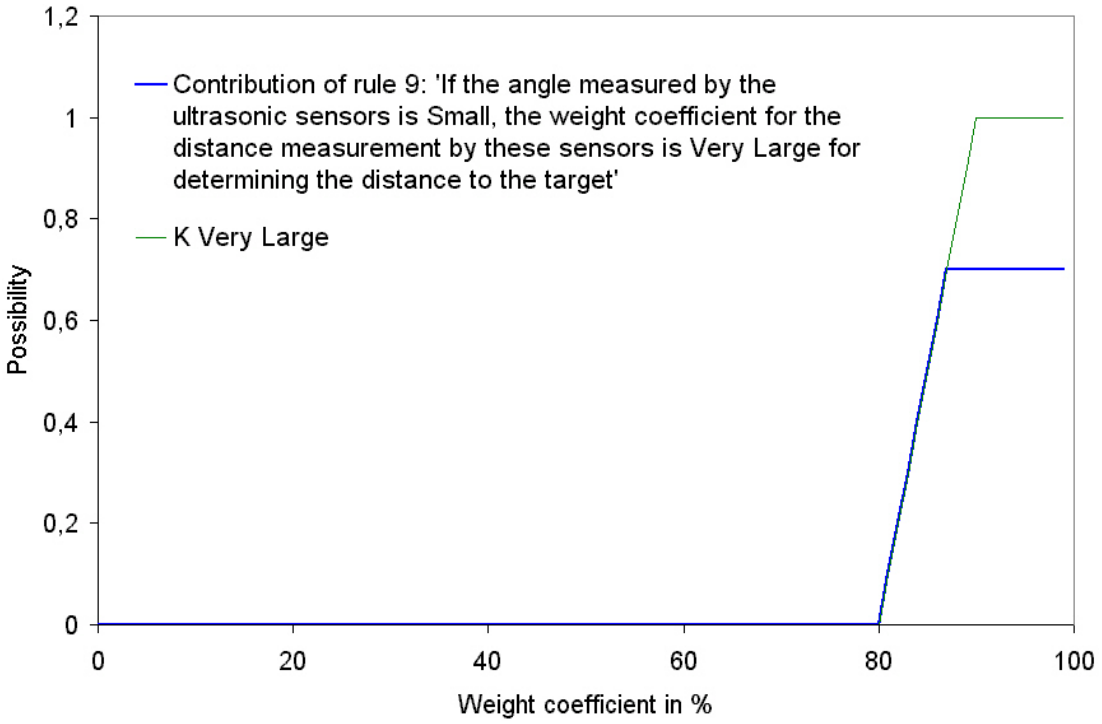

Figure 51: Fuzzy set for one rule

Now, this rule needs to be added to the existing rules, which is performed by a Max – operation:

$$P(y) = Max_i\left\{Min_i\left[t_i, B(y)\right]\right\}$$

These steps need to be iterated for every rule. The result is a weight function P(y), which is influenced by all the applied rules for the output parameter in question. At this point, a small deviation from the classical theory of fuzzy logic was made. It was noticed that by using the Max-Min inferention, multiple rules pointing into the same direction concerning the value that should be accorded to the output parameter, weigh no more in the final balance as one single rule proposing an opposite value, supposing this rule also reaches a maximum possibility. One way to tackle this problem could be to make use of a smaller conjunction operator as the used minimum operator is in fact the largest, or to use sub-normal subsets, but these are only makeshifts to deal with this situation. Therefore, instead of using the maximum operator to add rules, a simple addition was made. This leads to possibilities greater than one, which is a misdoing for classical fuzzy logic theory, but it leads to better and more logical results. After all, this is also what humans do: if a certain proposition is made several times, it is going to be believed more than another one stated only once, even if this was with a so-called probability of one.

A resulting distribution has for example the following form:

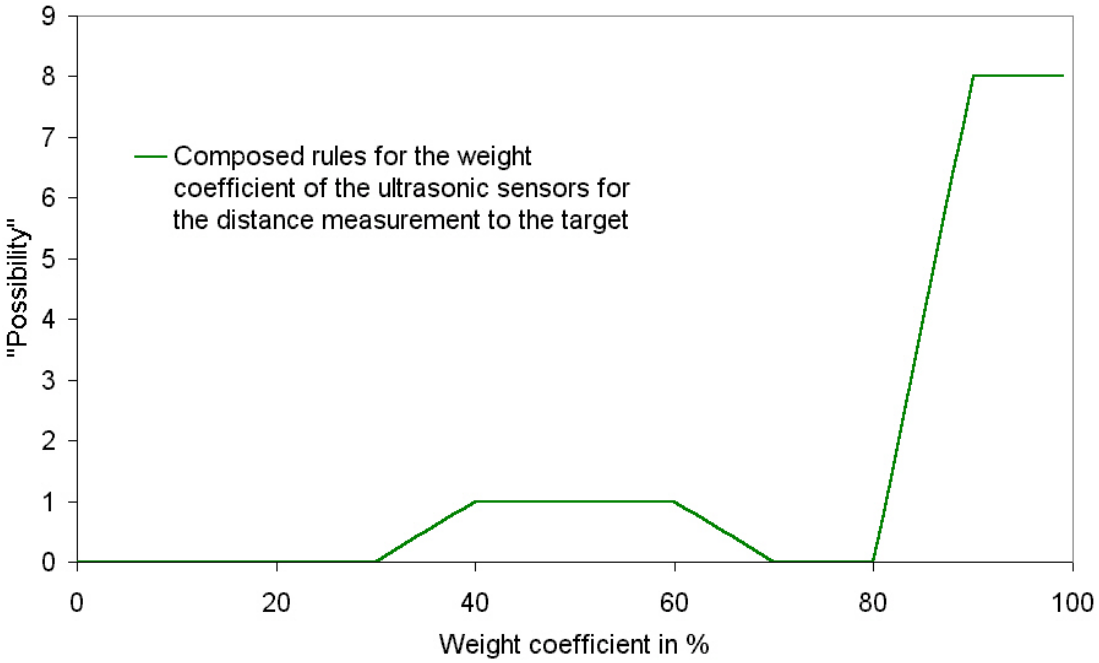

Figure 52: Weight function used for determining the weight coefficient of the ultrasonic sensor

Now all that needs to be done is to defuzzify this function to retrieve one useful value. The centre of gravity (COG) method is used for this process, meaning the average of the weight function is calculated as were it the centre of gravity of the area under this function:

$$Y_{mean} = \frac{\int P(y).y\ dy}{\int P(y)\ dy}$$

For the example case presented on the above charts, this defuzzification process leads to a weight coefficient of 83% for the ultrasonic sensors, whereas a value of 64% would have been found according to the classical theory. When looking at figure 50, this latter value seems less logic than the first one, which is why the adaptation was performed as this “logic” is a key aspect that needs to be brought into the robot control algorithm.

Programming issues

Before beginning the programming of the fuzzy logic based controller for sensor data fusion, a study was made of existing packages enabling to graphically build up a controller, yet all these programs do not offer the flexibility and tweaking capabilities possible with a the self-written program, so the sensor fusion module was build from scratch as for the rest of the program source - except for the camera routines of course. By doing this, the explicit accuracy bounds method could be seamlessly integrated with the fuzzy logic controller and the substitution of the maximum – operator by an addition could be implemented into the controller.
Having read the preceding explanation of the program function, it must be easy to understand the high-level program source code. As an example, the code for calculating one rule, more precise rule number 9, is presented below:

```
Validation = Supremum(US_Angle_Curve,Angle_Small,Angle_Precision);
Minimum(Validation,K_VeryLarge,NewRule,K_Precision);
Maximum(Rules,NewRule,Rules,K_Precision,Method);
```

The first line computes the degree of firing using the input set US_Angle_Curve and the membership function Angle_Small.
The second line does the calculation for the rule itself, using the degree of firing and the output membership function K_VeryLarge and stores the result in a NewRule function.
The last line aggregates the rules, using the maximum operator if method is set to one, or otherwise simply by adding the rules. The former Rules function and the newly calculated NewRule are thus stored in the Rules function and the program is ready to process a new rule.
In the end, the rule is defuzzified and the weighting factor for the ultrasonic sensor measurement is known:

```
K_US = Defuzzify(Regel,K_Precisie);
```

It may be clear that this program code is ready to be reused to build up other kinds of fuzzy logic based controllers, without the need for the knowledge of the entire fuzzy logic background.

Chapter 6: Map building and path planning

Introduction

In many robotics - applications and especially in the case of autonomous guided vehicles it is necessary for the robot to hold some sort of representation, or "map", of its environment. The specific nature of this representation depends strongly on the kind of information one wishes to derive from this map. In this case, the map is used to retrieve navigational information for the robot. As a result, the used path planning techniques are closely related to the type of implemented map, which is why these two subjects are handled together in this chapter. The type of input – data, which is used for the map building process, will also have its influence on the type of algorithm used, as does the used control architecture. Considering all these different possibilities, it should not come as a surprise that there exists a vast multitude of paradigms for map building, yet all these can be categorized into two distinctive sets of theorems: the grid-based and the topological approach.

In theory

Grid or topological map?

Comparison

When using a grid, the environment is represented by evenly – spaced grids indicating, for example, the presence of an obstacle in the corresponding region of the environment. When using the topological approach, the environment is represented by graphs. Each node in such a graph corresponds to a distinct place or landmark. Arcs connect these nodes if there exists a direct path between them. Both methods have their advantages and disadvantages [14]; the most important ones are stated below:

Grid map:

+ These maps are very easy to build and to maintain
+ The relationship between map and environment is straightforward
+ Multiple viewpoints can easily be integrated by using a coordinate – transformation
+ Calculation of shortest path is fairly easy
– Very memory and space consuming since the complexity of the map does not depend on the complexity of the environment

- As a result of the previous point the processing of these kind of maps can be very time – consuming
- The position of the robot itself must be known accurately
- Since a grid map provides a lower level of abstraction compared to topological maps, the returned data will require more pre-processing when used by symbolic problem solvers such as behaviour coordination algorithms.

Topological map:

+ Efficient planning, memory and space – saving, since the resolution depends on the complexity of the environment
+ Faster
+ The exact robot position isn't that important
+ Very convenient interface towards symbolic problem solvers
- Quite difficult to construct and to maintain
- Requires recognition of landmarks and places: one must have the sensory equipment for doing this and even then, it is no simple task
- Paths calculated based on topological maps may not be optimal in terms of energy consumption or distance travelled

Grid maps

Grid maps are in most cases discrete, 2-dimensional occupancy grids, in which each cell has a value attached that marks the belief of finding an obstacle in the corresponding region of the environment. The cell values, or occupancy values, are determined based on consecutive sensor readings. The building process of grid maps can be divided into 4 distinct components:

1. *Interpretation*: The sensor readings must result in occupancy values for each cell. The interpretation process is facilitated by the use of the fuzzy logic sensor-fusion component, which precedes the map-building procedure. Normally, one would consider using artificial neural networks for fusing the different raw sensor readings directly onto the map, but given the input parameters of the map-building procedure, this is no longer necessary. As not only mere position information regarding an obstacle or the target is provided, but also the standard deviation on this abstract measurement, this extra data must be taken into account usefully. This can be done by reconstructing the gauss-distribution on the map itself, so cells in the neighbourhood of the cell where the obstacle is reported will receive an occupancy value decreasing with their distance to this last cell. Of course, the dimensions of the obstacle-object on the map can be extended with a certain security distance taking into account that the robot isn't really a point-object, or some suspicion regarding the sensor readings.

2. *Integration*: Consecutive results of the interpretation process over time are integrated to come to more reliable maps, which can be performed by a whole number of techniques [16]. The integration of sensor measurements is also a process that the preceding data-fusion component has already taken care of.
3. *Position estimation*: The position of the robot on the map must generally be recalculated and re-estimated after every step. However, since the used map is relative to the robot, the robot position does not need to be referenced to the map after every move; it is the map that is re-referenced to the robot. This approach is used because the map will be initially empty, so the robot would not be able to reference its position to a certain landmark anyway. Changes in the map between movements will thus reflect newly found obstacles, improved position measurements of the target or an obstacle and dead reckoning errors.
4. *Path planning*: Based on a chosen criterion like minimal energy consumption or shortest path, the robot must find a path based on the map towards his goal. This last process is further handled in a following paragraph.

Topological maps

Topological maps are built on top of the grid maps. The idea is to partition the free-space of a grid map into a small number of regions, separated by critical lines. These critical lines correspond to narrow passages such as doorways. The partitioned map is then mapped into an isomorphic graph.
Since topological maps are difficult to maintain, whereas the map will have to be changed continually in the used application; and since they require the recognition of certain landmarks, the implementation of topological maps is not realistic option for this case, so they will not be further discussed.

Path planning techniques

Classical methods

To choose the path planning method used by the robot, first a study was made of some different possible approaches; these include vertex graph path planning, free space navigation, grid based navigation, distance transforms, stream field method and heuristic navigation. A brief introduction to these techniques that were not restrained can be found in appendix D.

Recursive algorithm

The first approach tried out to deal with the path-planning problem was to implement a recursive algorithm calculating all the possible reachable points. The algorithm used to compute these steps as shown on figure 2 was therefore extended to calculate paths to a designated target. However, in view of the recursive nature of the algorithm, calculation for longer distances took excessive amounts of time. An adjustment was made to work with linear paths at greater distances, so the recursive calculation was only made when nearing the target. This resulted in paths as shown on the next figure:

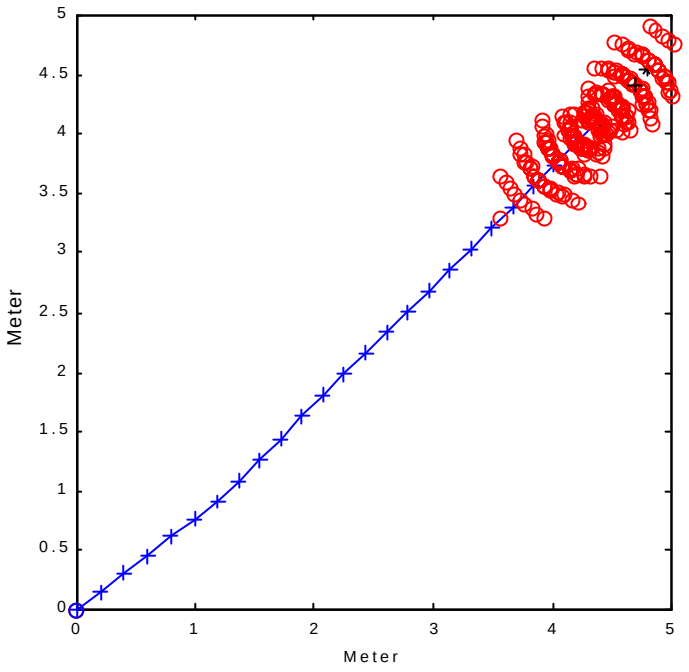

Figure 53: Path calculated with a recursive algorithm

However, this approached lacked the flexibility to easily deal with more complex environments, so this track was abandoned for further implementation.

Potential Field Navigation

This is the main navigation technique that is used by the robot; this choice was made because the potential field method provides a quite natural and logical framework for addressing path-planning problems. Moreover, the potential field method is one of the very few methods able to provide the required map robustness, which is needed, as the robot will have to deal with highly incomplete environmental data due to the limited field of view of its sensors. Potential field navigation techniques make use of artificial forces: repulsive forces at impassable areas or obstacles keep the vehicle away; an attractive force at the goal point moves it towards the goal.

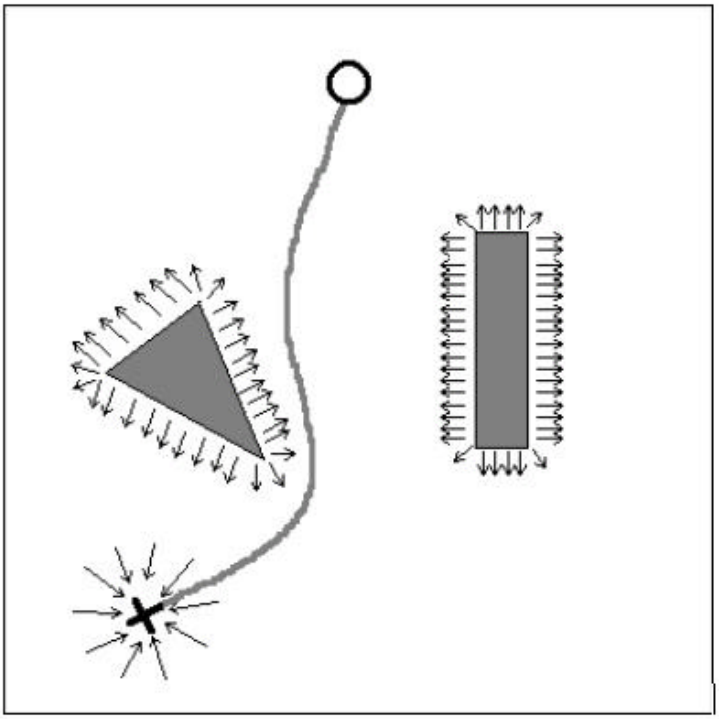

Figure 54: Potential Field Navigation

The potential field has to be tuned in a way such that the robot can never be dragged inside an obstacle and keeps a specified security distance, but moves towards the goal from all points in the environment. This is limited by the existence of local minima; there the resultant force on the robot disappears and a solution path cannot be found. Local minima exist for example on the opposite goal side of obstacles as shown on the following figure.

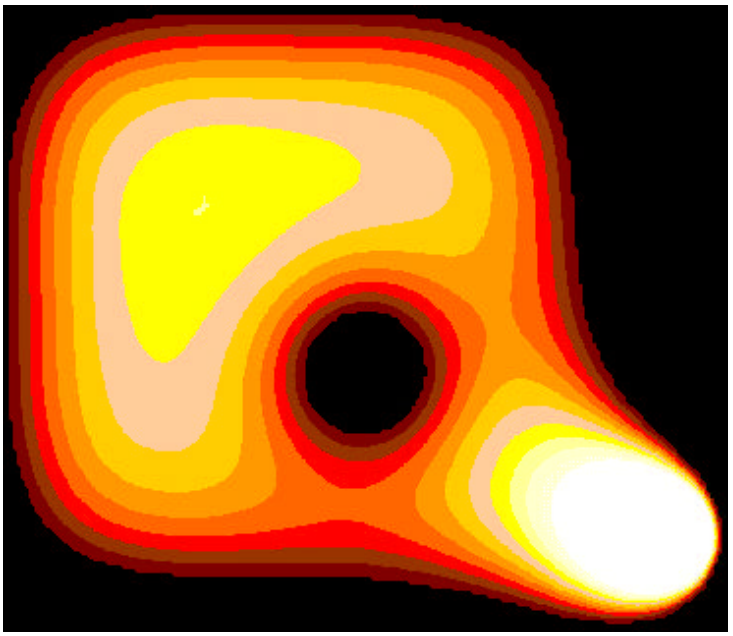

Figure 55: Local minimum on a map with one obstacle and one target

The local minima problem has not been solved completely to this day, though considerable attention has been given to it. One way to solve the problem is to introduce a random movement of the vehicle hoping to escape a local minimum; another is to relocate the goal temporarily when stuck in a local minimum, or to mark areas that have been visited already as impassable. All these methods do not avoid the existence of local minima and are not always successful in their solutions. The used technique for the robot to avoid local minima is simple, yet very effective: the robot is considered an obstacle itself. By doing this, the robot creates a repulsive force away from the current position and will not get stuck in local minima. A convenient side effect of using this technique is that the progress the robot is making is made visible on the map. A drawback from using this method is that even more of the correlation between the generated map and the physical reality is lost.

Potential field navigation is very suitable for local navigation since the environment has only to be known in the vicinity of the vehicle and only a short piece of the path is calculated with each evaluation of the force fields. This implicates that it is not necessary to calculate on the global map with each step, provided the boundary conditions are known, or that they are not that important for the given problem. Unfortunately, this is not the case; the boundary conditions for local maps used by the robot change in an unpredictable manner, since new obstacles can arise at any time, so it will always be necessary to work with a global map.

Path generation can be implemented easily and effectively, since this calculation can take direct use of the cell values of the map. The basic idea used in implementing the potential field method is to find a harmonic function [15] [18]. A harmonic function ϕ on a region is a function that satisfies Laplace's equation:

$$\Delta f = \frac{\partial^2 f}{\partial x_1^{\,2}} + \frac{\partial^2 f}{\partial x_2^{\,2}} + \ldots = 0$$

Having continuous second derivatives in the interior of the region.

It is possible to prove analytically that a harmonic function has neither a minimal point nor a maximal point in the interior of the region. Here the nature is explained intuitively. If a point is a minimal point, then a sum of the second derivatives has the positive sign. If a point is a maximal point, then a sum of the second derivatives has the negative sign. Therefore, if a sum of the second derivatives is zero, the point is neither a minimal point nor a maximal point. Consequently, there can be theoretically no minimal point in the interior of the region that satisfies Laplace's equation. In practice, adding arbitrary obstacles and targets to the potential field breaks the nature of the harmonic functions, so local minima do arise, but, as stated earlier, the technique of considering the robot as an obstacle deals with this problem adequately.

Behaviour Based Navigation

Behaviour Based Navigation for mobile vehicles is based upon breaking up the navigation task into basic behaviours, e.g. "avoid obstacles", "follow wall", "stay on path", etc. The combination of these "micro-behaviours" results in the desired "macro-behaviour" of the vehicle. This approach results in solution paths that may be hard to foresee, but present an opportunity for reactive navigation independent from geometric path modelling models.

These behaviours are defined as "Motor-schemas". Motor-schemas origin in psychology and neurology and describe the interaction between perception and action of living beings.

Motor-schemas are usually implemented using artificial potential fields. Each Motor-schema is implemented using a separate field or field property, e.g. an attractive force of the goal reflects “Move-towards-Goal”. Combining all Motor-schemas or behaviours is usually done by simply adding all forces at the vehicle’s position vectorially. One advantage of Motor-schemas is that they can be activated or de-activate separately as desired. For example, when travelling over a bridge a "stay-on-path" behaviour is essential, but it may be de-activated when travelling in a large area of free space in order to give the vehicle more flexibility in its path. Alternatively, the schemas might be given different priorities in case of opposing objectives.

The robot is actually given two such behaviours:

1. Search a path using the potential field method in normal situations.
2. Head straight for the target when there are no obstacles in the way and the robot is aligned towards the target.

The followed behaviour is determined by the sensor fusion component. More specifically, as soon as the explicit accuracy bounds method determines that the ultrasonic sensors are measuring the target and not an obstacle, the second behaviour is chosen as these sensors have a very limited opening angle, so the robot must be more or less orientated towards the target.

The reason for implementing this behaviour-based navigation is that it is not very intelligent to make time-consuming calculations to find a path towards the target if this target is straight ahead.

In practice

Defining the map parameters

Received Input

The map – building procedure gets its input parameters from the fuzzy logic data-fusion component, which fuses the sensor readings of the 4 abstract sensors. This fusion-procedure delivers the following output-data as an input to the map building process:

- Distance to the target
- Standard deviation on this distance
- Angle to the target
- Standard deviation on this angle

- Distance to an obstacle
- Standard deviation on this distance
- Angle to an obstacle
- Standard deviation on this angle

Required Output

The map building and path-planning module must return the best move the robot can do as a step towards reaching the goal. There is no need to perform a full path calculation every time, since it may be expected that the map will change as new information is gathered after the movement, so the rest of the path will become useless. This continuous alteration of the map is caused by the very limited field of view of the robot sensors, so the robot will have to manage with very incomplete maps and will have to recalculate a new path to the target after every move, since more information will be available every time.
It is also not necessary for the robot map to be a very exact representation of the physical reality, the goal is to do path planning, not to output detailed CAD-drawings of the explored environment.

Grid parameters

The number of cells used is an extremely important parameter in the map building process. For obtaining a decent resolution, the number of elements must be high enough. At a first instance, it seems logical to work with a map that has the same resolution as the sensors used. It is clear that a high resolution is always wanted, and consequently a lot of grid cells, but there's an extremely restrictive factor to be taken into account here: time. When using a direct method (e.g. Gauss-method) for solving the system, time consumed for calculation is proportionate to N^3, N being the number of grid cells. By using the Jacobi or Gauss-Seidel iterative techniques, one can reduce this to N^2, when using optimal relaxation to $N\sqrt{N}$ and with multigrid even to N [21].
The map used by the robot is a 250 x 250 map representing a 5m x 5m area, so the resolution is 2cm, which is about the sensor resolution level of the ultrasonic sensors. The most important guide for determining the grid distance was to make sure that two reachable points by the robot in the environment should be represented by two different cells on the map. Otherwise, problems could arise with the path planning procedure. The robot takes steps of 23cm (= 11.5cm on the map) and can turn over an angle of about 16°, so two reachable points are always more than 2cm apart.

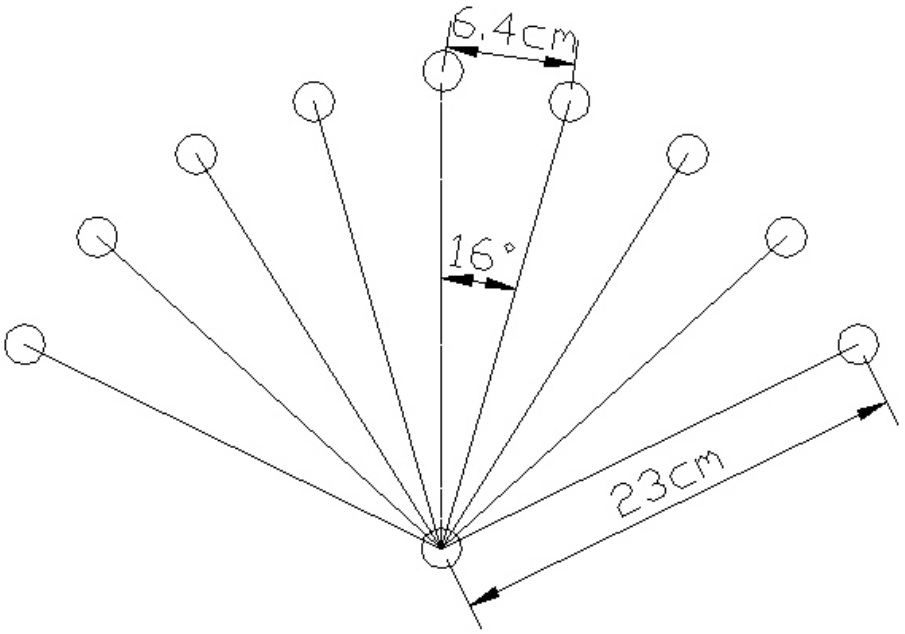

Figure 56: Two reachable points are always more than 2cm apart

A classic benchmark for finding a good map resolution is the so-called "doorway passing" problem. The resolution necessary to be able to pass through a door expressed as the minimum length reflected in the map - or the maximum side length of a cell in a grid-based map - is in general determined to:

$$s_{\min} = \frac{w - d_{robot} - d_{security}}{2}$$

Where w is the width of the smallest door that has to be passed, d_{robot} the diameter of the (assumed to be circular) vehicle and $d_{security}$ the security distance the robot has to keep to each obstacle. Our robot is hardly circular, so this d_{robot} is arbitrarily chosen to be the ultimate distance from the front feet to the measurement reference point, which is at the base of the camera socle. When applied to the presented robot and its parameters, the above formula enables to calculate the width of the smallest door that can be passed:

$$\begin{aligned} w &= 2 \,.\, d_{\min} + d_{robot} + d_{security} \\ &= 2 \,.\, 2\text{cm} + 30\text{cm} + 15\text{cm} \\ &\approx 50\text{cm} \end{aligned}$$

However, the values of 30cm and 15cm used respectively for the robot diameter and the security distance are quite arbitrary, so no exaggerated value should be accorded to this calculation.

Implementation techniques

The choice for the potential field method implicates that the Laplacian $\Delta f = 0$ of the presented field will have to be computed. These techniques were proposed for dealing with the problem:

Multigrid method

As stated above, the multigrid method is certainly the fastest of all the iterative methods. The basic idea is quite simple: the iteration is performed on multiple grids, each counting half as many points as the former, so the low frequency errors, which cause the other iterative methods to converge so slowly, are damped quickly on a coarser grid. The difficulty is to preserve the truncation error of the finest grid and to make sure that no new high-frequent errors are introduced. Since the programming of this method is not very simple and only existing source code for solving linear systems was available, this technique could not be used.

Analytical method

An interesting, yet quite unknown approach for solving the Laplacian is the analytical way [20]. The basic idea here is to write a general solution of the problem as a function of some unknown coefficients, which are calculated by performing a least-squares optimisation considering boundary conditions and singularities. This method would fit the map-building problem as posed for the robot nicely, since the used map consists typically of singularities as obstacles and target, and that is especially the kind of problem this method is designed for. As this technique was only learned about after the map building and path-planning module was actually written, it could not be introduced into the program structure due to a lack of time.

The Gauss-Seidel method

The iterative Gauss-Seidel method presents a technique to calculate a potential field in two-dimensional space. As this is the method actually used by the robot for its map calculations, this process is discussed more in detail below.

To begin, the Laplace equation must be transformed into a discrete form. If the step sizes are all equal, this can simply be done by writing:

$$f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j} = 0$$

While

$$f_{i,j} = \frac{1}{4}(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1})$$

This equation illustrates that a potential on a mesh point is the mean of the values on the adjacent points. In order to satisfy the Laplacian over the whole region, we apply the Gauss-Seidel iterative method:

$$f_{i,j}^{(n)} = \frac{1}{4}(f_{i+1,j}^{(n)} + f_{i-1,j}^{(n-1)} + f_{i,j+1}^{(n)} + f_{i,j-1}^{(n-1)})$$

Where ϕ is a numerical solution on the mesh point (i, j) obtained from the n^{th} iteration of the equation. In order to speed up the calculations, the Successive Over-Relaxation (SOR) was used. The idea is to artificially increase the change between the new and the old cell value with a certain factor ω.

$$f_{i,j}^{(n)} = f_{i,j}^{(n-1)} + w.\left[\frac{1}{4}(f_{i+1,j}^{(n)} + f_{i-1,j}^{(n-1)} + f_{i,j+1}^{(n)} + f_{i,j-1}^{(n-1)}) - f_{i,j}^{(n-1)}\right]$$

An optimal value for ω was found by trial and error; since this value is only 1.3, the performance gain is not that impressive, but it is noticeable.

In order to avoid a creation of an unexpected minimal point due to a numerical calculation error, an initial value is set on each mesh point in the interior of the free space before the Gauss-Seidel iteration begins. This initial value for the free space is a very time-determining factor in the iteration process. It's clear that when this value is quite different from the solution, the iteration will take more steps. This is the reason why the implemented program uses a high initialisation, meaning the free-space cells are set to a value closer to the obstacle (=high) - level than to the target (=low) - level. Concrete, short integers are used for the cell values as experiments with chars showed a lack of resolution, so boundaries and obstacles are set to a value of +32767 and the target is given the value –32768. For a faster execution, the free-space cells were not initialised to 0 or 1, but to 25000. In consequence of the initial condition, a lower potential value propagates from goal point while the iteration progress. As a result of the propagation, the value of a point that is closer to goal point becomes lower. According to the boundary conditions, the potential field takes a high value at the surface of obstacles and takes a minimum value at the goal point.

A field of discussion is the exit condition for the iteration loop. Normally, one would stop the iteration as soon as the gradient of the potential field around a start point becomes large enough to determine the direction to a goal, so the time required for the calculation of the potential field would depend on the location of the starting point. But it is also important to make sure that a newly added obstacle can carry through its influence to the position of the robot, so a certain number of iterations is required in every case. Another point to keep in mind is that during the calculation of the very first step the map changes a great deal, since the process must start with the given initial conditions, so the gradient of the potential field around a starting point could also undergo some dramatic changes before converging to a stable value. This is the reason why initially the change was calculated between newly found maps and the previous version. When this change became low enough, the iteration could be halted. The number of iterations needed for the first and the later map calculations to come to stable maps were noted and brought into the program. Thus, the time consuming operation of recording the changes made to the map is no longer necessary in the final program.

Programming issues

The map building and path planning process is the time determining step in the program flow as it involves a long iterative calculation of the potential field. As a result, great interest has been given to improving the performance of the algorithm and to speed up the calculations, numerous techniques were used:

- Successive overrelaxation
- High initialisation
- Prior determination of the required number of iterations
- More efficient memory management
- Behaviour based navigation

Most of these techniques are already discussed and the results are clear as the time between two consecutive steps was brought down from 8.5 minutes to 8.5 seconds. This time delay may still seem too much, but tests with the program on a more modern computer (AMD K7 750Mhz with 133MHz SDRAM) showed execution times within the time delay of two seconds, which is needed in every case for steering the pneumatic valves.

Experimental results

The actual working of the map building and path-planning module can be shown no better than by presenting the results of a real-world example. The environment set up for this experiment is sketched on the following figure:

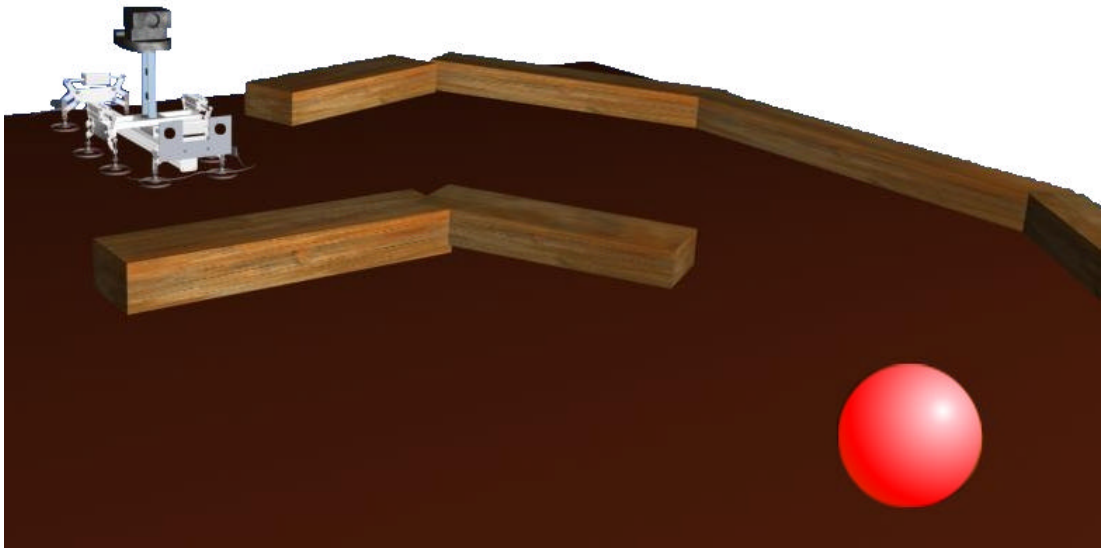

Figure 57: Testing environment put robot localisation numbers on graph

The charts presented on this page show the potential field map at different stages along the way towards the target. These different stages are also marked on the above figure.

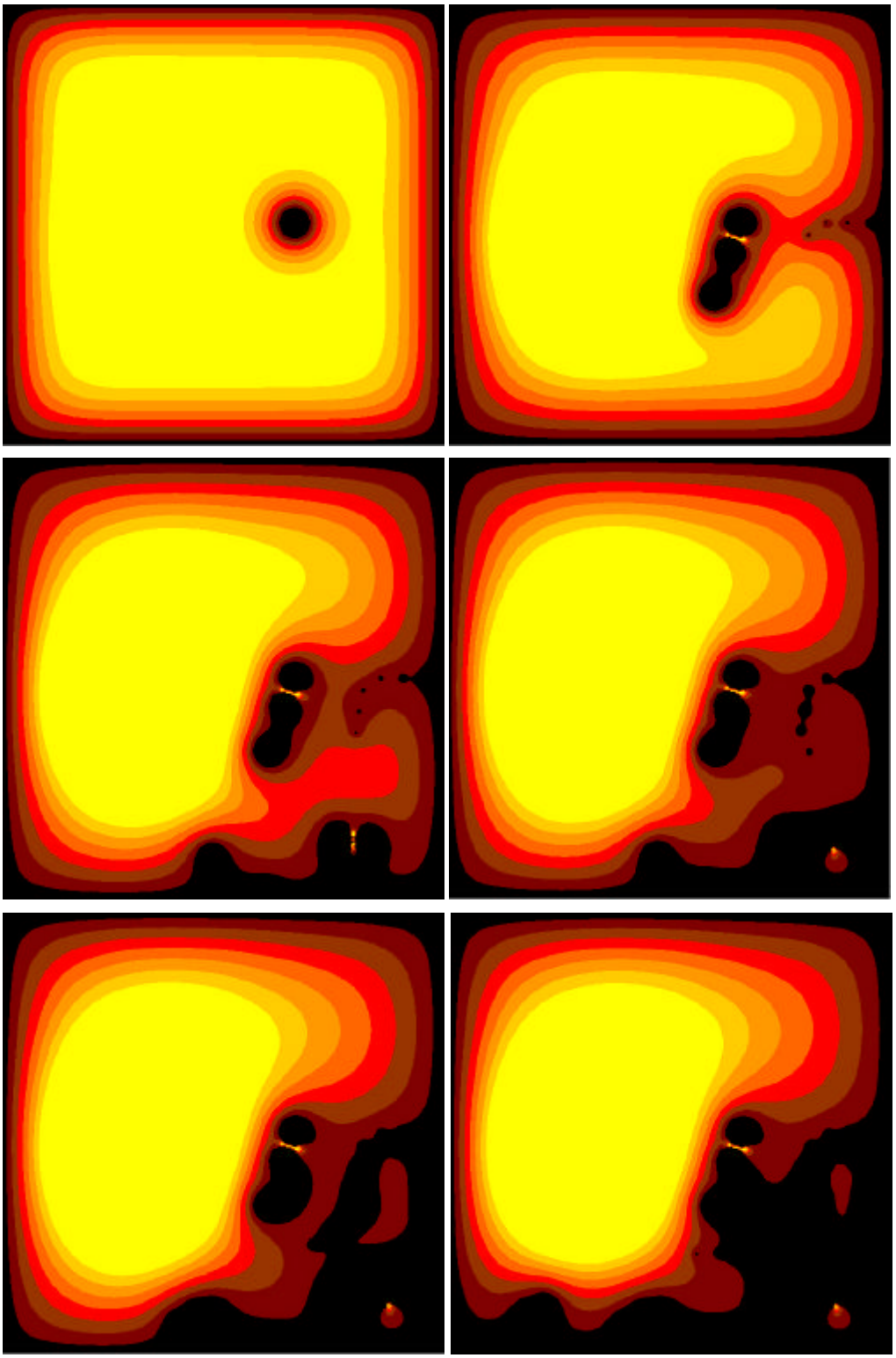

Figure 58: Charts of the potential field as the robot is advancing in the environment

On the final potential field graph shown on the next figure, one can clearly see the path followed by the robot. Note also the correspondence between the environment as shown in figure 55 and this potential field representation.

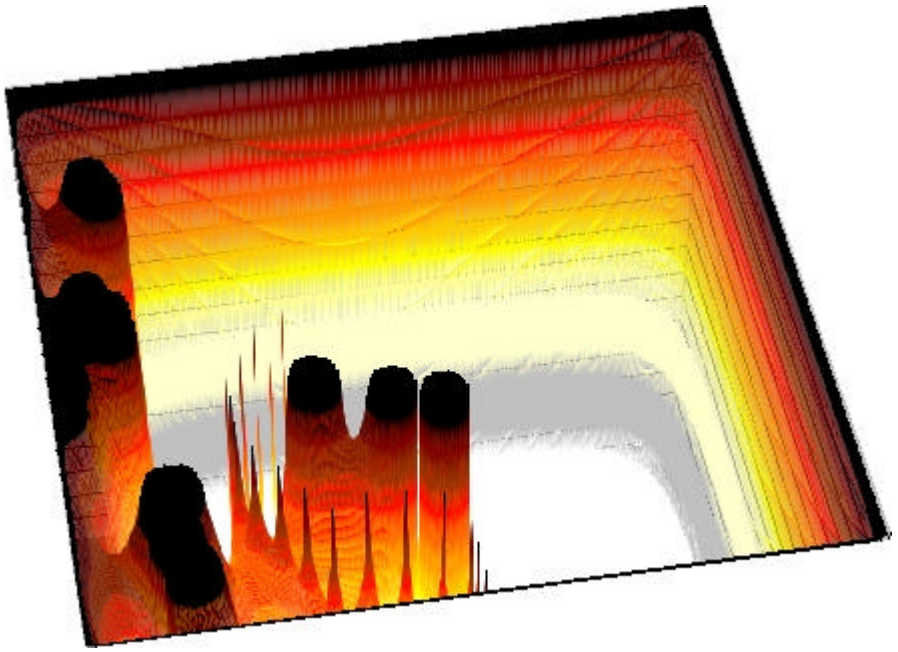

Figure 59: Potential field after completing a run

An interesting situation arises when the doorway passing is made smaller. Eventually the robot will decide to go the other way round as shown on the chart below.

Figure 60: Robot making a large detour before reaching its target

This behaviour can be explained by following the robot logic systematically, which is done here for the key stages denoted on the figure above:

1. The robot starts here
2. The robot chooses to go left as the bal is located to the left of the central axis
3. The robot starts turning left as the detected obstacles on its right side cause a high potential there. It keeps turning left, eventually turning a 180°
4. The robot manoeuvres itself in between a passing on the right side without colliding with one of the obstacles
5. The robot reaches the target point

This conduct may seem erratic to the reader, but note that following the right passageway around the central obstacle was in fact the shortest path solution. However, the robot has no means of knowing this a priori due to the very limited field of view of its sensors. So it is normal for the robot to turn left first, to realise its mistake at a later stage as it has gathered more environmental information and to follow the shortest path eventually. In spite of all these justifications, it cannot be denied that using the potential field navigation technique together with the extremely limited field of view of the robot sensors, results in sometimes less logic behaviour. However, it must be noted that even in conditions where the robot intelligence fails in finding the actual shortest path, the robot still succeeds in reaching its target after a while without any collisions on the way.

Chapter 7: Future perspectives

Introduction

This final term project was the second one to work on the pneumatic robot; in fact it filled in some of the proposals for improving the robot mentioned in the first thesis dissertation [1]. The other future perspectives which were handled in this work and which were not implemented thus far remain valid, but they will not be repeated here.

Blackboard control architecture

The presently used control architecture, which is basically a serial SMPA architecture, is a quite simple solution and therefore, lacks some power to deal with an increasing dataflow in real-time. To address this problem, it may be suited to implement a complete blackboard control architecture as this approach enables parallel processing of all the different modules. This way, the ultrasonic measurement, camera target tracking and measurement, sensor fusion, map building and path-planning component would all be considered as separate threads sharing their information through the blackboard. It may be clear that timing and controlling problems make the actual implementation of such a structure quite a challenge. At present, the blackboard parallel processing technique is only used to integrate the camera target tracking process with the rest of the robot control program and to enable an emergency stop procedure.

Zooming Target Tracking

If the target-tracking algorithm were to have a zooming capability, objects could be recognised and tracked at greater distances. This feature wasn't implemented during the time of this project due to problems concerning the knowledge of the internal working of the target tracking algorithm and other problems concerning the adaptation of the camera control and distance measurement routines.

Multiple ultrasonic sensors

If more ultrasonic sensors could be used, this would mean that the robot would gain much more information about its surroundings at a time. Now, the robots field of view is extremely limited as the two sensors used provide only information about the environment straight ahead. Using more ultrasonic sensors would implicate having better maps, the possibility to work with topological maps and to use other means of path planning. On the other hand, it would pose a new challenge to intelligently fuse the now more complex data flow. In this context, an interesting option is to use multiple sensors as receivers for one measurement, coming to a tri-aural ultrasonic sensor architecture as shown on the following image:

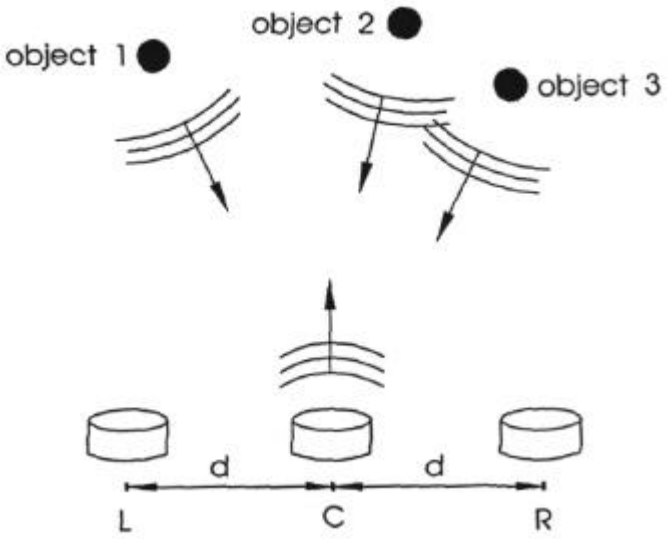

Figure 61: Tri-aural sensor array

The central sensor is used both as a transmitter and a receiver, the two peripheral ones only as receivers. Using the phase-shift of a reflected signal over the three sensors, a far better angle measurement is possible than with a normal sensor configuration. By applying neural network techniques on the returned sensor data, as explained in [22], qualitative information can be retrieved out of the environment.

Object recognition with ultrasonic sensors

A common problem of ultrasonic sensors is that they are unaware of what they are actually measuring. This must not be accepted without posing questions. When looking at nature, some animals such as dolphins and bats show that it is perfectly possible to recognize preys, or in the robots' case objects, using ultrasonic sensors. A procedure for achieving object recognition is based upon the analysis of the returned echo in the frequency domain. Another way is to make use of the envelope function, which is a simple graph of the returned analogue signal. An example of both characteristics is shown below:

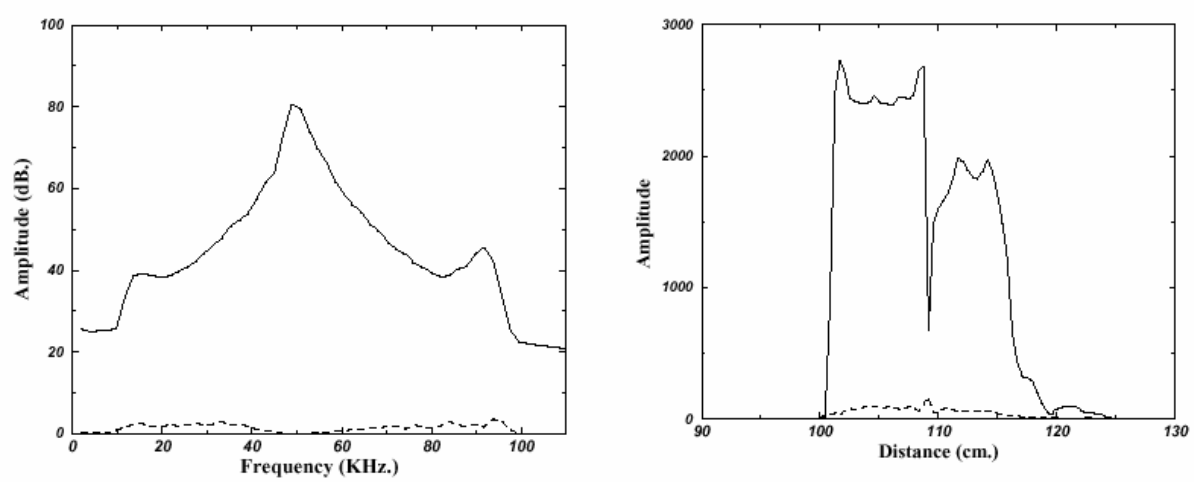

Figure 62 : Power Spectral Density diagram and Envelope – function

These characteristics can be recognized by a fuzzy neural network and can thus lead to the recognition of the reflecting object.

Proximity sensors

In its current condition, the robot is actually blind at close distances, since neither the ultrasonic sensors, neither the camera provide useful data at this range. Of course, the bumper switches are still present but firstly they are still quite unreliable and fragile and secondly, one could state that it is already too late when they return a signal. Therefore, it may still be a good idea to add some sensors based upon other physical principles of measurement such as infrared or tactile sensors

Intermediate positions possible with pistons

One of the main problems limiting the robot in its applications is that it is only capable of raw binary movements: steps are always 23cm and rotations are always 16°. To improve the spatial resolution of the robot, an interesting improvement would be to build a system to control the pressure to the pistons, using pulse width modulation. This would enable the cylinder, and thus also the robot, to reach intermediate positions.

Gripper

Now the robot is capable of walking towards a certain target object, the logical next step is to grab this object and to return it to the user. This can sadly enough not be done by simply adding a gripper to the robot in its current state as the lack of spatial resolution doesn't allow the robot to position itself precisely enough for such an operation. Another problem would be that the robot would not "see" this target object to be gripped as it would be at a too close range, so it may be clear that some other improvements stated above must be realised first before a gripper can be added to the robot, given one can find a suited insertion location for it on the robot.

Chapter 8: Conclusions

The actual result of this final term project is a robot control program, which enables the robot to meet the expectations set up at the beginning of the project. These functionalities are summarised and discussed on the basis of the user interface of this control program as presented on the next figure:

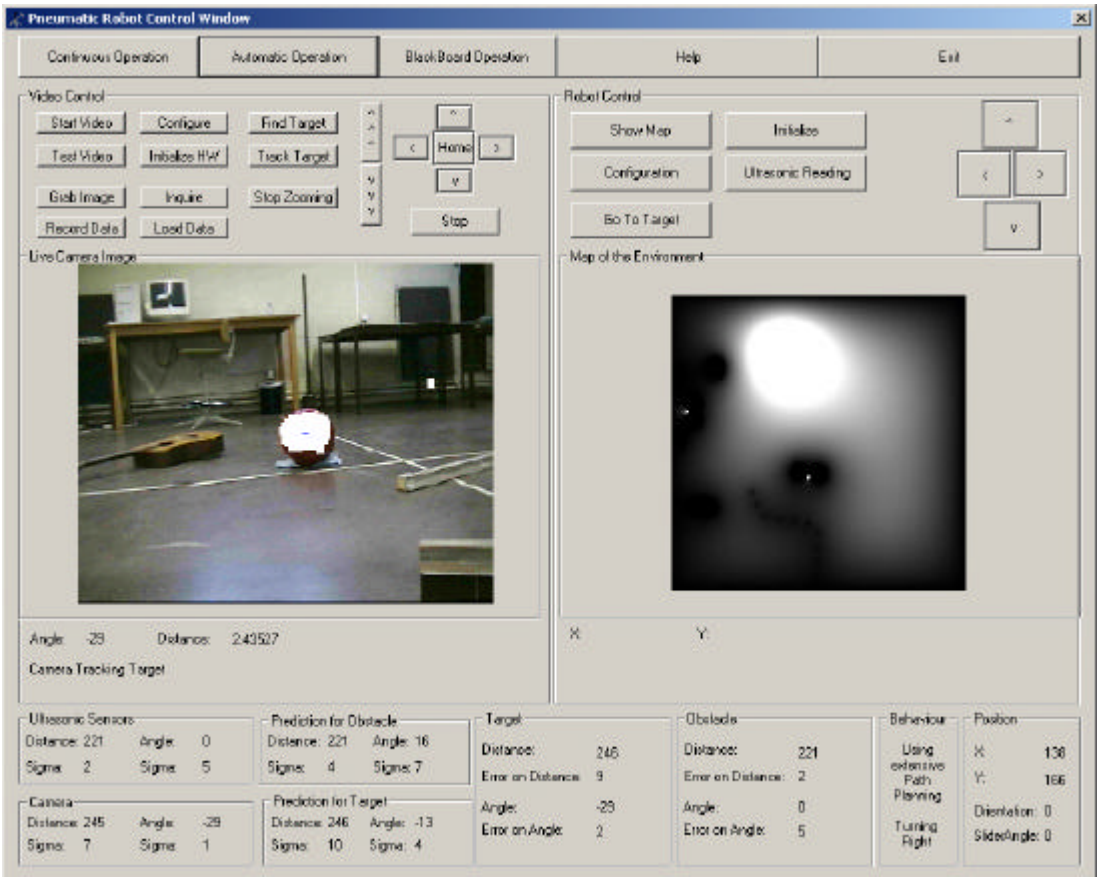

Figure 63: Snapshot of the robot control program

Replace figure with better snapshot

On the top left side of the interface, the user can choose for one of the three operation modes for the robot:

- Automatic Operation: The robot walks towards the target guided by the camera, but with this camera target tracking process integrated in the serial SMPA structure, so no parallel processing is performed.
- Continuous Operation: Generally the same as Automatic Operation, with this exception that the camera never stops tracking the target when this target has been reached after a run.

When the target object is moved later, the camera will detect this and order the robot to start walking towards this new target position again.

- Blackboard Operation: In this working mode, the camera target tracking procedure runs as a parallel process, implementing as such a blackboard control architecture.

In the middle part of the interface, a clear distinction has been made between the camera control to the left and the robot control to the right. A number of different control buttons enable the user to control the robot and camera directly, to set up parameters or to access basic functions. In order to enhance the user-friendliness of the program and to enable a possible remote surveillance, the camera video image and the potential field shadow map as calculated by the robot, are shown to the user.
In the lower right corner, the sixteen measurements of the four abstract sensors are made visible and more to the right, one can analyse how these readings have been fused to come to unambiguous data about target and obstacle. Even more towards the lower right corner, the program shows which behaviour and control action were deduced from these sensor readings. Finally, one can observe the robot position and orientation in the lower right corner.

Using this control program, the robot is able to navigate itself towards a certain target in a complex and a priori unknown environment, as was asked. The robot is furthermore capable of dealing with moving targets as long as the camera is able to follow this target object, which is limited by the speed of the camera target tracking algorithm and the controlling computer, and the maximum camera pan angle. On the other hand, the robot control program isn't designed to cope with non static environments, or more specifically with moving obstacles. The reason is that to deal with the problem of the extremely limited view of the robot sensors, the high potential of any detected obstacle is artificially kept into this high state for the rest of the operation time, unlike what is done for the target. This implicates that obstacles are never removed from the map and are therefore considered immobile. This approach is inevitable, however, because otherwise the robot would in general never have the notion of more than one obstacle at a time, which would be fatal for a decent path planning.

Though "logic behaviour" has been a key aspect pursued throughout the implementation of the different robot control program modules, it must be admitted that this robot behaviour is sometimes not that logic, due to the combination of the potential field navigation method with the always very incomplete knowledge of the environment. However, with the given sensory equipment, this cannot be averted, unless the camera should also be used for retrieving other environmental information than the target object location. The potential field technique shows furthermore its robustness as tests showed that the target object could be put out of view for a considerable part of the operation time, without keeping the robot from reaching this target in the end.

Another imperfection one could point out is the slow movement speed of the robot, as it needs about ten seconds of "thinking time" between two consecutive movements. As opposed to the logic behaviour, speed was not a key consideration throughout the implementation of the robot control program, as the pneumatic robot was already slow by itself and it was never the idea to make it a racing robot. Nevertheless, a bt of efforts have been made during the implementation process to accelerate all the different routines used, the most dramatic result being the reduction of the calculation time for the potential field. This computing time is still considerable however, but as already discussed in the map-building chapter, this delay would be hardly noticeable on a modern computer.

In spite of all its limitations, the robot succeeds in performing the task that was set up at the beginning of the project: to walk towards a (moving) target in a complete unknown and complex environment with obstacles. This was not a simple demand in view of the limited sensory equipment of the robot and the fact that this pneumatic robot was not actually built to perform such a task. Moreover, reusability of all the different programming components has been ensured, because it happens too often that interesting project results are disassembled and never used again due to a lack of portability.

Abbreviations

AMD.....Advanced Micro Devices
ATM.....Arc Transversal Median
CAD.....Computer Aided Design
CCD.....Charge Coupled Device
COG.....Centre Of Gravity
COM.....Communications
I/O.....Input / Output
JIRA.....Japanese Industrial Robot Association
LSA.....Logical Sensor / Actuator
NIDAQ.....National Instruments Digital Acquisition
PC.....Personal Computer
PCI.....Peripheral Component Interconnect
RGB.....Red Green Blue
RMA.....Royal Military Academy
PII.....Intel Pentium 2
SDRAM.....Synchronous Dynamic Random Access Memory
SMPA.....Sense Model Plan Act
US.....Ultrasonic Sensor
VISCA.....Video System Control Architecture

List of figures

References

[1] “Ontwerp van een pneumatisch aangedreven robot” by Ronald Van Ham, VUB, Brussels, Belgium, 1999

[2] “Memory Management, Navigation and Map Building” by Gerd G. Schenkel, Universität Stuttgart, Stuttgart, Germany, 1994

[3] “Achieving Goals Through Interaction with Sensors and Actuators” by John Budenske and Maria Gini, Department of Computer Science, University of Minnesota, Minneapolis, USA, 1992

[4] ”Colour target detection and tracking” by P. Hong, H.Sahli, E. Colon and Y.Baudoin, Proceedings of the Third International Conference on Climbing and Walking Robots CLAWAR 2000, pp. 711 – 721, Madrid, Spain, 2 – 4 October 2000

[5] “Multi-Sensor Fusion” by Richard R. Brooks and S.S. Iyengar, Prentice Hall, New Jersey, USA, 1998

[6] “Optimal Fusion of Sensors”, Ph. D. Dissertation by Thomas Dall Larsen, Technical University of Denmark, Department of Automation, Lyngby, September 1998

[7] “A Multivalued Logic Approach to Integrating Planning and Control”, Allessandro Saffiotti, Kurt Konolige and Enrique H. Ruspini, Artificial Intelligence Center, Menlo Park, Califonia, USA, 1995

[8] “Object Recognition with Ultrasonic Sensors”, M. Ihsan Ecemis and Paolo Gaudiano, Boston University Neurobotics Laboratory, Dept. of Cognitive and Neural Systems, Boston, USA, 1999

[9] “NI-DAQ™ Function Reference Manual for PC Compatibles”, Version 6.6, National Instruments Corporation, Austin, Texas, USA, 1999

[10] “Fuzzy logic in Autonomous Robot Navigation”, Alessandro Saffiotti, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, november 1995

[11] “Triangulation based Fusion of Ultrasonic Sensor Data”, O. Wijk, P. Jensfelt and H.I Christensen, S3 Automatic Control and Autonomous Systems, Kungliga Tekniska Högskolan Stockholm, Sweden, 1998

[12] “Fuzzy Logic in Autonomous Robotics: behavior coordination”, Alessandro Saffotti Proceedings of the 6th IEEE International Conference on Fuzzy Systems, Barcelona, Spain, July 1997

[13] “An Introduction to the Kalman Filter” by Greg Welch and Gary Bishop, University of North Carolina, Department of Computer Science, Chapel Hill, USA, 1996

[14] “Learning Maps for Indoor Mobile Robot Navigation” by Sebastian Thrun and Arno Bücken, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, April 1996

[15] “Path Planning Using Laplace’s Equation” by C.I. Connolly, J.B. Burns and R. Weiss, University of Massachusetts, Computer and Information Science Department, Amherst, USA, February 1994

[16] “Generating Sonar Maps in Highly Specular Environments” by Andrew Howard and Les Kitchen, University of Melbourne, Department of Computer Science, Victoria, Australia, 1992

[17] “The Arc-Transversal Median Algorithm: an Approach to Increasing Ultrasonic Sensor Accuracy” by Keiji Nagatani, Howie Choset and Nicole Lazar, Carnegie Mellon University, Departments of Mechanical Engineering and Statistics and CALD, Pittsburgh, USA, 1999

[18] “Deadlock-Free Motion Planning using the Laplace Potential Field” by Keisuke SATO, University of Tokyo, Tokyo, Japan, 1988

[19] "Data Fusion and Sensor Integration: State-of-the-Art 1990's" by R.C. Lou and M.G. Kay in Data Fusion in Robotics and Machine Intelligence, M.A. Abidi and R.C. Gonzalez, Academic Press, pp. 7-135, 1992.

[20] “Solving Problems with Singularities Using Boundary Elements” by Dirk Lefeber, Topics in Engineering, Computational Mechanics Publications, Southampton, May 1989

[21] “Numerieke Technieken in de Aërodynamica” by C.Hirsch, course notes edited by Tom Fabri, Brussels, 1999

[22] “Ultrasonic Mobile Robot Perception using Neural Network Techniques” by Jie Chen, University of Ghent, Department of Electronics and Information Systems, Gent, Belgium, 1996

[23] “Véhicule autonome guide” by Bey Temsamani Abdellatif, Université de Mons – Hainaut, Mons, Belgium, 1996

[24] “A Dynamics Approach to Mobile Robot Motion: The Stream Field Method” by D. Keymeulen, Vrije Universiteit Brussel, Brussels, Belgium, June 1994

Appendices

Appendix A: Drawings and pictures

Some Autocad drawings of the robot parts and some photographs will be inserted here.

Appendix B: Datasheets for the Ultrasonic Sensors

Polaroid 6500 Ranging Module

Features:

- Accurate Sonar Ranging from 6 inches to 35 feet
- Drives 50-kHz Electrostatic Transducer with No Additional Interface
- Operates from Single Supply
- Accurate Clock Output Provided for External Use
- Selective Echo Exclusion
- TTL-Compatible
- Multiple Measurement Capability
- Uses TI TL851 and Polaroid 614906 Sonar Ranging Integrated Circuits
- Socketed Digital Chip
- Convenient Terminal Connector
- Variable Gain Control Potentiometer

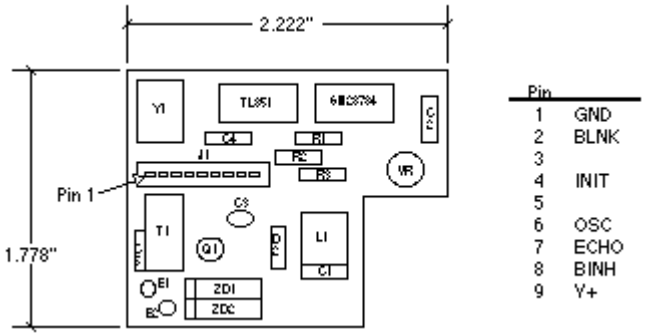

Figure B.1: Ranging Module

The 6500 Series is an economical sonar ranging module that can drive all Polaroid electrostatic transducers with no additional interface. This module, with a simple interface, is able to measure distances from 6 inches to 35 feet. The typical absolute accuracy is ± 10f the reading over the entire range.

This module has an external blanking input that allows selective echo exclusion for operation on a multiple-echo mode. The module is able to differentiate echoes from objects that are only three inches apart. The digitally controlled-gain, variable-bandwidth amplifier minimizes noise and side-lobe detection in sonar applications.

The module has an accurate ceramic-resonator-controlled 420-kHz time-base generator. An output based on the 420-kilohertz time base is provided for external use. The sonar transmit output is 16 cycles at a frequency of 49.4 kilohertz.

The 6500 Series module operates over a supply range of 4.5 volts to 6.8 volts and is characterized for operation from 0° C to 40° C.

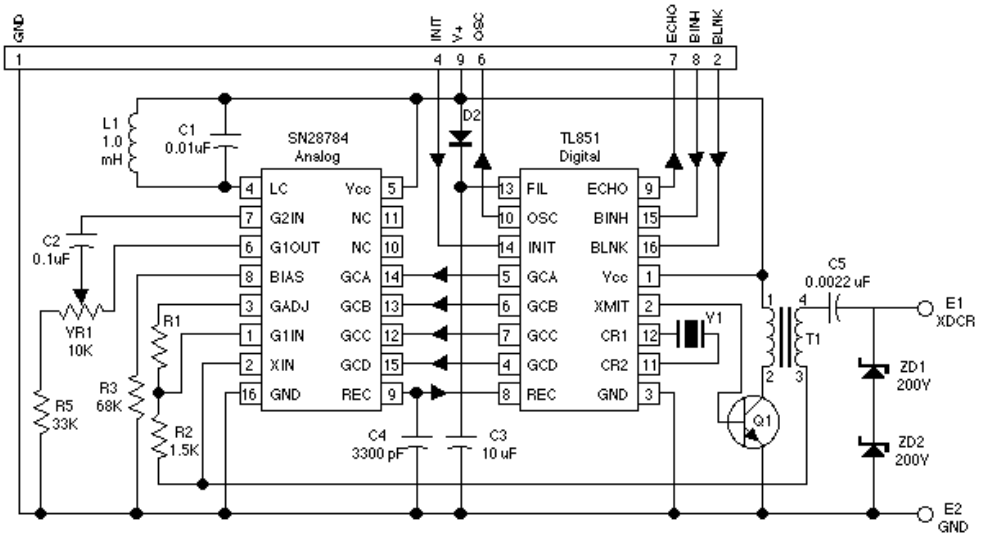

Figure B.2: 6500 Board Schematic

Absolute Maximum Ratings

Voltage from any pin to ground	7 V
Voltage from any pin except XDCR to V_{CC}	-7 to 0.5 V
Operating free-air temperature range	0° C to 40° C
Storage temperature range	-40° C to 85° C

Recommended Operating Conditions

		Min.	Max.	Unit
Supply voltage, V_{CC}		4.5	6.8	V
High-level input voltage, V_{IH}	BLNK, BINH, INIT	2.1		V
Low-level input voltage, V_{IL}	BLNK, BINH, INIT		0.6	V
ECHO and OSC output voltage			6.8	V
Delay time, power up to INIT high			5	ms
Recycle period			80	ms
Operating free-air temperature, T_A		0	40	° C

Electrical Characteristics Over Recommended Ranges of Supply Voltage and Operating Free-Air Temperature (unless otherwise noted)

Parameter		Test Conditions	Min.	Typ.	Max.	Unit
Input current	BLNK, BINH, INIT	V_I = 2.1 V			1	mA
High-level output current, I_{OH}	ECHO, OSC	V_{OH} = 5.5 V			100	uA
Low-level output voltage, V_{OL}	ECHO, OSC	I_{OL} = 1.6 mA			0.4	V
Transducer bias voltage		T_A = 25° C		200		V
Transducer output voltage (peak to peak)		T_A = 25° C		400		V
Number of cycles for XDCR output to reach 400 V		C = 500 pF			7	
Internal blanking interval				2.38*		ms
Frequency during 16-pulse transmit period	OSC output XMIT output			49.4* 49.4*		kHz
Frequency after 16-pulse transmit period	OSC output XMIT output			93.3* 0		kHz
Supply Current, I_{CC}	During transmit period After transmit period				2000 100	mA

* These typical values apply for a 420-kHz ceramic resonator.

Operation with Polaroid Electrostatic Transducer

There are two basic modes of operation for the 6500 Series Sonar ranging module: single-echo mode and multiple-echo mode. The application of power (V_{CC}), the activation of the Initiate (INIT) input, and the resulting transmit output, and the use of the Blanking Inhibit (BINH) input are basically the same for either mode of operation. After applying power (V_{CC}) a minimum of 5 milliseconds must elapse before the INIT input can be taken high. During this time, all internal circuitry is reset and the internal oscillator stabilizes. When INIT is taken high, drive to the Transducer XDCR output occurs. Sixteen pulses at 49.4 kilohertz with 400-volt amplitude will excite the transducer as transmission occurs. At the end of the 16 transmit pulses, a dc bias of 200 volts will remain on the transducer as recommended for optimum operation.

In order to eliminate ringing of the transducer from being detected as a return signal, the Receive (REC) input of the ranging control IC Is inhibited by internal blanking for 2.38 milliseconds after the initiate signal. If a reduced blanking time is desired, then the BINH input can be taken high to end the blanking of the Receive input anytime prior to internal blanking. This may be desirable to detect objects closer than 40cm corresponding to 2.38 milliseconds and may be done if transducer damping is sufficient so that ringing is not detected as a return signal.

In the single-echo mode of operation, all that must be done next is to wait for the return of the transmitted signal, traveling at approximately 340 m/s out and back. The returning signal is amplified and appears as a high-logic-level echo output. The time between INIT going high and the Echo (ECHO) output going high is proportional to the distance of the target from the transducer. If desired, the cycle can now be repeated by returning INIT to a low logic level and then taking it high when the next transmission is desired.

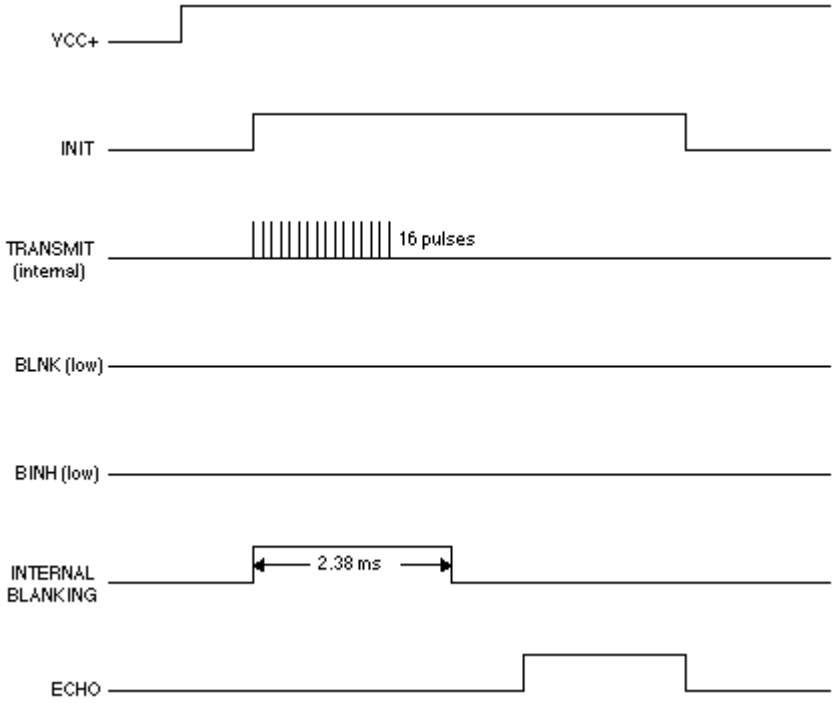

Figure B.3: Example of Single-Echo-Mode Cycle without Blanking Input

If there is more than one target and multiple echoes will be detected from a single transmission, then the cycle is slightly different. After receiving the first return signal which causes the ECHO output to go high, the Blanking (BLNK) input must be taken high then back low to reset the ECHO output for the next return signal. The blanking signal must be at least 0.44 milliseconds in duration to account for all 16 returning pulses from the first target and allow for internal delay times. This corresponds to the two targets being at least 8cm apart.

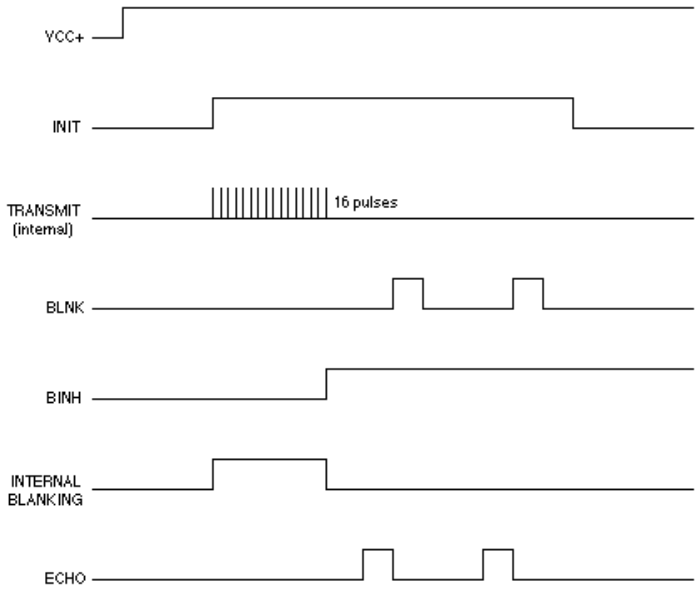

Figure B.4: Example of Multiple-Echo-Mode Cycle with Blanking Input

The Polaroid Sensors

The transducer used with this module is the instrument-grade Polaroid electrostatic transducer, which acts as a speaker in the transmit mode and a microphone in the receive mode. The transducer (shown on the next two figures) is 3.8cm in diameter and consists of a 3-millimeter gold-plated foil stretched over a concentrically grooved aluminum disc.

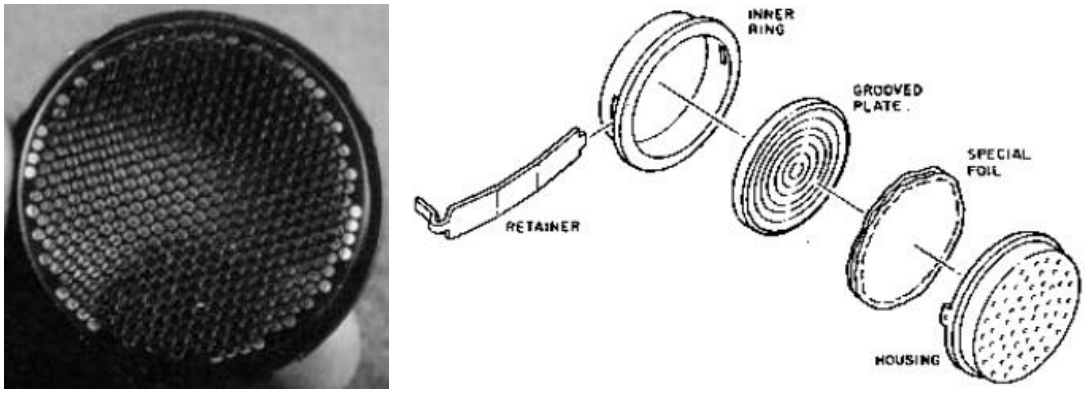

Figure B.5: Polaroid electrostatic transducer

The foil, electrically insulated yet bonded closely to the metallic backplate, forms a capacitor. The foil is the moving element in the transducer that converts electrical energy into sound and the returning echo into electrical energy. The diameter of the transducer determines its directional sensitivity. The Polaroid unit is very directional, as indicated in the graph of acoustical signal strength shown in figure 16.

Appendix C: Camera Datasheets

Description of the Sony EVI-D31 camera

Color pan-tilt-zoom camera, high speed wide range pan tilt head, integrated 12X high speed auto focus zoom lens, auto tracking and motion detection, fully controllable remotely via RS-232C/VISCA, infrared remote commander supplied.

Features

AT (Auto Tracking) Mode:

AT is a function which continually extracts a subject that the user pre-defines. After picking up pixels of similar color and brightness around the selected subject, EVI-D31 extracts the target by using the subject model based on light reflection and nonlinear camera processing. There are four modes for pre-defining the subject.

AT-PAN/TILT:

This function follows the moving subject automatically by controlling the pan & tilt motors without the use of special sensors.

Auto Zoom:

This function automatically controls the zoom lens to ensure that the size of the subject remains constant on the screen.

Auto Exposure:

The EVI-D31 employs the auto exposure and advanced backlight compensation systems to ensure that the subject remains bright even in harsh backlight conditions. Because the subject position is know a comparison can be made between its brightness and that of the background and the camera subsequently adjusted to compensate for the conditions.

MD (Motion Detector) Mode:

MD basically detects the difference between the initial reference image and the current image. The conventional technique employed in MD uses only the brightness of the video signal. The EVI-D31 uses both the brightness and color which enables even an object of the same brightness as the background to be detected.

Highlights

- High Speed, Wide Range Pan/tilter
- X12 Optical Zoom, High Speed Auto-Focus Lens
- 6 Position Preset

- Auto Tracking/Motion Detector
- RS232C Serial Control
- IR remote Commander
- Time, Date Generator

Specifications

Video Signal: PAL

Image Sensor: 1/3" IT Color CCD

Effective Pixels: 752 (H) x 585 (V)

H. Resolution: 450 TV lines

V. Resolution: 400 TV lines

Lens: X12 Power Zoom, f = 5.4 to 64.8mm, F1.8 to F2.7

Horizontal Angle of View: 4.3° (tele end) to 48.8° (wide end)

Vertical Angle of View: 3.2° (tele end) to 37.6° (wide end)

Shortest Subject Distance: 10mm (WIDE end), 800mm (TELE end)

Min. Illumination: 7 lux (F1.8)

Illumination Range: 7 to 100,000 lux

Auto Exposure: Auto Iris, AGC

Shutter Speed: 1/50 to 1/10,000 (VISCA™ control)

Gain: Auto/ Manual (VISCA™ Control)

White Balance: TTL Auto Tracing/ One Push Hold, Indoor Preset, Outdoor Preset (VISCA control)

S/N Ratio: more than 48dB

Pan/Tilt: Horizontal ± 100° (Max speed 80°/ sec), Vertical ± 25° (Max speed 50°/ sec)

Video Output: RCA pin jack, 1Vp-p, 75ohm unbalanced

S Video Output: 4 pin mini DIN

Audio Output: RCA pin jack (monaural), Rated output 327mV, Output impedance less than 2.2 kΩ

Control Terminal: RS232C, 8 pin mini DIN, 9600bps, Data 8 bit, Stop 1 bit

Microphone Input Terminal: Mini jack (monaural) (diameter 3.5), Rated input 0.775mV DC3V for low impedance microphone, Input Impedance more than 10kOhms

Power Terminal: DC IN 13.5V (EIAJ unified polarity type)

Power Requirements: DC12 to 14V

Power Consumption: 11W

Operating Temperature: 0 to 40°C

Storage Temperature: -20 to 60°C

Dimensions(W/H/D): Camera 142 x 109 x 164 mm; Remote Commander 56 x 26 x 210 mm

Weight: Camera 1200g, Remote Commander 109g

VISCA protocol command list summary

Communication Specifications

◇ Communication specification (RS-232C)

- Communication speed: 9600 bps
- Start bit : 1
- Stop bit : 1
- Data bits : 8
- Parity : None
- MSB first

◇ Communication protocol

● Communication from the controller

Communication is started by header which comprises sender's address and receiver's address followed by message and ended by terminator. The message part comprises communication mode (2 bytes), category code (2 bytes) and parameters. The maximum length of the message is 14 bytes. The terminator is fixed to FFH and the controller should check the value to terminate communication. The bit 15 should be 0 in the message part.

Header	Com-mode	Category code	Message	Terminator
8x	01	rr	.. ..	FF

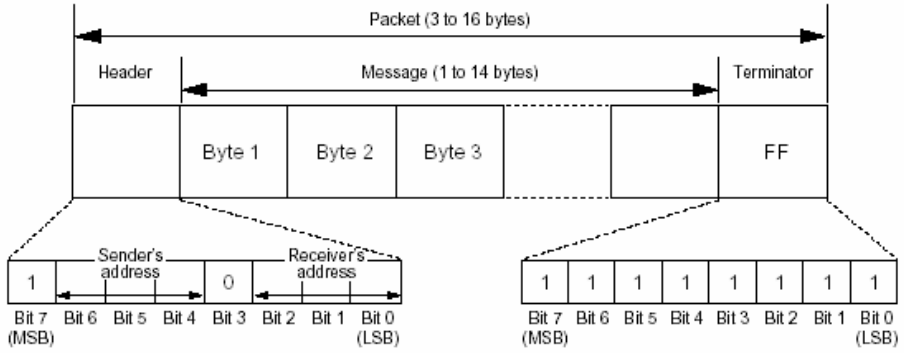

Fig. 3

Header	: Signifies the start of the communication and comprises the sender's address and receiver's address. Since the address of the controller is fixed to 0, header is 8x in which x is the receiver's address. (The value of x should be from 1 to 7) In case of broad cast, the header should be 88H.
Com-mode	: Code which specifies the category of command.
Control command	: 01H
Information request command	: 09H
Net-keeping command	: 00H
Category	: Code which roughly specifies the category the command is applicable.
Main message	: Part between header and terminator. 14 bytes maximum. Comprises command and parameter if any.
Terminator	: Code which signifies the end of communication. Fixed to FFH.

EVI-D30/D31 Command List (1/3)

Command Set	Command	VISCA™ Packet	Comments
AddressSet	broadcast	88 30 01 FF	Send Address_set command and IF_clear command before starting communication. Broadcast.
IF_Clear	broadcast	88 01 00 01 FF	
CommandCancel		8x 2Z FF	Z is a socket number. 0 or 1
CAM_Power	On	8x 01 04 00 02 FF	When Camera main power is on, camera can be changed to Power Save Mode.
	Off	8x 01 04 00 03 FF	
CAM_Zoom	Stop	8x 01 04 07 00 FF	
	Tele (Standard)	8x 01 04 07 02 FF	
	Wide (Standard)	8x 01 04 07 03 FF	
	Tele (Variable)	8x 01 04 07 2Z FF	Z: Speed Parameter, 2 (Low) to 7 (High)
	Wide (Variable)	8x 01 04 07 3Z FF	
	Direct	8x 01 04 47 0Z 0Z 0Z 0Z FF	ZZZZ: Zoom Data, 0000 (Wide) to 03FF (Tele)
CAM_Focus	Stop	8x 01 04 08 00 FF	Focus control. When adjust the focus, change the mode to Manual the send Far/Near or Direct command.
	Far	8x 01 04 08 02 FF	
	Near	8x 01 04 08 03 FF	
	Auto focus on	8x 01 04 38 02 FF	
	Manual focus on	8x 01 04 38 03 FF	
	Auto/Manual	8x 01 04 38 10 FF	
	Direct	8x 01 04 48 0Z 0Z 0Z 0Z FF	ZZZZ: Focus Data. Infinity = 1000, close = 9FFF
CAM_WB	Auto	8x 01 04 35 00 FF	White Balance Setting. Auto: Trace the light source automatically. Indoor/Outdoor: Fixed at Factory. Pull-in to White with a Trigger then hold the data until next Trigger coming
	Indoor mode	8x 01 04 35 01 FF	
	Outdoor mode	8x 01 04 35 02 FF	
	OnePush mode	8x 01 04 35 03 FF	
	OnePush trigger	8x 01 04 10 05 FF	
CAM_AE	Full Auto	8x 01 04 39 00 FF	Auto Exposure Mode
	Manual	8x 01 04 39 03 FF	Iris, Shutter and Gain can be set individually.
	Shutter priority	8x 01 04 39 0A FF	Shutter fixed Auto Exposure Mode. Shutter speed can be selected.
	Iris priority	8x 01 04 39 0B FF	Iris fixed Auto Exposure Mode. Iris can be selected.
	Bright mode	8x 01 04 39 0D FF	Fixed Exposure Mode. When turning on to Bright Mode, Iris, Gain and Shutter at the time then increase or decrease 3 dB/ step using UP/DOWN command.
CAM_Bright	Reset	8x 01 04 0D 00 FF	
	Up	8x 01 04 0D 02 FF	
	Down	8x 01 04 0D 03 FF	
CAM_Shutter	Reset	8x 01 04 0A 00 FF	Electronic Shutter Setting. Enable on AE_Manual, Shutter_Priority
	Up	8x 01 04 0A 02 FF	
	Down	8x 01 04 0A 03 FF	
	Direct	8x 01 04 4A 0Z 0Z 0Z 0Z FF	ZZZZ = 0000: 1/60, 001B: 1/10000 second
CAM_Iris	Reset	8x 01 04 0B 00 FF	Iris Setting. Enable on AE_Manual or Iris_Priority
	Up	8x 01 04 0B 02 FF	
	Down	8x 01 04 0B 03 FF	
	Direct	8x 01 04 4B 0Z 0Z 0Z 0Z FF	ZZZZ = 0000: close to 0011: F1.8
CAM_Gain	Reset	8x 01 04 0C 00 FF	Gain Setting. Enable on AE_Manual only
	Up	8x 01 04 0C 02 FF	
	Down	8x 01 04 0C 03 FF	
	Direct	8x 01 04 4C 0Z 0Z 0Z 0Z FF	ZZZZ = 0001: 0 dB to 0007: +18 dB
CAM_Backlight	On	8x 01 04 33 02 FF	Back light compensation Gain-up to 6 dB max.
	Off	8x 01 04 33 03 FF	

EVI-D30/D31 Command List (2/3)

Command Set	Command	VISCA™ Packet	Comments
CAM_Memory	Reset	8x 01 04 3F 00 0Z FF	Preset memory for memorize camera condition.
	Set	8x 01 04 3F 01 0Z FF	Z = 0 to 5, 6 positions
	Recall	8x 01 04 3F 02 0Z FF	
CAM_KeyLock	Off	8x 01 04 17 00 FF	Enable/Disable for RS-232C and key control.
	On	8x 01 04 17 02 FF	
IR_Receive	On	8x 01 06 08 02 FF	Enable/Disable for IR remote commander.
	Off	8x 01 06 08 03 FF	
	On/Off	8x 01 06 08 10 FF	
IR_ReceiveReturn	On	8x 01 7D 01 03 00 00 FF	Send replies what command received from IR Commander.
	Off	8x 01 7D 01 13 00 00 FF	
Wide_conLensSet		8x 01 07 26 00 0Z FF	Automatic Target Trace ability Compensation when a wide conversion lens installed. Z = 0: No Conversion to 7: X0.6 Conversion
Pan-tiltDrive	Up	8x 01 06 01 VV WW 03 01 FF	VV: pan speed 01 to 18,
	Down	8x 01 06 01 VV WW 03 02 FF	WW: tilt speed 01 to 14
	Left	8x 01 06 01 VV WW 01 03 FF	YYYY: pan position: approx. FC90 to 0370
	Right	8x 01 06 01 VV WW 02 03 FF	(center 0000)
	UpLeft	8x 01 06 01 VV WW 01 01 FF	ZZZZ: tilt position: approx. FED4 to 012C
	UpRight	8x 01 06 01 VV WW 02 01 FF	(center 0000)
	DownLeft	8x 01 06 01 VV WW 01 02 FF	
	DownRight	8x 01 06 01 VV WW 02 02 FF	
	Stop	8x 01 06 01 VV WW 03 03 FF	
	Absolute position	8x 01 06 02 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	Absolute Position Drive
	Relative position	8x 01 06 03 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	Relative Position Drive. Set the relative coordinates between current position to the the target position.
	Home	8x 01 06 04 FF	
	Reset	8x 01 06 05 FF	Pan/Tilt Initialize command
Pan-tiltLimitSet	Limit set	8x 01 06 07 00 0W 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	Pan/Tilt limit set YYYY: pan position FC90 to 0370 (center 0000)
	Limit clear	8x 01 06 07 01 0W 07 0F 0F 0F 07 0F 0F 0F FF	ZZZZ: tilt position FED4 to 012C (center 0000) W: 1 UpRight, 0 DownLeft
Datascreen	On	8x 01 06 06 02 FF	On screen Data Display ON/OFF
	Off	8x 01 06 06 03 FF	
	On/Off	8x 01 06 06 10 FF	
AT_Mode	On	8x 01 07 01 02 FF	Target Tracking Mode ON/OFF
	Off	8x 01 07 01 03 FF	
	On/Off	8x 01 07 01 10 FF	
AT_AE	On	8x 01 07 02 02 FF	Auto Exposure for the target
	Off	8x 01 07 02 03 FF	
	On/Off	8x 01 07 02 10 FF	
AT_AutoZoom	On	8x 01 07 03 02 FF	Automatic Zooming for the target
	Off	8x 01 07 03 03 FF	
	On/Off	8x 01 07 03 10 FF	
AT/MD_Frame _Display	On	8x 01 07 04 02 FF	Sensing Frame Display ON/OFF
	Off	8x 01 07 04 03 FF	
	On/Off	8x 01 07 04 10 FF	

EVI-D30/D31 Command List (3/3)

Command Set	Command	VISCA™ Packet	Comments
AT_Offset	On	8x 01 07 05 02 FF	Shifting the Sensing Frame for AT For Shifting use Pan/Tilt Drive Command
	Off	8x 01 07 05 03 FF	
	On/Off	8x 01 07 05 10 FF	
AT/MD_Start/Stop	Start/Stop	8x 01 07 06 10 FF	Tracking or Detecting Start/Stop
AT_Chase	Chase1	8x 01 07 07 00 FF	Select a Tracking Mode
	Chase2	8x 01 07 07 01 FF	
	Chase3	8x 01 07 07 02 FF	
	Chase1/2/3	8x 01 07 07 10 FF	
AT_Entry	Entry1	8x 01 07 15 00 FF	Select target study mode for AT
	Entry2	8x 01 07 15 01 FF	
	Entry3	8x 01 07 15 02 FF	
	Entry4	8x 01 07 15 03 FF	
MD_Mode	On	8x 01 07 08 02 FF	Motion Detector Mode ON/OFF
	Off	8x 01 07 08 03 FF	
	On/Off	8x 01 07 08 10 FF	
MD_Frame	Setting	8x 01 07 09 FF	Detecting Area Set (Size or Position)
MD_Detect	Frame1/2/1 or 2	8x 01 07 0A 10 FF	Select Detecting Frame (1 or 2 or 1 + 2)
AT_LostInfo		8x 01 06 20 07 20 FF	Reply a completion when the camera lost the target in AT mode.
MD_LostInfo		8x 01 06 20 07 21 FF	Reply a completion when the camera detected a motion of image in MD mode.
MD_Adjust	Y Level	8x 01 07 0B 00 0Z FF	Set Detecting Condition Z = 0 to F
	Hue Level	8x 01 07 0C 00 0Z FF	
	Size	8x 01 07 0D 00 0Z FF	
	Display time	8x 01 07 0F 00 0Z FF	
	Refresh mode1	8x 01 07 10 00 FF	
	Refresh mode2	8x 01 07 10 01 FF	
	Refresh mode3	8x 01 07 10 02 FF	
	Refresh time	8x 01 07 0B 00 0Z FF	Z = 0 to F
Measure_Mode1	On	8x 01 07 27 02 FF	Target Condition Measure Mode for More Accurate Setting for Motion Detector.
	Off	8x 01 07 27 03 FF	
	On/Off	8x 01 07 27 10 FF	
Measure_Mode2	On	8x 01 07 28 02 FF	
	Off	8x 01 07 28 03 FF	
	On/Off	8x 01 07 28 10 FF	

Inquiry Command (1/2)

Inquiry	Packet Inq	Packet Reply	Description
CAM_PowerInq	8x 09 04 00 FF	Y0 50 02 FF	On
		Y0 50 03 FF	Off
CAM_ZoomPosInq	8x 09 04 47 FF	Y0 50 0Z 0Z 0Z 0Z FF	ZZZZ: position
CAM_FocusAFModeInq	8x 09 04 38 FF	Y0 50 02 FF	Auto
		Y0 50 03 FF	Manual
CAM_FocusPosInq	8x 09 04 48 FF	Y0 50 0Z 0Z 0Z 0Z FF	ZZZZ: position
CAM_WBModeInq	8x 09 04 35 FF	Y0 50 00 FF	Auto
		Y0 50 01 FF	Indoor mode
		Y0 50 02 FF	Outdoor mode
		Y0 50 03 FF	OnePush mode
CAM_AEModeInq	8x 09 04 39 FF	Y0 50 00 FF	Full Auto
		Y0 50 03 FF	Manual
		Y0 50 0A FF	Shutter priority
		Y0 50 0B FF	Iris priority
		Y0 50 0D FF	Bright mode
CAM_ShutterPosInq	8x 09 04 4A FF	Y0 50 0Z 0Z 0Z 0Z FF	ZZZZ: position
CAM_IrisPosInq	8x 09 04 4B FF	Y0 50 0Z 0Z 0Z 0Z FF	ZZZZ: position
CAM_GainPosInq	8x 09 04 4C FF	Y0 50 0Z 0Z 0Z 0Z FF	ZZZZ: position
CAM_Backlight	8x 09 04 33 FF	Y0 50 02 FF	On
Mode Inq		Y0 50 03 FF	Off
CAM_MemoryInq	8x 09 04 3F FF	Y0 50 0Z FF	Z: 0 to 5
CAM_KeyLockInq	8x 09 04 17 FF	Y0 50 00 FF	Off
		Y0 50 02 FF	On
CAM_IDInq	8x 09 04 22 FF	Y0 50 0Z 0Z FF	ZZ: ID
VideoSystemInq	8x 09 06 23 FF	Y0 50 00 FF	NTSC
		Y0 50 01 FF	PAL
Wide_conLensInq	8x 09 07 26 FF	Y0 50 00 0Z FF	Z: lens No.
Pan-tiltModeInq	8x 09 06 10 FF	Y0 50 ZZ ZZ FF	ZZZZ: status
Pan-tiltMaxSpeedInq	8x 09 06 11 FF	Y0 50 WW ZZ FF	WW: pan, ZZ: tilt
Pan-tiltPosInq	8x 09 06 12 FF	Y0 50 0W 0W 0W 0W	WWWW: pan
		0Z 0Z 0Z 0Z FF	ZZZZ: tilt
DatascreenInq	8x 09 06 06 FF	Y0 50 02 FF	On
		Y0 50 03 FF	Off
AT/MD_ModeInq	8x 09 07 22 FF	Y0 50 00 FF	Normal mode
		Y0 50 01 FF	AT mode
		Y0 50 02 FF	MD mode
AT_ModeInq	8x 09 07 23 FF	Y0 50 ZZ ZZ FF	ZZ: status
AT_EntryInq	8x 09 07 15 FF	Y0 50 00 FF	entry mode 1
		Y0 50 01 FF	entry mode 2
		Y0 50 02 FF	entry mode 3
		Y0 50 03 FF	entry mode 4
MD_ModeInq	8x 09 07 24 FF	Y0 50 ZZ ZZ FF	ZZ: status
AT_ObjectPosInq	8x 09 07 20 FF	Y0 50 VV WW 0Z FF	Dividing a screen by 48 × 30 pixels, Return the center position of the detecting Frame. VV: X, WW: Y X 04-2A, Y 03-1B Z: status
MD_ObjectPosInq	8x 09 07 21 FF	Y0 50 VV WW 0Z FF	
MD_Y LevelInq	8x 09 07 0B FF	Y0 50 00 0Z FF	Z: 0 to F

Inquiry Command (2/2)

Inquiry	Packet Inq	Packet Reply	Description
MD_Hue LevelInq	8x 09 07 0C FF	Y0 50 00 0Z FF	Z: 0 to F
MD_SizeInq	8x 09 07 0D FF	Y0 50 00 0Z FF	Z: 0 to F
MD_Disp.TimeInq	8x 09 07 0F FF	Y0 50 00 0Z FF	Z: 0 to F
MD_Ref.ModeInq	8x 09 07 10 FF	Y0 50 00 FF	Refresh mode 1
		Y0 50 01 FF	Refresh mode 2
		Y0 50 02 FF	Refresh mode 3
MD_Ref.TimeInq	8x 09 07 11 FF	Y0 50 00 0Z FF	Z: 0 to F
IR_ReceiveReturn		Y0 07 7D 01 04 00 FF	Power ON/OFF
		Y0 07 7D 01 04 07 FF	Zoom tele/Wide
		Y0 07 7D 01 04 38 FF	AF ON/OFF
		Y0 07 7D 01 04 33 FF	CAM_Backlight
		Y0 07 7D 01 04 3F FF	CAM_Memory
		Y0 07 7D 01 06 01 FF	Pan-tiltDrive
		Y0 07 7D 01 07 23 FF	AT_Mode ON/OFF
		Y0 07 7D 01 07 24 FF	MD_Mode ON/OFF

Appendix D: Classical path planning methods

Vertex Graph Path Planning

Vertex Graph path planning bases on a map that models all obstacles in the environment geometrically. Prior to the path planning the obstacles are expanded by the radius of the robot (assuming circular shape of the platform) plus a security distance, the robot is then considered a point. All possible collision free paths are constructed by connecting the vertices of the expanded obstacles that are of free line of sight. These paths are then searched for an optimal path using the desired optimisation criteria using a standard search algorithm.

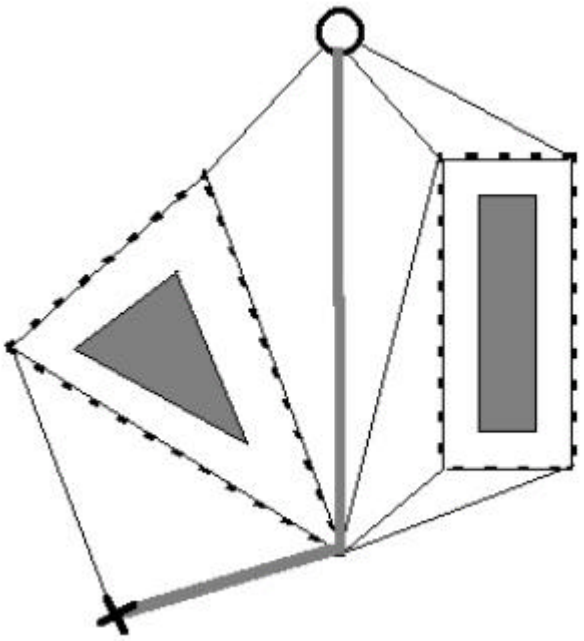

Figure D.1: Vertex Graph Navigation

The path from the start position (circle) to the goal position (cross) is determined considering two obstacles. All possible paths are shown in fine black lines, the solution path in a thick grey line.
A drawback of Vertex Graph Navigation is its limitation to optimisation criteria that are related to properties of the straight lines between the vertices, e.g. the shortest path. It has no mechanism to deal with unknown regions of the environment. Used for reactive (local) navigation the entire path from the current location to the goal has to be re-planned continuously, which would waste computing resources. Vertex Graph path planning is therefore more suitable for Global Navigation.

Free Space Path Planning

Free space navigation considers free space rather than obstacles to determine the path for a vehicle. Free space is modelled as convex polygons, generalised cones or a combination called "mixed space. All these methods have in common that a set of possible paths is constructed linking the centres of passable free space corridors. This set is searched for an optimal solution using search algorithms as in the Vertex Graph method.

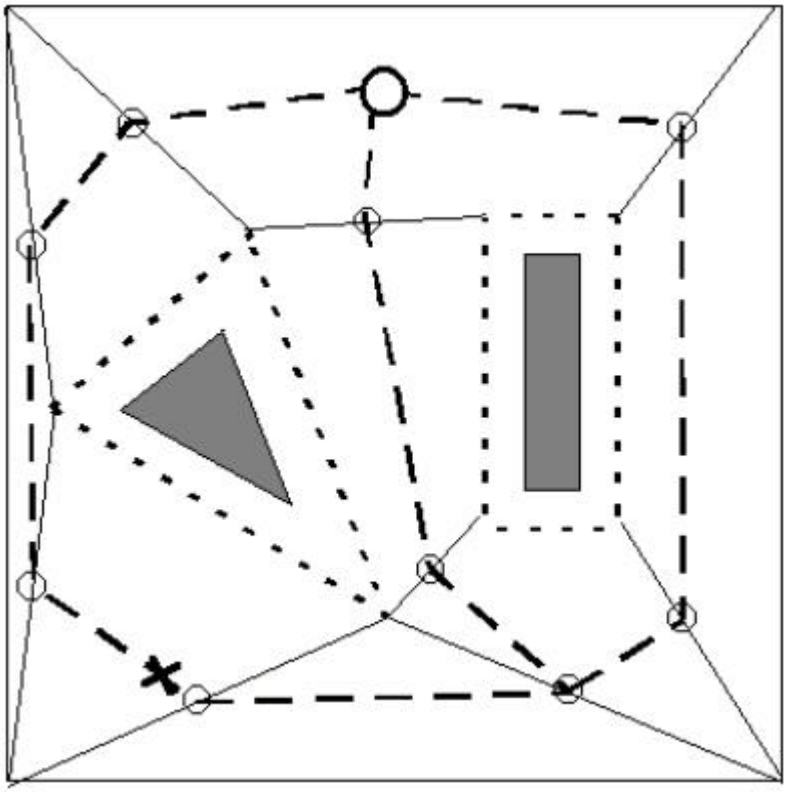

Figure D.2: Free Space Navigation

All possible paths are shown in broken lines.

The main drawback of this method is often referred to as the "too far problem", which means the solution path tends to be too conservative; the robot is keeping a distance from obstacles that might be larger than the specified security distance. Free space navigation is therefore limited to find the safest path. This is fatal in large environments populated with only a few obstacles where the solution path might be much longer than necessary. Furthermore free space navigation suffers the same limitations with respect to reactive navigation as Vertex Graph navigation.

Grid Based Navigation

This navigation technique uses a grid map or needs to superimpose a grid on the existing map. Each grid cell is marked as free or occupied. The obstacles are again expanded by the vehicles diameter plus a security distance. Each grid point can now be "four or eight connected" to its neighbour points, depending on the inclusion or exclusion of diagonal neighbours. The set of possible paths is now searched for an optimal path using one of the standard search algorithms.

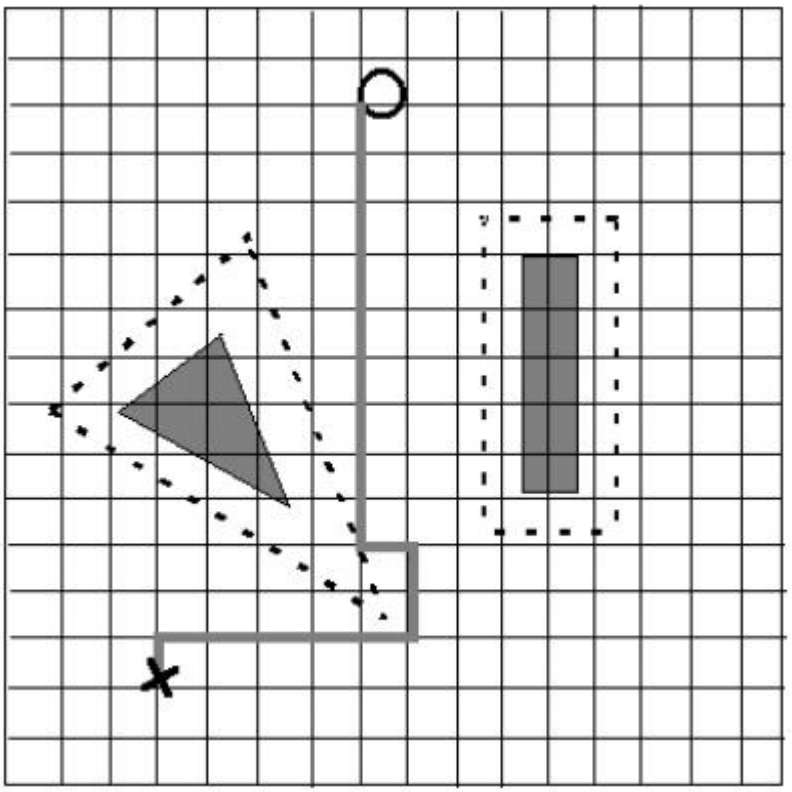

Figure D.3: Four Connected Grid Navigation

Obviously the resolution of the path is dependent on the resolution of the grid. On order to obtain a path that is not too conservative a relatively small grid size is necessary. This however leads to a very large set of possible paths and a need for powerful computing resources. Another drawback is the extra effort to build the grid if not already present in form of a grid-based map.

Distance Transforms

Distance Transforms are a novel approach of path planning offering some significant advantages. A grid is superimposed on the environment map and each cell is assigned a value that represents the distance from this cell to the cell in which the goal is. These distance values are calculated from the goal (=0) "flowing" around obstacles whose cells are assigned the value infinity until all cells are calculated. The solution path is then found as a sequence of cells going downhill in distance values from the start cell to the goal cell. If there is no downhill path from a cell than it can be concluded that there is no solution path, i.e. the goal is unreachable. If there are two neighbour cells with the same value, the two paths are equivalent.

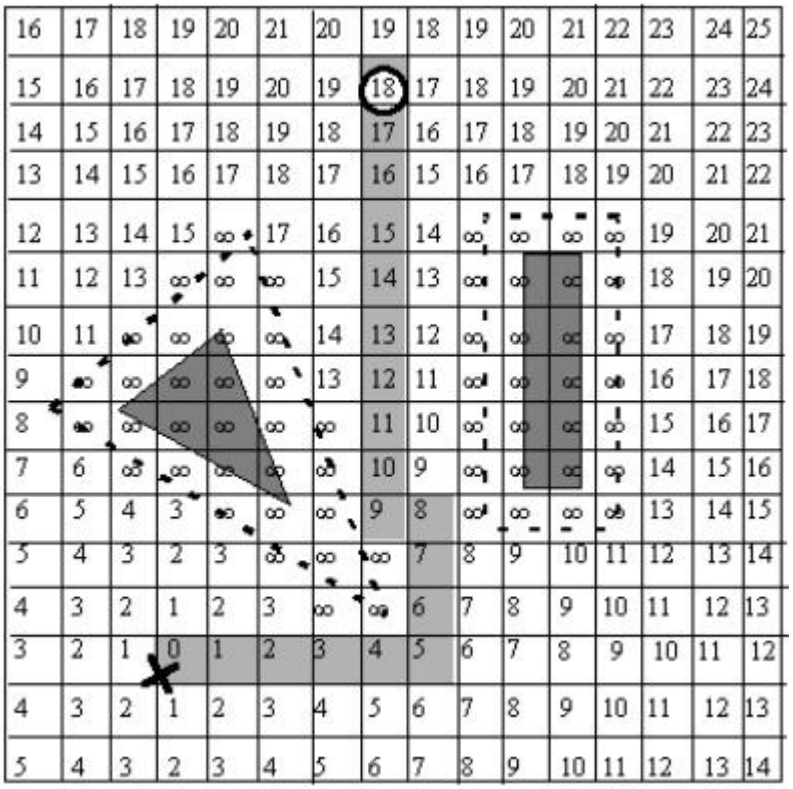

Figure D.4: Distance Transform Navigation

The main drawback of distance transforms is the large computing overhead for building the grid, if not already present, and calculating the distance transform values.

However it offers some major advantages: so is the optimal path known from every grid cell in the map, such that multiple robot systems are supported. Also multiple goals of same priority are easily considered; in this case the vehicle selects automatically the goal that can be reached with minimum cost.

Very important is that the distance transform values are not limited to reflect the Euclidean distance to the goal. It is straightforward to implement all kinds of cost functions, such that the distance transform values reflect directly the cost of the path from each cell to the goal. This way, path characteristics such as "conservative", "adventurous", etc. can be realised. Distance transform methods are furthermore very flexible in their implementation. They can be very accurate when considering not only perpendicular neighbours, but also diagonal neighbours with the distance calculated from the centre of the cells.

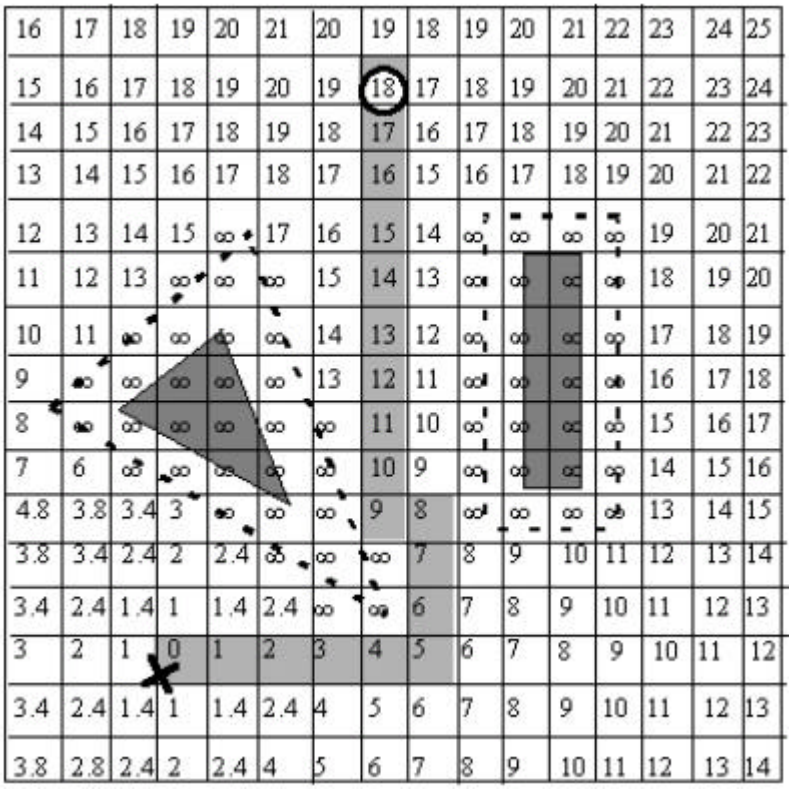

Figure D.5: Diagonal Distance Transform

Of considerable importance, especially with large cell size, is the transition from the solution cell sequence to the actual solution path. This is a purely geometric step and can be done in many ways. The easiest method is to connect the centres of the cells. This introduces some conservativeness and can be replaced with more sophisticated methods.

Distance transform techniques are not suitable for reactive navigation due to their large computational overhead. However they are a flexible and efficient method for off-line pre-planning of a vehicle's path (Global Navigation).

Heuristic Navigation

Heuristic navigation does not use an environment model (map) but can use sensor information directly. The robot behaviour consists of simple rules, e.g. minimising the current distance to the goal, minimise the deviation angle form the current moving direction to the straight line to the goal, etc.
Heuristic navigation is very limited in the situations it can solve but might be a cost effective fast alternative for some simple applications.

Stream Field Methods

The stream field approach consists - according to [24] - to construct the path guiding the robot from its initial position to its final position by computing the internal path(s) of a continuous velocity field of a fluid flow from a source at the initial position to a sink at the destination. These techniques implicate the calculation of the dynamic equations for a certain viscid or inviscid fluid flow. This means a performant computer infrastructure is needed to process all the data in real time. On the other hand, these methods have the advantage that they can take direct use of the different calculation acceleration techniques known in the field of numerical aerodynamics.

Appendix E: Source Code

Here, the source codes of the Matlab programs for the recursive algorithm and the robot control program will be inserted. As this latter is rather large (about 250 pages), I'll print two pages on one side.

View publication stats