UNIVERSITY OF TRIESTE
Administrative Location of Doctoral Program

DOCTORATE IN INFORMATION ENGINEERING
XIX CYCLE

Methods and Devices for Mobile Robot Navigation and Mapping in Unstructured Environments

DOCTORAL CANDIDATE
Kristijan Lenac

To my family.

Contents

Chapter 1

Introduction

The work described in this thesis has been carried out in the context of the exploration of an unknown environment with a mobile robot. This chapter provides background information and outlines some of the research areas concerning the work carried out in this thesis. It then explains which challenging problems have been tackled and briefly introduces the contributions, which are restated later in more depth at the end of the thesis chapters.

1.1 Background and motivation

1.1.1 Mobile robot navigation and mapping

The goal of an autonomous mobile robot is to operate without human intervention in a real world environment which has not been specifically engineered for the robot. Different robot architectures have been developed that range from purely reactive robots which do not keep an internal state to layered architectures with a deliberative layer planning on actions. With the exception of the purely reactive robot which responds to the environment based only on the sensory input at that moment, all other robots usually need some sort of a spatial model of the surrounding physical environment in order to execute meaningful tasks, i.e. a map. In fact it is rather difficult to imagine a robot that is truly autonomous without being capable of acquiring a model of its environment. This model can be built by the robot exploring the environment and registering the data collected with the sensors over time. The sensory data is organized into a consistent representation in one of many types of maps, depending on the characteristics of the sensory data and a method used to collect these data. Some maps attempt to represent the space in absolute metric terms, and others try to represent it using shapes, or even creating graphs that represent spaces and the connections between them.

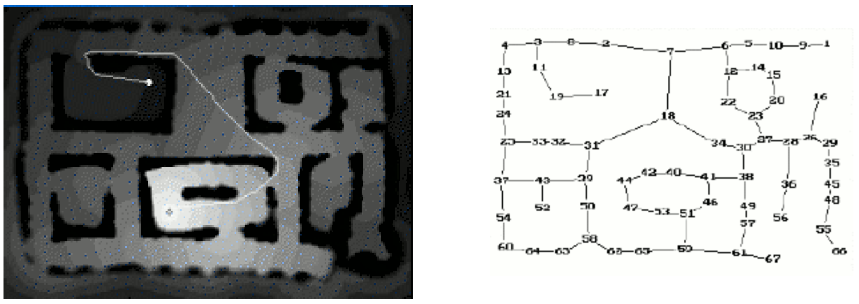


Figure 1.1: Grid based map (left) and the extracted topological map (Thrun, Bucken, 1994).

Robot navigation is concerned with the problem of taking a robot from one point to another of the environment. With the help of the mapping and localization methods the basic navigation tasks consist of path planning, obstacles avoidance and robot control.

Robotic Mapping Problem

Essentially robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots. It is considered as one of the most important problems in building truly autonomous mobile robots. There has been a significant progress in this area in the last decades, nevertheless it still poses great challenges. Currently there exist robust methods for mapping environments that are static, structured, and of limited size. Mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.

S.Thrun in his comprehensive introduction into the field of robotic mapping [64] presents five aspects of the robotic mapping that make it a challenging problem.

1. Measurement noise

   Modeling problems, such as robotic mapping, are usually relatively easy to solve if the noise in different measurements is statistically independent. If this were the case, a robot could simply take more and more measurements to cancel out the effects of the noise. Unfortunately, in robotic mapping, the measurement errors are statistically dependent. This is because errors in control accumulate over time, and they affect the way future sensor measurements are interpreted. Accommodating such systematic errors is key to building maps successfully, and it is also a key complicating factor in robotic mapping. Many existing mapping algorithms are therefore surprisingly complex, both from a mathematical and from an implementation point of view [64].

2. High dimensionality of the entities that are being mapped.

   The modeling and the description of a simple environment like the home that surround us requires a huge amount of data. If one confines oneself to the description of major topological entities, such as corridors, intersections, rooms and doors, a few dozen numbers might suffice. A detailed two-dimensional floor plan, which is an equally common representation of robotic maps, often requires thousands of numbers. But a detailed 3D visual map of a building (or of an ocean floor) may easily require millions of numbers. From a statistical point of view, each such number is a dimension of the underlying estimation problem. Thus, the mapping problem can be extremely high dimensional [64].

3. Correspondence problem

   A third and possibly the hardest problem in robotic mapping is the correspondence problem, also known as the data association problem. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world. The correspondence problem is difficult, since the number of possible hypotheses can grow exponentially over time.

4. Dynamism of robot environments

   Fourth, environments change over time. Some changes may be relatively slow, such as the change of appearance of a tree across different seasons, or the structural changes that most office buildings are subjected to over time. Others are faster, such as the change of door status or the location of furniture items, such as chairs. Even faster may be the change of location of other agents in the environment, such as cars or people. The dynamism of robot environments creates a big challenge, since it adds yet another way in which seemingly inconsistent sensor measurements can be explained. To see, imagine a robot facing a closed door that previously was modeled as open. Such an observation may be explained by two hypotheses, namely that the door status changed, or that the robot is not where it believes to be. Unfortunately, there are almost no mapping algorithms that can learn meaningful maps of dynamic environments. Instead, the predominant paradigm relies on a static world assumption, in which the robot is the only time-variant quantity (and everything else that moves is just noise). Consequently, most techniques are only applied in relatively short time windows, during which the respective environments are static [64].

5. Robotic exploration in real time

   A fifth and final challenge arises from the fact that robots must choose their way during mapping. The task of generating robot motion in the pursuit

> of building a map is commonly referred to as robotic exploration. While optimal robot motion is relatively well-understood in fully modeled environments, exploring robots have to cope with partial and incomplete models. Hence, any viable exploration strategy has to be able to accommodate contingencies and surprises that might arise during map acquisition. For this reason, exploration is a challenging planning problem, which is often solved sub-optimally via simple heuristics. When choosing where to move, various quantities have to be traded off: the expected gain in map information, the time and energy it takes to gain this information, the possible loss of pose information along the way, and so on. Furthermore, the underlying map estimation technique must be able to generate maps in real-time, which is an important restriction that rules out many existing approaches [64].

Today, mapping is largely considered the most difficult perceptual problem in robotics. Progress in robot mapping is bound to impact a much broader range of related perceptual problems, such as sensor based manipulation and interaction with people.

Sensors

A robot perceives the outside world through it's sensors. Using sensors it is able to acquire a map and to localize itself on the map. The most common sensors used for these tasks are range finders using sonar, laser, and infrared technology, cameras, tactile sensors, devices for dead reckoning like wheel encoders and inertial sensors, active beacons, compasses, and Global Positioning Systems (GPS). However, all these sensors are subject to errors, often referred to as measurement noise. More importantly, most robot sensors are subject to strict range limitations. For example, light and sound cannot penetrate walls. These range limitations makes it necessary for a robot to navigate through its environment when building a map. The motion commands (controls) issued during environment exploration carry important information for building maps, since they convey information about the locations at which different sensor measurements were taken. Robot motion is also subject to errors, and the controls alone are therefore insufficient to determine a robots pose (location and orientation) relative to its environment.

Localization

All mapping methods must consider how the positional information of the robot is maintained, as all mapping methods depend on a consistency and accuracy of positional information. The primary method of localization on most robots is the use of dead-reckoning. Dead-reckoning uses the wheel encoders of the robot, which record how many revolutions each of the wheels has gone forward and backward. This information is used to compute a relative position of the robot. This method does not reference the outside world in any way, hence the name dead-reckoning. Unfortunately, the wheel encoders on any robot are prone to error, and these errors are compounded over time if not corrected. Thus, the robot's estimations of its own position monotonically worsen as time passes in a system that uses only dead-reckoning. A robot that cannot reference the outside world to correct its estimation of its own position will inevitably become so inaccurate that maps constructed on the faulty estimation will be virtually useless [16].

Because the accurate estimation of position is so important for mapping tasks, every robotic mapping system that needs accuracy in the long term must use some method of localization, using information from the outside world to make position estimates more accurate. Ideally, some system like GPS could be used to determine the position of a robot to a very high-degree of absolute accuracy, but unfortunately such systems require an open view of the satellites and do not work indoors. Many methods of localization have been implemented.

SLAM

If the robot's pose is known all the time then it is simple to build a map. If we rely on an a-priory available map then there are different efficient algorithms for determining the robot's pose. On the other hand the problem of building a map while exploring an unknown territory, is usually known as Simultaneous Localization and Mapping (SLAM). Many algorithms for map building address the localization issue in isolation and not while the map is being constructed since the combined localization and mapping is not as straightforward as it would seem. In essence, both the robot localization and the map are uncertain, and by focusing just on one the other introduces systematic noise. A variety of approaches have been proposed for representing the uncertainty inherent to sensor data and robot motion. The most popular solutions to SLAM are probabilistic techniques. Many proposed techniques are based on the Kalman filter. The Kalman filter provides the optimal linear recursive solution to SLAM when certain assumptions hold, such as perfect data association, linear motion and measurement models, and Gaussian error models. A popular nonlinear alternative is the FastSLAM [48]. Estimating the robot's pose and the map at the same time has the property that both the measurement and the control noise are independent with regards to the properties that are being estimated (the state).

In order to integrate an existing localization technique in a SLAM framework the former should be able to provide multiple position estimates together with a measure of confidence in the position estimate, usually in the form of a covariance matrix.

1.1.2 Embedded systems and real-time operation

The computer that controls the robot is usually hidden somewhere inside its body. It does not interface with the outside world through familiar computer peripheral devices such as a keyboard, a mouse and a graphic display. Instead, the interaction with the outside world is through interfaces such as sensors, actuators and specific communication links. Depending on the robot architecture, one or more of these embedded computers are in charge of the robot and its subsystems. Real-time embedded systems operate in constrained environments in which computer memory and processing power are limited. They often need to provide their services within strict time deadlines to their users and to the surrounding world. It is these memory, speed and timing constraints that dictate the use of real-time operating systems in embedded software.

Real-time problems have not been deeply investigated for robot architectures. Reactivity is typically viewed simply as very fast response, regardless of the effective timing constraints. From one point of view, this results in sometimes unnecessary responsiveness; from another, since there is no guarantee on the response time, sometimes the reaction may not be sufficiently fast.

A common conceptual distinction of real-time operating systems (RTOS in the following) is in hard and soft real time systems. Hard real-time systems are used when it is imperative that an event is reacted to within a strict deadline. A soft real-time system on the other hand can tolerate some delays decreasing the service quality (e.g., dropping frames while displaying a video). Applications discussed in this thesis are best served by a hard real time operating system, so the term RTOS refers to a hard real-time operating system in this thesis.

General operating systems[*] are full of sources of non deterministic behavior (cache, interrupts, DMA, virtual memory). In a real-time operating system the system services consume only known and expected amounts of time. All the system primitives must have a defined maximum execution time so not to introduce non deterministic delays. Interrupts must be handled with care, memory managed with constant time.

In the embedded world fortunately the requirements placed on an operating system are of a different nature and somewhat simpler: all the tasks are usually known a priori at design time and there are usually fewer tasks with simple resource sharing requirements. It is therefore possible (and desirable) to use a reduced set of kernel primitives resulting in a much smaller system with low memory footprint. Sometimes a flexibility of a real-time operating system may be seen as an important feature. It refers to the ease with which it can be configured and/or modified to suite the needs of a developer.

[*]as of version 2.6, a Linux kernel has introduced a series of real-time features. To my knowledge a properly configured scaled down version of 2.6 kernel may be successfully used as a soft real time os.

Real-time operating systems in use today almost exclusively use preemptive scheduling[†] in order to guarantee a quick responsiveness for high priority tasks. However, non-preemptive scheduling policies have some advantages over preemptive scheduling which are particularly important in embedded systems, as will be shown in Chapter 4.

Although some important results have been found for non preemptive scheduling, the research field still has some open problems.

1.2 Problems addressed in the thesis

Most techniques in the literature focus on environments which posses a lot of structure. Outdoor environments may be completely unstructured or just partially structured, anyhow the environments come in a much larger variety than indoor ones. One important goal is to extend the existing techniques to unstructured environments. This thesis contributes to this goal by proposing new methods and devices and improving the existing ones for mapping of unstructured environments in real time during exploration with a mobile robot.

It is established opinion that in approximately two decades of research the field of robotic mapping has become mature in the sense that many good algorithms have been developed for indoor environments. The existing techniques are robust to noise and allow the building of detailed maps in real-time. Nevertheless, a large number of challenging open problems remains. Some critical issues that must be improved include the need to reduce the amount of processing requirements which in current techniques is very high and to develop accurate and reliable algorithms for matching local maps to the stored map. These approaches must be tested in real-world environments.

Some of these open problems are addressed in this thesis.

The next chapter of this thesis addresses the problems of ultrasonic sensors which are widely used for mapping and obstacle detection during exploration. The developed system focuses the ultrasonic beam of the most common ultrasonic sensor to extend its range and improve the spatial resolution. Extended range makes this sensor much more suitable for mapping of outdoor environments which are typically larger. Improved spatial resolution enables the usage of recent laser scan matching techniques on the sonar scans of the environment collected with the sensor. Furthermore, an algorithm is proposed to mitigate some undesirable effects and problems of the ultrasonic sensor.

Scan matching techniques presented and discussed in Chapter 3 address the robotic mapping problem, more precisely they solve the problem of matching local maps to the stored map. These techniques are used to cancel out the

[†]In preemptive scheduling the scheduler may stop any task at any point in its execution in order to start a task with higher priority. After the completion of the high priority task the execution of the preempted task may resume. See § 4.2.1 in Chapter 4 for details.

effects of noise on localization and mapping. In particular during exploration scan matching is often used to correct the accumulated positional error using dead reckoning sensors like odometry and inertial sensors. Another open problem that may be approached with global scan matching techniques is the estimation of the overlap of the maps acquired locally by teams of robots during exploration. In multi-robot collaborative mapping the relative initial position of the robots may be unknown. In applications suitable for scan matching, when scan matching is successfully applied, it effectively solves the correspondence problem.

Of many scan matching techniques that have been proposed in the last years, few are suitable for implementation in feature-poor unstructured environments. Even less algorithms are robust to high sensor noise, as is the case with the sonar readings used in this thesis which are much noisier than laser scanners. A further constraint is that the frequency at which the scans become available may be very low. In some applications continuous laser scanning may not be desirable and if sonar sensor is used, it is intrinsically a slow sensor[‡] as opposed to laser scanner. Furthermore in a real world application the power consumption is a concern. In this thesis a scan matching solution capable of coping with such conditions while still not placing a high computational burden on the processor is presented and compared to closest rivals. A further benefit of the technique is that it provides a multi-modal solution that allows to handle the ambiguities in the mapping process and may easily be integrated in a SLAM framework. In particular the algorithm is not affected by the other two challenging aspects of the robotic mapping problem as mentioned in the previous section. In fact it copes very well with the high dimensionality of the problem and is suitable for real time implementation. Finally it is worth mentioning that many techniques developed in mobile robotics for solving the correspondence problem have their counterparts in computer vision. This is also true for the algorithm presented in the Chapter 3 but it has not yet been fully explored.

As already said for current research in mobile robotics it is critical to evaluate the above mentioned methods and devices in real world applications on a mobile robot with limited power and computational resources. This problem has been studied in the Chapter 4 of this thesis dealing with embedded systems and real time scheduling. Some new theoretical results are derived concerning open problems in non-preemptive scheduling of periodic tasks on a uniprocessor. This results are then used to propose a design methodology which is used in an application on a mobile robot. The mobile robot is equipped with an embedded system running a new real-time kernel with a non-preemptive scheduler of periodic tasks. The application is described and some preliminary mapping results are presented.

[‡]ultrasonic sensors use the Time-Of-Flight principle. Each sensor reading is limited by the speed of sound in the air as opposed to speed of light of laser scanners

The contributions of this thesis are stated in more detail at the end of corresponding chapters.

1.3 Thesis organization

This thesis is structured as follows:

In Chapter 2 a solution which uses a focused ultrasonic sensor and a fast algorithm to process the acquired readings with the aim of developing a reliable map of the environment are presented. Chapter 3 introduces the scan matching problem in mobile robotics and presents a new fast genetic algorithm suitable for unstructured environments as a robust solution. Chapter 4 discusses the issues regarding the implementation of the techniques on a real-time operating system running on an embedded platform in the robot. In Chapter 5 concluding remarks and further work are discussed.

An exploration algorithm using the above techniques is presented in the Appendix A. The Appendixes B and C offer some details on a real-time kernel and the used ultrasonic sensor respectively.

The list of publications that originated from the research is included at the end of the thesis.

Chapter 2

Ultrasonic sensors

2.1 Introduction

Exploration of an unknown environment with a mobile robot requires registering the data coming from a suitable sensor in order to acquire partial knowledge of the environment. Widely used sensors to make such measurements are ultrasonic and laser transducers [8].

Ultrasonic sensors are very popular in robotics. They are economical external sensing systems mainly used for obstacle detection and map building [8]. Despite their popularity, ultrasonic sensors have two main shortcomings leading to disappointing performance: uncertainty in target location and multiple reflections. The former is caused by wide beam width and the latter gives erroneous distance measurements because of the insertion of spikes not directly connected to the target.

A large amount of research has been carried out on ultrasonic sensors [37, 43, 34]. Research in ultrasonic sensors has been mainly concentrated on fixed arrays, while very little efforts have been made on rotating sonar models. However, provided that the ultrasonic beam is focused, rotating sonar models have some advantages over sonar arrays. The main advantage is spatial resolution. With a stationary sonar array, in fact, the resolution of a sonar scan is limited by the number of sensors that can be fit into the array, while with a rotating model the sonar can be rotated at a very fine rate. The result is a more accurate representation of the environment. Another advantage in using a rotating sonar on a mobile robot, is that the scan can be performed without the robot moving its base which could lead to loss of orientation. Moreover, the ultrasonic beam of rotating sonars can be focused at a much higher degree than fixed sonar arrays.

Most rotating systems used in exploration are based on laser range scanners which provide narrow and precise beam resulting in an accurate contour of the environment. The advantage, however, of the ultrasonic sensor in general, over the laser range finder is that it is not hazardous for human eyes and that it may be used in environments with smoke or fog. It is not affected by the sunlight which, coming through the windows, adds noise in laser scanners acquired maps. It is also more economical. Anyway, the sensor data must be processed to enhance the signal and extract the information content of the registration.

With the aim of overcoming the problems of ultrasonic sensors, this chapter presents a solution which uses a focused ultrasonic sensor and a fast algorithm to process the acquired readings in order to develop a reliable map of the environment. The algorithm outperforms classical approaches in case of high irregularities and missing reflections.

In the first stage the ultrasonic beam is focused by using a parabolic lens. When the parabolic lens is rotated, then a sonar view of the environment is obtained. In general, the transducer should be put in different points because the view depends on the position where the transducer is located.

If a single sonar measurement is considered, the sensor reading consists of a number of pulses reflected by an obstacle. From a series of sensor readings (at different sonar angles) the sequence of pulses reflected by the environment changes according to the distance between the sensor and the environment. This results in an image of reflections that can be built by representing the reading angle on the horizontal axis and the echoes acquired by the sensor on the vertical one. The image thus represents the echoes in a time-distance plane thereof called the time-distance image. The characteristics of a sonar emission result in a texture embedded in the image as will be clear shortly. This texture is the key to the algorithm.

The algorithm described in this chapter performs a 2D texture analysis of the sonar reflections image in such a way that the texture continuity is analyzed at the overall image scale, thus enabling the correction of the texture continuity by restoring weak or missing reflections.

Basically the algorithm can be divided into two parts. The first deals with signal enhancement and correction. It is based on extracting geometric semantic attributes from the images representing acquired raw sensor data. The image is studied by performing texture analysis and statistically classifying each texel into which the image is divided. Next, the missing parts of the signal and/or those corrupted by noise are restored where possible.

The second part of the algorithm applies heuristic rules to find the leading pulse of the echo and to estimate the obstacle location in points where otherwise it would not be possible due to noise or lack of signal.

This chapter is structured as follows: Section 2.2 concentrates on the preliminaries regarding the practical applications of ultrasonic sensors; Section 2.3 provides an overview from literature as to the solution to such problems; Section 2.4

tackles the beam focusing; Section 2.5 discusses texture analysis of ultrasonic sensorial data. The two techniques are combined, resulting in the description of an enhancing technique as per Section 2.5.1; Section 2.6 illustrates some experimental results of acquisition in a real environment, and finally Section 2.7 provides the conclusions.

2.2 Preliminaries

In the discussion that follows the robot acquires the data about the environment by performing a scan with the rotating sensor in different positions. It is assumed that the robot stays in a fixed position while performing the scan. The scan is achieved with a stepping motor rotating the transducer and stopping for a while on each step to take the measurement. The transducer is than rotated to the next position, where a new measurement is taken.

The ultrasonic transducer used in this thesis is a Polaroid module, described in the Appendix § C.

The time necessary for a measurement is the time needed for a burst of ultrasound to travel to the obstacle and back. In single measurements of typical ranges below 10 m the suitable time interval for performing one step is around 50 ms. With the spatial resolution of 0.9^o for a single step, our motor completes the whole scan of the environment in 20 s. However, depending on the exploration strategy, it is seldom necessary to scan the whole way round and therefore the spatial resolution can be reduced to selected areas of interest; in this case the scanning time is greatly reduced. The rotation is performed on a plane, thus the resulting map is related to a plane at a given height.

The utility of the increased spatial resolution gained by rotating the sensor is severely reduced by some important shortcomings of ultrasonic sensors which are well known in literature:

- large beam width;
- side lobes;
- specular reflection;
- short range.

Large beam width is responsible for walls appearing as arcs because the echo is reflected from the same surface in several consecutive steps, so the apparent distance remains the same. The side lobes may trigger false readings while specular reflection leads to either missing reflections where no echo is detected or produces 'ghost' walls in the map because the echo bounces from two or more surfaces before returning to the sensor, resulting in a longer path traveled by

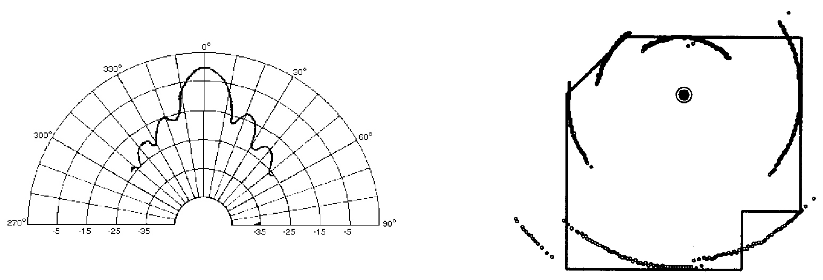


Figure 2.1: The left figure shows a typical propagation pattern for the Polaroid 6500 Series ultrasonic module. The map obtained interpreting the scan of a small room is shown in the right figure, where the digital output of the module is considered. Ghost walls and arc patterns are typical.

the echo. It is particularly difficult to deal with the specular reflection near the corners of indoor smooth walls.

2.3 Related work

Focusing ultrasonic beams with horns, paraboloid reflectors or lens has been known for a long time. Crowley, for example, has used a focusing horn [15] reducing the beam width to 5^o. However, in more recent work in robotics it was rarely used, neglecting the benefits it brings. A different approach uses sensor arrays for reducing the beam width [11].

Several approaches for estimation of maps were studied, some using a single rotating sonar scanning and others using sonar arrays. The typical operation performed is to start from the digital output signal of the ultrasonic module and then to interpolate the points forming the targets.

Leonard and Durrant-Whyte [34] presented an extended Kalman filter-based localization algorithm, which provides fast vehicle position updates by matching individual sonar returns from a ring of sensors to an a-priori map of geometric beacons (planes, corners and cylinders). The feature extracted from sonar data is the so-called Region of Constant Depth (RCD). The RCD is a connected set of sonar returns with constant range data, and it is a sensor feature that can be acquired only from distinctive objects like walls, corners, edges and cylinders. These features are particularly evident in figure 2.1 as arc patterns due to the large beam width of the used ultrasonic sensor. Leonard and Durrant-Whyte [35] have extensively studied RCDs and successfully used them in localization and navigation.

In [43] the problem of building a map of an unknown environment for mobile vehicle navigation was studied using sonar scan. The algorithm proposed was based upon multiple-hypothesis multi target tracking methodology using probabilistic interpretations of possible measurement-to-features associations. No a-priori information is required. The geometric features of the environment are classified in confirmed and tentative feature tracks. From the sonar scan, RCDs are extracted.

The relocation using echo location estimation is issued in [37], where both single rotating sonar and a ring array are considered. This algorithm estimates the position determining the best match between the range returns and the environmental model. The features extracted are RCDs.

In [29] the data obtained from sonar scan are fused with laser data in order to obtain contours of the environments. In this case, as well, the features used by sonar scan are RCDs.

In any case, in classical algorithms it is impossible to know whether the returning echoes are related to an artifact or not. Typically, classical algorithms perform a filtering of echoes, mostly by interpolation. The algorithm described in this thesis, instead, can be aware of the nature of the echoes; that is if the echo is due to noise or not.

2.4 Enhancing the signal: Focusing

A system was built with an ultrasonic transducer put in the focus of the paraboloid reflector. The reflector was obtained from a small satellite dish and is therefore lightweight and cheap. The system is 30 cm long with a 20 cm diameter reflector. Placing the stepping motor axes in the middle results in a minimum radius of 15 cm which is to be taken into consideration when mounting on the robot. It

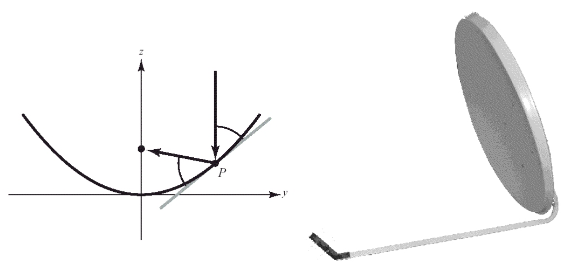


Figure 2.2: The transducer is placed in the focus of a paraboloid reflector.

turned out experimentally that this solution is an effective remedy for some deficiencies of ultrasonic sensors: the wide beam width, side lobes and short range, as reported hereafter.

As compared to the non-focused transducer, the strength of the signal is much stronger, the beam width is narrower (less than 2^o) and the sensitivity of the sensor is increased. It is worth noting that it is even narrower than what was reported in Borenstein [11] in the case of fixed sensors.

A stronger signal means that obstacles at a greater distance may be detected. Obstacles were detected at a range of over 12 m. Detecting objects at such distances for mapping purposes with a simple ultrasonic sensor in most cases would not make sense because the width of the beam would make it impossible to know the precise location of the object. However, with the new sensor, the focused and constant width of the beam makes that possible and desirable.

The narrow beam width is very attractive because it greatly reduces the uncertainty of the measurement process and provides greater spatial resolution, adding value and confirming the benefit of the exploration with the rotating sonar.

Specular reflection is still present, but with increased sensor sensitivity it occurs only for very smooth surfaces. In that case it is presumably easier to manage in the interpretation phase given the fact that it is produced by a narrower beam. It is possible to sense walls at nearly all incident angles, provided that the wall itself is not a perfectly smooth surface.

The sensor can sense the echo returning from flat office like surfaces at over 45 incidence angle if the surface presents small imperfections in millimeter range. This fact comes somehow as a surprise knowing that a wavelength of the ultrasonic burst is around 7 mm. The fact that the perfectly mirror-like smooth surfaces are difficult to detect causing specular reflection and that at least minor irregularities should be present makes the sensor better suited for outdoor or industrial applications, where rough surfaces are more frequent. This is consistent with the fact that this thesis focuses on more challenging outdoor environments. Focused beam greatly improves the effectiveness of the rotating sensor for the mapping process at a loss of compactness, however some problems still remain. Firstly, the variation of the signal strength when the beam traverses surfaces with different reflective properties or at different incidence angles. This variation of the signal strength translates in a variation of the distance estimated by the module. Secondly, there are missing reflections due to the absorbing characteristics of the target.

2.5 Enhancing the signal: Texture Analysis

The analog signal generated by the ultrasonic transducer was converted into digital format using a 200 kHz sampling rate acquisition system. Fig. 2.3 shows the raw signal acquired from the ultrasonic sensor. Fig. 2.4 shows how the strength of the returned signal is affected by different surfaces. On the left of Fig. 2.4 the sonar scans obtained with the rotating sensor in the small room depicted on the right of Fig. 2.4 are shown. The walls are, in some parts, covered by posters.

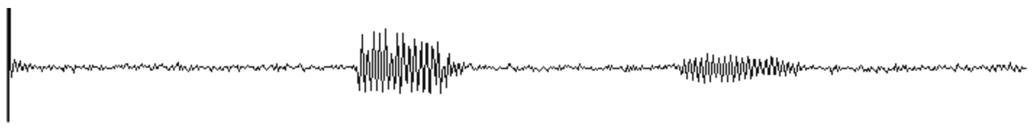


Figure 2.3: Raw signal acquired directly on the transducer. This is a strong echo reflected directly from orthogonal surface and bouncing front and back causing secondary echoes to appear. Weaker echoes are sometimes below the noise level and difficult to detect.

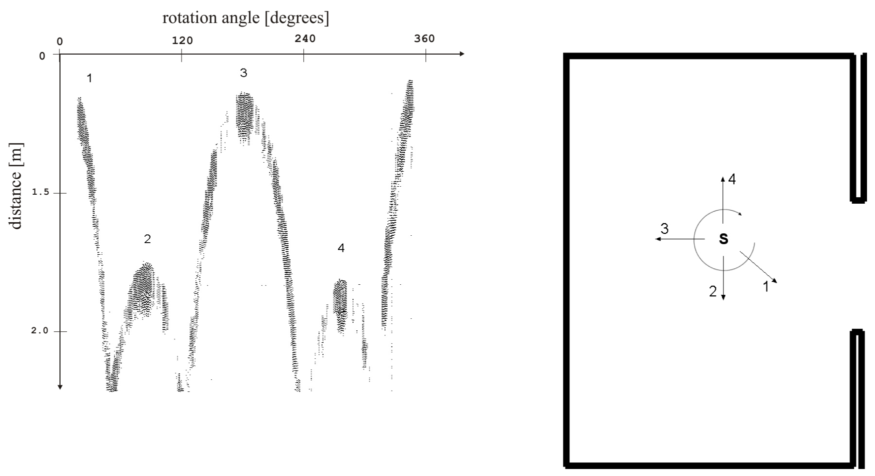


Figure 2.4: One scan of the small room. The stepping motor rotates the sensor in a clockwise direction and performs a reading at each step. The raw signal of each reading is in columns along the x axis. The black pixels are values above the threshold and identify the echo. As the sensor rotates, the distance of the echo varies, tracing the pattern which is may be explained more easily if compared with the figure on the right.

Each column of the picture represents a reading, that is a single step of the scan. The black and white pixels are obtained applying a threshold to the raw reading of the transducer as opposed to the digital output of the Polaroid module like the one shown on the right of Fig. 2.1.

Fig. 2.5 shows a 40^o portion around the position marked as 3 in Fig. 2.4. In an ideal scenario, a portion of the echo returns to the sensor and is detected regardless of the incidence angle and surface properties, thus creating a profile like the one depicted with a solid line on the left of Fig. 2.5. Nevertheless, the digital

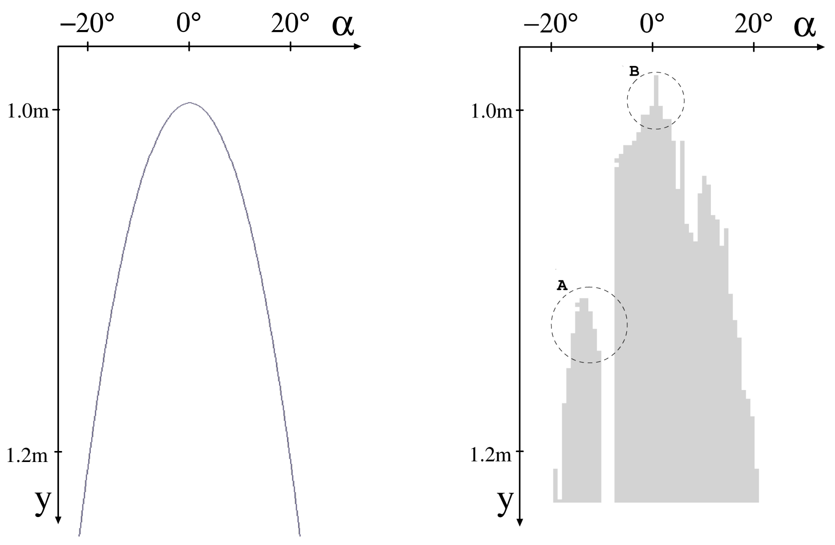


Figure 2.5: A portion of the same environment from Fig. 2.4 as represented on the digital output of the Polaroid module. Solid line on the left is the ideal profile: for a straight wall it is a function of stepping motor angle, given by $\frac{1}{\cos(x)}$.

output has some major problems, shown in the actual measurement depicted on the right of Fig. 2.5.

First, as indicated with **A** in Fig. 2.5, the distance read by the module deviates from the ideal one because the beam traverses (at a non-orthogonal incidence angle) a region where smooth surface reflects the echo away from the sensor like a mirror. As the beam abandons the previous region the echo becomes weaker, and finally missing readings occur when the whole beam is directed on the smooth surface so no echo returns to the sensor.

Second, when the beam is orthogonal to the surface, the reflected echo is very strong, much stronger then usual (as indicated with **B** in Fig. 2.5).

Furthermore, missing reflections may occur during the mapping of the environment due to several causes. They are usually caused by specular reflections where the portion of the signal that returns to the sensor is too small to be detected, but there are other important causes that result in interruptions in time-distance image, for instance when a person or animal passes in front of the sensor, or the interference of occasional objects such as wires, grass, particles etc. Multiple echoes have similar effect. They arise when the scan is too fast, thus resulting in the previous echo being still around or in environments with multiple sensors where crosstalk occurs. The developed algorithm corrects the missing reflections by estimating the location of the echo on the basis of neighboring readings, as will be explained in detail in the next section. It is worth noting, at this point, that the correction of missing reflections can cancel real small objects or openings from the map. If the purpose of the algorithm is to accurately map small objects then this correction should be taken into account.

In order to estimate the position of the missing echo a simple approach would be to use only the distances provided by the module and then to perform an interpolation among adjacent measurements. This approach is, however, based on a single measure for each reading: the distance estimate from a control module corresponding roughly to the first impulse of the burst that constitutes the echo and thus changes with the variation of the signal strength. Using the Hough transform [56], for example, a line fitting technique is used. This approach requires some assumption about the environment configuration: small obstacles or irregularities are filtered out with Hough transform.

To study the environment with irregular surface profiles, a panel was folded and inserted in the environment as shown in Fig. 2.6, on the left. Some areas of the panel had different reflective properties which caused problems visible in Fig. 2.6, on the right. Putting together our experimental findings it can be said that in some cases the signal was too weak to be detected. In these cases, it is better to look at the whole time frequency image in order to assist us in the signal detection.

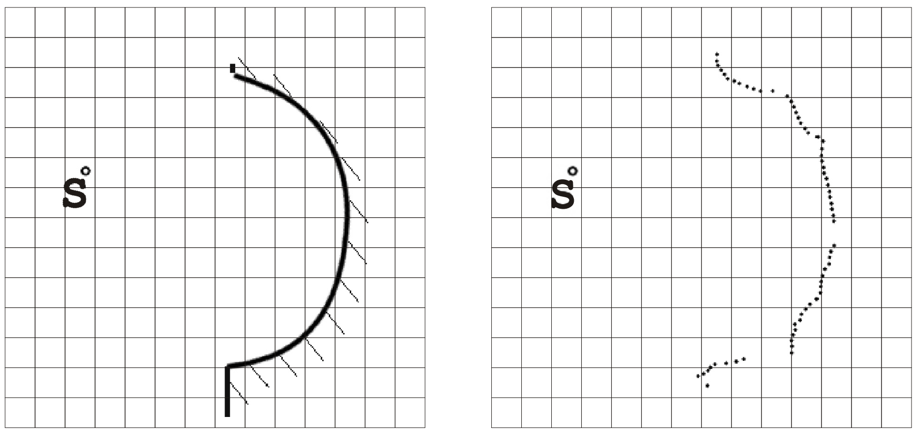


Figure 2.6: Left: actual profile of the environment. Right: the profile obtained with a focused sensor. The grid step is 20 cm for both directions.

Due to the characteristics of the Polaroid module, if a single sonar measurement is considered, the sensor reading is formed by a burst of pulses reflected by the obstacle. Strong echoes have 20 or more easily detectable pulses while the weak ones can have just a few above noise level, for example near the center of the echo, as per Fig. 2.3 and Fig. 2.4. The algorithm is based on the idea of using the whole burst of pulses, and not just the first pulse. As it is not based solely on the first pulse it is less sensible to signal strength variation.

The algorithm works on the time-distance image (Fig. 2.4) and is described in the next section, pointing out how it enhances the time-distance image and then registers the profile of the echo.

2.5.1 The fast texture analysis algorithm

The first step of the algorithm is to read the time-distance image, such as that of Fig. 2.4, selecting the pixels which represent meaningful reflections regardless of their amplitudes. This is performed using a threshold on the acquired signal of each reading of the sensor. The threshold is adaptive, because it should be high enough to eliminate noise and perturbations, but also as low as possible to be able to sense weak echoes, which are the majority. The time-distance bitmap is then divided into $16x16$ pixel texels. If an acquisition is performed every 50 ms,

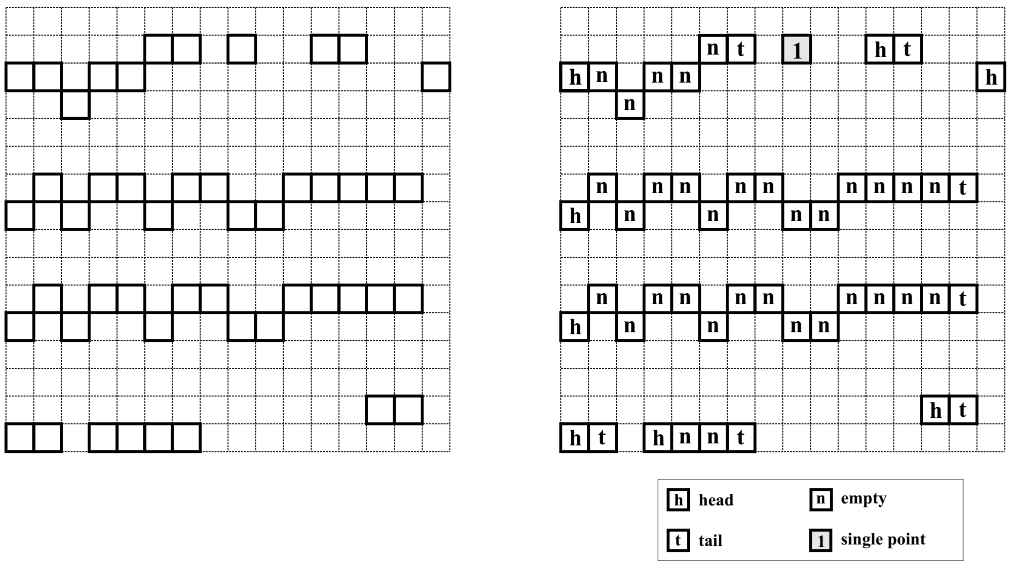


Figure 2.7: Examples of elementary runs detected inside a texel from the image reported in fig. 2.4.

the duration of a texel is equal to 0.8 s. In figure 2.7, left panel, a texel is reported where each bold square represents a pixel greater than adaptive threshold. This figure shows that the reflections appear as formed by a concatenation of elementary runs, which are sets of adjacent pixels greater than the threshold in the image. Runs extraction is performed in each texel by exploring the neighbors of each pixel at right, upper right and at lower right. The runs construction is presented in figure 2.7, right panel; at the end of the process a pixel called *head* and a pixel called *tail* of the run appear. The runs are reconstructed by connecting head and tail with a straight line.

According to [27, 13], each texel T_{ij} is associated to one statistical measure, given by (2.1). The measure reported in (2.1) emphasizes the presence of long elementary runs; the indexes i, j explore the horizontal and vertical texels of the texture.

$$L_{ij} = \frac{\sum_{l=1}^{n} l^2 R(l)}{\sum_{l=1}^{n} R(l)} \tag{2.1}$$

In (2.1), $R(l)$ is the number of runs of length equal to l pixels, and n is the maximum length. The operator (2.1) is applied for each of the following slope ranges, numbered from -3 to $+3$:

$$\begin{aligned}(-45^o..-31^o),(-31^o..-19^o),(-19^o..-7^o),(-7^o..+7^o),\\ (+7^o..+19^o),(+19^o..+31^o),(+31^o..+45^o)\end{aligned} \tag{2.2}$$

resulting in vectors $L = (L_{ij}(-3), \ldots, L_{ij}(+3))$ for each texel T_{ij}, aiming at measuring the presence of long elementary runs at each of the ranges of (2.2).

Each texel is then labelled as $(-3, -2, \ldots, +3)$ according to the predominant slope of the runs contained in it, looking at the range of (2.2) where the predominant slope falls. Texel labelling is described below.

First of all, vector L is normalized to obtain confidence value Π_{ij} using (2.3) where thr takes into account the maximum values observed in L. Empirically, it was observed that good values for thr, in this case, are around 100.

$$\Pi_{ij}(k) = \begin{cases} 1 & \text{if } L_{ij}(k) \geq thr \\ \frac{L_{ij}(k)}{thr} & \text{otherwise} \end{cases} \tag{2.3}$$

Confidence values $\Pi_{i,j}(k)$ for the predominant direction of the runs in each texel are thus evaluated by measuring at what degree long elementary runs are present in the texel.

By normalizing the confidence values, i.e. by setting $P_{ij}(k) = \frac{\Pi_{ij}(k)}{\sum_{n=-3}^{+3} \Pi_{ij}(n)}$, the $P_{ij}(k)$ values are in the range $[0, 1]$ and, moreover, $\sum_{k=-3}^{+3} P_{ij}(k) = 1$. Each texel is thus characterized by seven probabilities $P_{ij}(k)$, $k = -3, -2, .., +3$ which measure the presence of long runs in the 7 directions reported in (2.2).

It may happen that the probabilities in one or more texels in a column are incorrect due to noise. The texels in a column should have, however, similar probabilities because they are related to the same reflection. The probabilities in a column are thus corrected using an iterative relaxation. At each step, the probability value of a given label in a texel is increased if the adjacent texels in the column have a high probability for the same label. Conversely, the probability should be decreased if adjacent texels in the column have a low probability for that label. Otherwise, it should remain unaltered.

To perform this improvement, the iterative relaxation process proposed in [57] has been used in the following way.

```
for each column of texels do
```

```
while (iteration not converged) Do {
  for each texel do
    for each predominant slope k Do {
      compute the sum of the Pij(k) for the upper
      and the lower adjacent texels;
      compute the correction factors Qij(k) by mapping
      to [-1..1] the above sums;
    }
  for each texel Do
    update the probabilities Pij(k) using
```

$$P_{ij}^{(r+1)}(k) = \frac{P_{ij}^{(r)}(k)[1 + Q_{ij}^{(r)}(k)]}{\sum_l P_{ij}^{(r)}(l)[1 + Q_{ij}^{(r)}(l)]} \tag{2.4}$$

```
} // End while
```

Suppose, in fact, that there is a high similarity in adjacent texels for a given label: the correction factor $Q(k)$ becomes close to 1 and therefore the probability of the actual texel $P(k)$ is increased. On the other hand, if there is low similarity, then the correcting factor tends to -1, thus decreasing the probability value. The curves of fig. 2.8 represent a typical iteration correcting the probabilities.

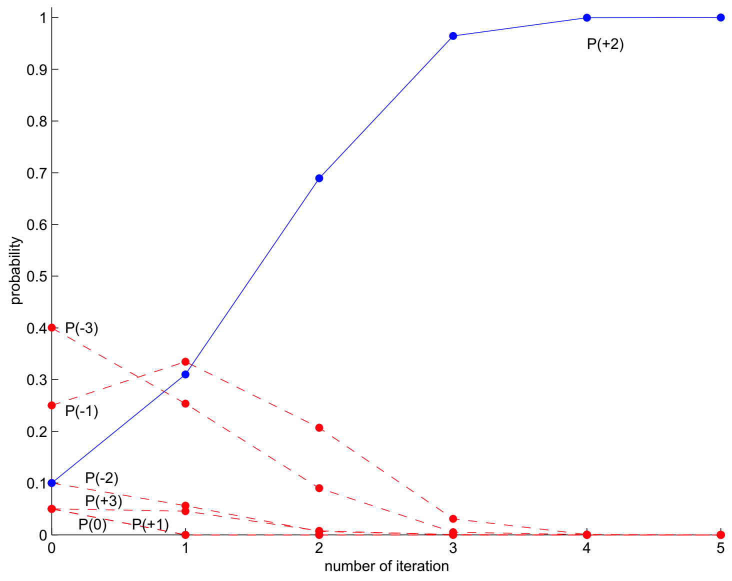


Figure 2.8: Probabilities of each orientation for one texel in five iteration of the relaxation procedure.

This figure describes the behaviour of the probabilities in a given texel during the relaxation process. Each curve represents the probability for different orientations, from -3 $(-45^o..-31^o)$ to $+3$ $(+31^o..+45^o)$. The starting values are such that probabilities $P(-3)$ and $P(+2)$ are the only two not similar from the same probabilities of the adjacent texels, being equal to 0.4 and 0.1 respectively while the adjacent texels are around 0.1 and 0.4 respectively. The result of the relaxation is that the same probabilities of all the texels in a column become very similar. As is shown in fig. 2.8, $P(2)$ becomes 1 and all the others become 0; the same happens in the other texels. Moreover, this figure shows that the relaxation converges in a few steps.

It is worth noting that in each column, the texture spreads across four or five texels. Recall in fact that, since the sampling frequency is 200 kHz and the ultrasonic frequency is 50 kHz, a single pulse of the chirp is represented by 4 samples; hence, the returning echo, which always contains more than 16 pulses due to residual vibrations of the sensor, is represented by at least 64 samples, usually more, which correspond to 4 or 5 texels. Therefore the relaxation process is applied to triples of texels, from the upper to the lower, and then the procedure is repeated until all the 4 texels have similar probabilities.

Texel T_{ij} is finally labelled from -3 to $+3$ according to the predominant slope in the texel as follows:

$$Label_{ij} = argmax\{P_{ij}(k)\}, \quad k = -3, -2, .., +3 \tag{2.5}$$

If a texel is not labelled with the above procedure, then it is marked as *not classified* and labelled with 4.

The next step of the algorithm is connecting the elementary runs. During the connection, the continuity of the reflections is restored in points where runs were interrupted by noise. The connection is performed only in texels belonging to the texture. In such texels, some heuristic rules for connection are applied. The rules basically state that, since in principle every couple of runs orientation is possible, a point in which the orientation must be considered is in the texel where the run seeking a connection ends. In order to know in which direction the search must be explored, it is also necessary to know the orientation of the adjacent texel on the right.

The heuristic is implemented using a search window, described with a quadratic function, which is computed as follows: the orientation of the texel in which the run terminates defines the starting slope of the window (γ_1), while the neighboring texel defines the ending slope (γ_2). Taking the derivatives of the second order function in starting and ending points of the two texels and imposing the slopes respectively yields the following relation for the central points of the window:

$$j = \frac{\gamma_1 i - (\gamma_2 - \gamma_1) i^2}{64} \tag{2.6}$$

where γ_1 and γ_2 are the orientations of the left and right texels, and i and j are the row and the column of the pixels. The minimum size of detectable obstacles is related to the search window depth.

The actual search window starts directly from the terminating pixel (it is shifted accordingly) and the search involves three pixels around the central point in the order as depicted in figures 2.9, 2.10 and 2.11, which show the search window in case of a run ending in a texel with an orientation equal to +3 ($+31^o$..$+45^o$) that must be connected to another run and the adjacent texel has an orientation equal to -3 ($-45^0..-31^o$), 0 ($-7^o..+7^o$) and $+3$ ($+31^o..+45^o$) respectively. The pixel to be extended is the one denoted by a gray level and the numbers inside the squares denote the order of search. The search window has a maximum depth of 32 pixels and a height of 3 pixels.

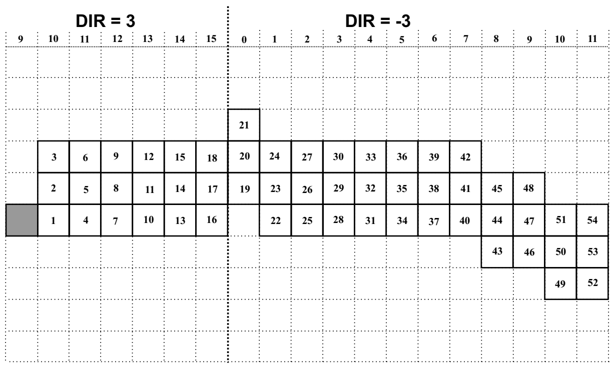


Figure 2.9: Search windows connecting a run in a texel with a slope +3 ($+31^o..+45^o$) with a run in a texel with a slope -3 ($-45^o..-31^o$).

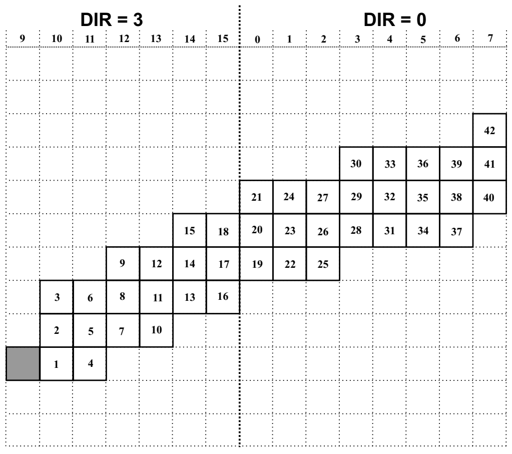


Figure 2.10: Search windows connecting a run in a texel with a slope +3 ($+31^o..+45^o$) with a run in a texel with a slope 0 ($-7^o..+7^o$).

In the final step of the algorithm, the distance from obstacles is calculated by searching for the texture representing the reflected impulse train in every column of the image enhanced by the connection process.

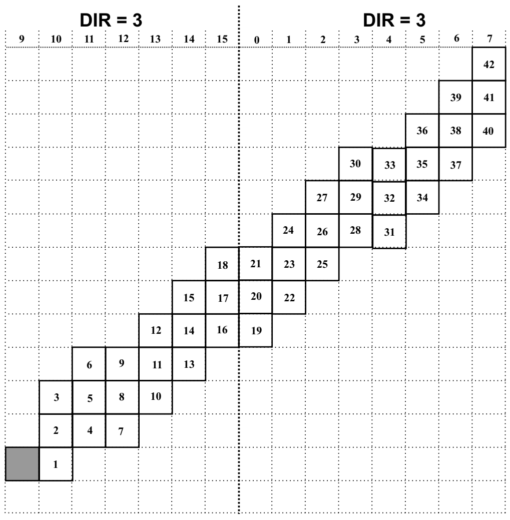


Figure 2.11: Search windows connecting a run in a texel with a slope +3 ($+31^o..+45^o$) with a run in a texel with a slope +3 ($+31^o..+45^o$).

2.6 Experimental results

In this section some results obtained with the described algorithm are presented. All figures show real measurement data and the corresponding output of the algorithm.

The first thing to stress is that the algorithm can be used as far as the elementary runs can be extracted from the time-distance bitmap. An elementary run means that a sequence of adjacent pixels is identified and thus the slope formed by two adjacent pixels must be in the range $-45^o..+45^o$. If the slope is greater than 45^o, the pixel cannot be adjacent anymore. This is the limit of the algorithm, which works only if the runs have a slope in that range. Therefore, if the number of measurements and the step angle are fixed, there are sections of obstacles which cannot be analyzed with the algorithm. However, the section can be somehow enlarged in the following ways. A rough initial estimate of the position of the echo is available in the initial thresholding phase. This information can be used to distort the image enabling larger sections of the signal to be elaborated by the algorithm. This approach has been verified and used in all the examples.

The following figures present the results obtained with real measurements. Time-distance image before and after the processing with the algorithm is shown in Fig. 2.12. The algorithm is applied to the raw signal acquired from the focused ultrasonic sensor as discussed in Section 2.5. It performed the signal enhancement as shown in Fig. 2.12, on the right, and then evaluated the profile of the echo as indicated in bold. The profile is estimated only in the main portion of the cone due to the limit on the slope discussed in 2.5.1.

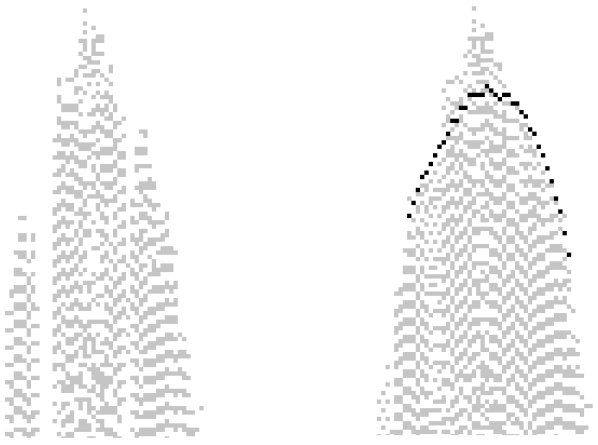


Figure 2.12: Left: raw signal. Right: the algorithm enhances the image on the left and then evaluates the profile which is indicated in bold.

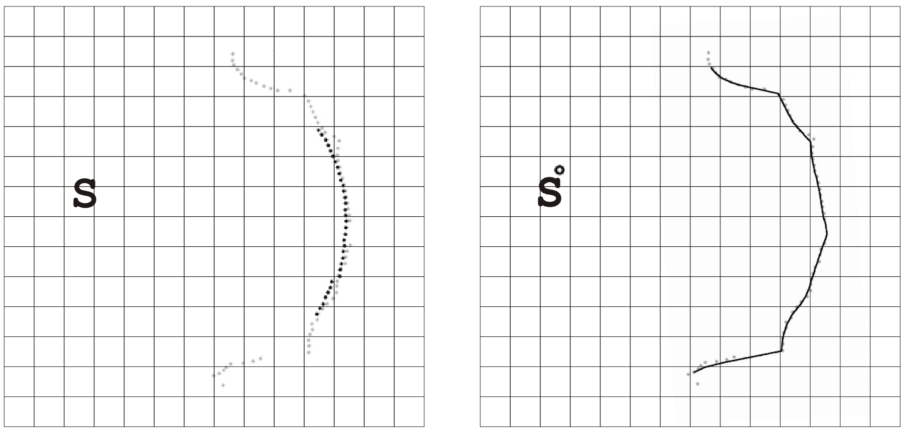


Figure 2.13: Left: the profile after the application of the algorithm. The corrected section is in bold. Right: the profile obtained with a focused sensor using Hough transform.

In the case of a concave surface profile as per Fig. 2.6 the missing reflections and errors of estimated distances are successfully corrected by the algorithm as shown in Fig. 2.13, on the left. For comparison, the profile obtained with the Hough transform is shown on the right of Fig. 2.13. It is clear from Section 2.5.1 that the echo corrections performed by the algorithm result from an image understanding procedure based on the texture of the ultrasonic image. Therefore, since the echoes are selected on the basis of their significance, the algorithm is able to accurately detect surface irregularities which are otherwise filtered out with the classical approaches.

Fig. 2.14 presents examples of mapping the environment with the data obtained with focused ultrasonic beam. The quality of the maps is significantly improved by the focused beam if compared to the simple sensor. The algorithm then further improves the map in sections where it can be applied.

The experimental environment is shown in the upper part of Fig. 2.14; it is made of two rooms, one smaller, on the left, and one larger. The smaller room is the one depicted in Fig. 2.4, on the right.

Other real measurements are presented in Fig. 2.14 obtained by putting the transducer in two positions in the larger environment.

The walls of the environment are partly covered with posters, paintings or paper announcements, all of which is identified under the word “poster” on the map in Fig. 2.14. A poster usually acts as a perfectly specular surface, but this is not the rule because the borders are sometimes caught.

Two scans with a focused beam provide a very good map on a large scale. If compared to the map of Fig. 2.1 there is a marked improvement. There are no ghost walls or arc patterns and walls are detected at all incidence angles. Even the glass door to the right did not cause as many problems as expected thanks to its orthogonal position, a poster and a junction that caused the echo to be detected by the sensor. However some problems are visible in the areas where smooth surfaces cause specular reflection. These problems are corrected by the algorithm as shown in Fig. 2.14 where two corrected sections are superimposed on the map.

Real measurements confirmed that the algorithm effectively corrects the scans in areas where it can be applied. The area corrected from a single position is limited and depends on the distance and incidence angle. In order to cover larger portions of the environment it is therefore necessary to reiterate the method in different positions during the exploration.

2.7 Final remarks and conclusion

Ultrasonic sensors are the most popular external sensing system used in mobile robotics. Besides the common arrangement of ultrasonic sensors in arrays mounted on side of a robot, a single ultrasonic transducer may be used in a rotating configuration to obtain a scan of the surrounding environment. In this work the sensors of the mobile robot were augmented with such a rotating ultrasonic system. In order to increase the poor spatial resolution of the system limited mostly by the sensors’ large beam width, the focusing of the ultrasonic beam was proposed. Experiments with a system consisting of the transducer in the focus of a parabolic deflector confirmed the feasibility of ultrasonic beam focusing which

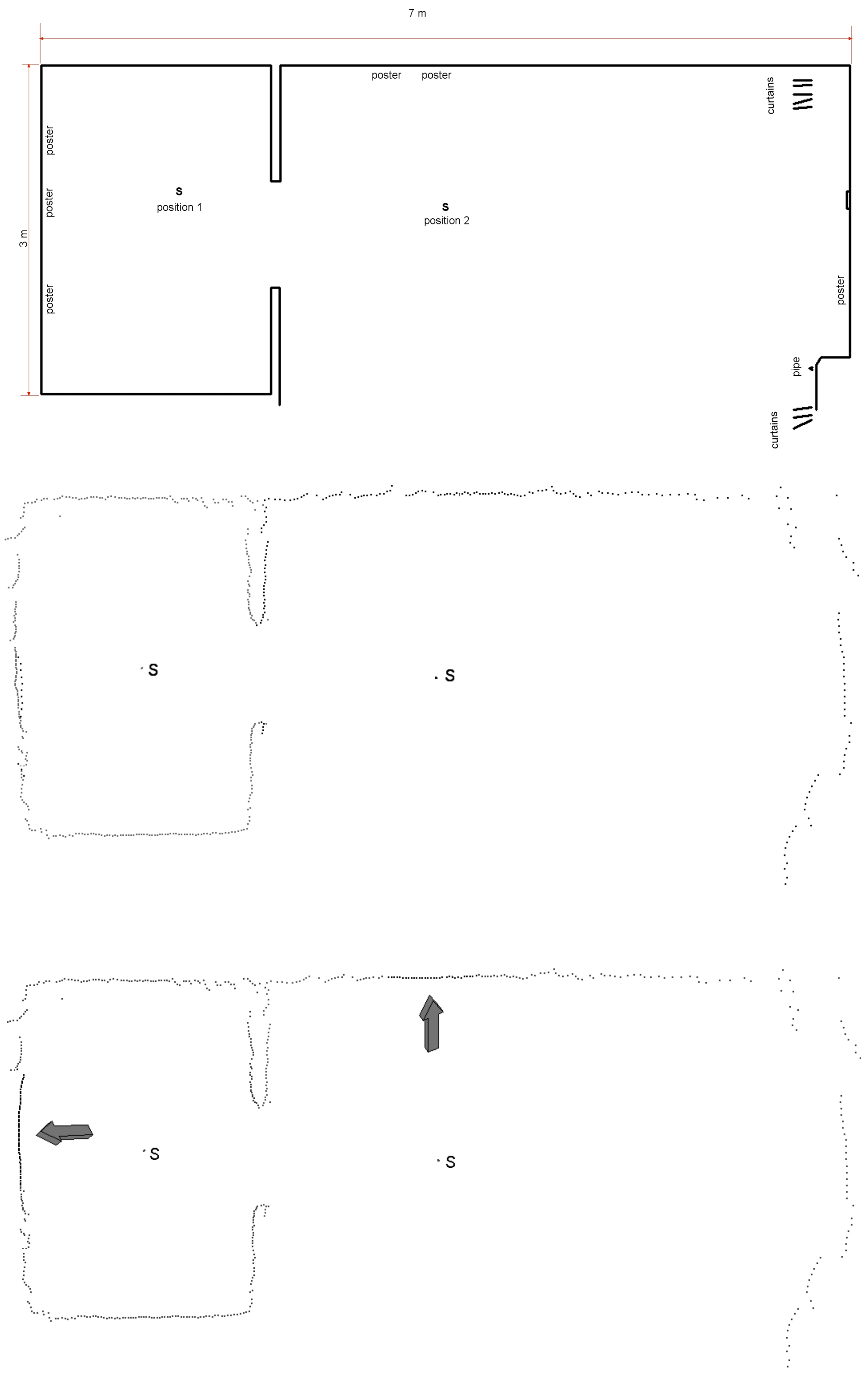


Figure 2.14: Top: real map of the laboratory. Salient features that could affect the measurements are indicated as are the two positions in which the scan was performed. Middle: the map obtained from two scans with the focused ultrasonic beam (in black and gray, respectively). Bottom: two sections corrected by the algorithm are superimposed (in bold).

was studied and characterized. A smaller prototype of the rotating sensor was later mounted on the robot for mapping purposes. The high spatial resolution provided by the system brings the resulting scans closer to those obtained with more expensive laser range finder.

This may be particularly interesting because over the last decade new techniques have emerged in the field of laser scan matching which are more robust to noise. This thesis shows that scan matching techniques may successfully be used with a proposed focused ultrasonic sensor. In the next chapter scan matching will be thoroughly discussed and applied to data gathered with the described system.

However, several shortcomings of ultrasonic sensors reduce the efficiency of the method. Aiming at overcoming specific problems of rotating sonars' environmental scans, an algorithm using raw ultrasonic signal interpretation was developed.

The algorithm processes the readings from a rotating ultrasonic sensor in order to develop a reliable map of the environment. It outperforms classical approaches in case of high irregularities and missing reflections.

Both the developed sensor prototype and the algorithm have some limitations. A rotating focused sonar used in the experiments is rather large and cumbersome and the combined robot motion and the rotation of the prototype may cause vibrations adding noise to measurements. Even though these effects were reduced with careful repeated adjustments of the rotation speed, the engineering problem of developing a more compact and rugged device must still be solved to employ the focused rotating sensor in outdoor conditions during prolonged exploration missions in unstructured environments. The algorithm's main limitation is small coverage area. This area however increases during exploration as more scans are processed from different positions. Moreover, the areas in which the algorithm does not fully perform may be addressed similarly to unexplored areas by moving the robot towards these areas for further exploration. It must also be pointed out that small objects can be detected by the described algorithm if their size is greater than the search windows described in Section 2.5.1.

The algorithm described in this work is suitable for map building during mobile robot exploration missions. Moreover, with small changes the algorithm can easily be applied to other fields where the signal is represented in 2D images with continuous curves that can be studied and therefore enhanced and/or restored.

Chapter 3

Scan matching

3.1 Introduction

The matching and analysis of geometric shapes is an important problem that arises in various applications areas, in particular computer vision, pattern recognition and robotics. In a typical scenario we have two objects and we want to find the optimal transformation (translation and rotation), which match one object on the other one as accurately as possible. Objects are typically represented by finite sets of points in two or three dimensions. An important area of matching of geometric shapes is scan matching applied to robot self-localization. Mobile robots, in fact, must have the ability to ascertain their pose (position and orientation) while navigating or exploring an environment.

Many localization techniques, such as odometry, GPS, inertial systems, range sensors, etc. are typically used in mobile robotics. A robot's position on the map is most easily tracked using dead-reckoning that determines the robot's location by integrating data from wheel encoders (that count the number of wheel rotations). In many cases, however, dead-reckoning fails to accurately position the robot for many reasons, including wheel slippage. If the robot slips, the wheel rotation does not correspond to the robot's motion and thus encoder data, which represents the state of the wheel rotation, does not reflect the robot's net motion, thereby causing positioning error. GPS offers an alternative to dead-reckoning, but GPS signals may not be available, for example in indoor environments. High performance inertial systems, on the other hand, are very expensive, and therefore not suited for a mobile robot. Landmark based localization can be a better choice, but the environments may be unknown and not structured. Therefore, a precise and stable localization becomes a challenging problem when the robot experiences unacceptable positioning error with the odometry, does not have an external positioning device like GPS, and moves in an unknown environment with no artificial or natural landmarks.

This chapter deals with localization methods based on matching of scan data obtained from a range device, namely a laser range finder (LRF) or rotating ultrasonic range devices (Fig. 3.3). Each scan provides a contour of the surrounding environment.

Generally speaking, there are two different approaches to determine the robots pose with scan matching: relative and absolute. In relative approaches - or movement estimation approaches - given an arbitrary initial pose, e.g., $p_0 = (0,0,0)$, the robot's current pose relative to p_0 is incrementally updated when the robot is moving. Let S be the current scan and R the reference scan, e.g., a scan acquired previous to S. If both scans have been acquired from different robot poses, the transformation that maps S onto R is calculated that corresponds to the movement of the robot between the scans. In absolute approaches - or position estimation approaches - on the other hand, the position and orientation of the robot within an a priori given map or a known map is calculated.

Based upon the availability of an approximate alignment of two scans prior to performing the matching, scan matching can be further classified into local and global. Local scan matching is when an initial position estimate (IPE) is available from which to start the search. It is generally used for robot position tracking where some other localization methods like odometry provide an IPE. The provided IPE has typically a small localization error and local scan matching is used to further reduce this error. On the other hand, with global scan matching we are able to generate and evaluate a position hypotheses independently from initial estimates, thus providing the capacity to correct position errors of arbitrary scale.

Even if we narrow down to consider only scan matching techniques pertaining to robot localization and one of the above mentioned approaches, the efficiency of one technique over the other varies very much with the application at hand, i.e. with the type and size of the environment, sensor characteristics, robot speed, etc.

In this chapter the general scan matching problem will be described. The state of the art and important techniques will be tackled before focusing in more depth on most efficient techniques when applied to robot localization in unstructured environments.

Finally a new technique will be proposed which uses a genetic algorithm to solve a problem. A new genetic scan matching algorithm is presented, called GLASM, with the following general properties.

- The algorithm is based on genetic optimization and it is quite robust against noise or incomplete data.
- The devised fitness function does not need to compute any correspondences between range scan points.

- The best results are obtained with Gray coding of the chromosome and with binary representation.
- Typically, the success ratio is better than classical methods like ICP [4] and GCP [68].
- The algorithm is very fast mainly because it is based on a look-up table; moreover the fitness computational complexity is $O(N)$ where N is the number of scan points.

The structure of this chapter is the following. Section 2 addresses the problem of scan matching in general, providing general definitions used in the rest of the chapter and briefly describes the sensors used in this work, in particular ultrasonic focused sonars and laser range sensors. Section 3 reports a brief review of related literature while Section 4 discusses implementation issues and their effect on the performance of scan matching algorithms. Section 5 describes the GLASM algorithm. In Section 6 it is compared with other classical approaches. Final remarks and conclusion are discussed in Section 6.

3.2 Problem formulation

The main issue in robot self-localization is how to match sensed data, acquired with devices such as laser range finders or ultrasonic range sensors, against reference map information. The reference map can be obtained from a previous scan or from an a-priori known map. Given the reference scan from a known position and the new scan in unknown or approximately known position, the scan matching algorithm should provide a position estimate which is close to the true robot position from which the new scan was acquired. This can be done by matching the new scan against the reference scan, i.e by maximizing the degree of overlap between the two scans.

Usually, the problem is solved by defining a distance measure between the two scans and searching for an appropriate rigid transformation which minimizes the distance. In this chapter, we consider a two dimensional case, i.e. we consider a mobile robot on a flat ground. Its pose is described by a triple (x, y, φ) where (x, y) and φ denote the robot position and orientation respectively.

Scan matching can be described in an intuitive way by considering one point of the reference scan and one of the new scan under the assumption that they sample the same point in the environment but from different positions. Consider a point $P(x, y)$ in the (x, y) coordinate system and a point $P'(x', y')$ in the (x', y') coordinate system, as shown in Fig. 3.1. The two coordinate systems differ by a rotation φ and a translation, represented by a bias vector $\bar{b} = (b_x, b_y)$.

The problem consists in finding the two factors $\bar{b}$ amd φ that makes the two points P and P' correspond. Using a rotation/translation operator, the point P'_i

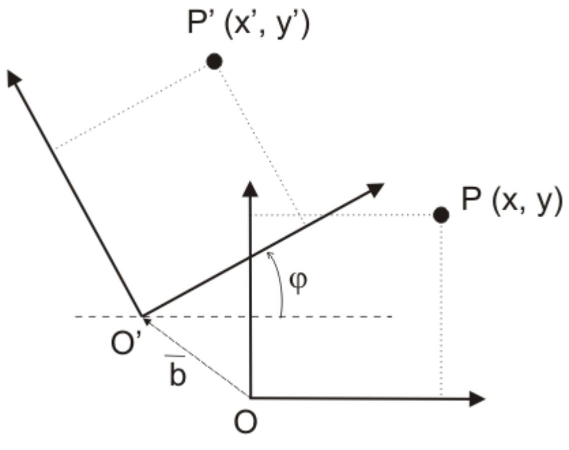


Figure 3.1: Corresponding points of two different scans in different reference frames.

is mapped onto the point P_i as follows:

$$\begin{cases} x_i = x'_i \cos(\varphi) - y'_i \sin(\varphi) + b_{i,x} \\ y_i = x'_i \sin(\varphi) + y'_i \cos(\varphi) + b_{i,y} \end{cases}$$

Normally the φ and $\overline{b}$ terms are unknown and must be estimated. The estimation can be performed by minimizing the error between the i-th points P_i and P'_i:

$$\overline{E_i} = \begin{pmatrix} E_{i,x} \\ E_{i,y} \end{pmatrix} = \begin{pmatrix} x_i - x'_i \cos(\varphi) + y'_i \sin(\varphi) - b_{i,x} \\ y_i - x' \sin(\varphi) - y'_i \cos(\varphi) - b_{i,y} \end{pmatrix}$$

If only one point is considered, however, it is possible to find the optimum bias but it is impossible to determine the rotation. For this purpose, the problem must be turned in a least square problem by considering a sufficient number N of corresponding points. In this case, the square error is the following:

$$E = \sum_{i=0}^{N-1} |E_i|^2 = \sum_{i=0}^{N-1} (E_{i,x}^2 + E_{i,y}^2)$$

Lu and Milios [42] have shown that the result of this minimization error is the following:

$$\phi = \arctan\left(\frac{S_{xy'} - S_{yy'}}{S_{xx'} + S_{yy'}}\right)$$

$$b_x = \overline{x'} - (\overline{x}\cos(\varphi) - y\sin(\varphi)), b_y = \overline{y'} - (\overline{x}\sin(\varphi) + \overline{y'}\cos(\varphi))$$

where $\overline{x}$, $\overline{x'}$, $\overline{y}$ and $\overline{y'}$ are the points averages, and the S terms are covariances. This approach requires that exact correspondences of points are established. In

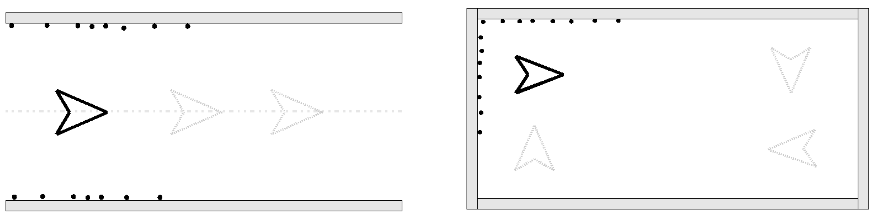


Figure 3.2: Ambiguity problem. The scan may match several features of the reference map.

practise, however, exact point correspondence is impossible to obtain due to many sources of noise, such as mechanical deformation during robot movement, terrain unevenness, occluded areas, sensor noise etc.

To reduce the search space size, most of the scan matching methods used in pose tracking algorithms require an approximate alignment of two scans which is obtained from odometry readings [55][42].

For any scan matching algorithm to work, it is necessary that scans have a sufficient number of overlapping points to be successfully matched. Furthermore, the lack of distinct features in the environment and noisy range measurements may cause the ambiguity problem, where the scan taken at the actual robot position could match several parts of the reference map (Fig. 3.2). One way to consider these uncertainties is to estimate the covariance of the measurements errors, P_k^{ij}. In that case, a more sophisticated method, such as Multi-hypothesis Tracking may be used to maintain more than one possible robot position.

Thus, a high degree of robustness is required for scan matching algorithms to operate in real-life environments.

3.2.1 The sensors used in this work

The scan devices used in this work are rotating focused sonar devices, described in [50], and laser range sensors. Both systems are based on the time-of-flight range measurement principle. However, ultrasonic devices have some advantages over laser sensors: for example, the signal is not hazardous for humans, they can be used in presence of smoke or fog and can detect transparent objects like glasses. In any case, the sensor data must be processed to enhance the signal and extract the information content of the registration. On the other hand, lasers scan devices have better spatial resolution than ultrasonic devices, due to the narrow beam. It is worth noting that ultrasonic sensors have a beamwidth in the order of 10 degrees, and even the focalized beam described in [50] is about four degrees, which is much higher than laser. As a result, ultrasonic readings are less dense than laser's. Generally, the reflection characteristics are also better for lasers than sonars.

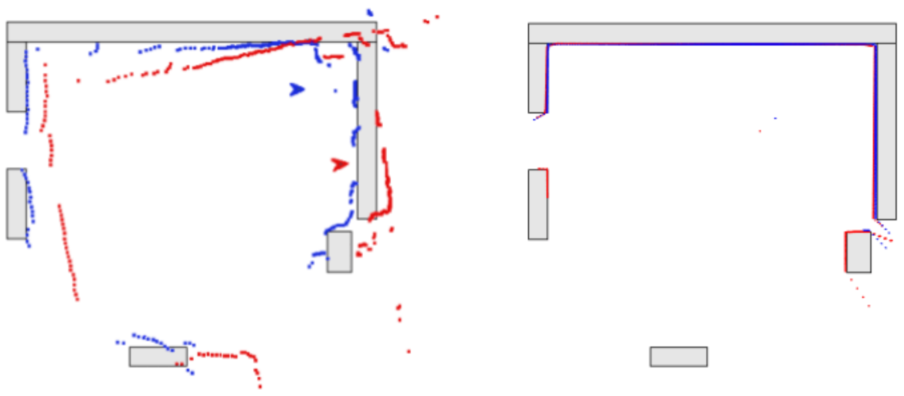


Figure 3.3: Scan matching performed on scans obtained with rotating ultrasonic focused sensor (left) and laser scanner (right). The points in light gray represent raw data and in dark gray the aligned data.

As an example, in the left panel of Fig. 3.3 we report an ultrasonic scan realized with the rotating device of [50] matched against an a-priori map, while in the right panel we report similar results for a laser device of the Sick laser measurement systems family.

Once established the correct functioning of the algorithm with the two types of sensors, the results reported hereafter are based on real acquisitions from the rotating focused sonar device described in [50].

3.3 Related work

The literature on scan matching is very large. In this section we report a brief description of some known approaches which put the results presented in this chapter in the correct context.

Scan matching algorithms based on correspondences can be categorized as:

- Feature to feature correspondences. In this case features like line segments, corners or range extrema [39] are extracted from the actual and reference scans and matched between them. Of course these approaches require the environment to be structured or to contain the considered features.

- Point to feature correspondences. The points of the scan are matched to features such as lines, which can be part of an a-priori known map [7]. Other authors extract features obtained with certain signal processing. Biber in [5] considered Gaussian distributions with mean and variances computed from the point falling into cells of the grid. Also these approaches require the presence of the chosen features in the environment.

- Point to point correspondences. This group of algorithms does not rely on the existence of features in the environment. Correspondences between raw scan points are established and used to directly compute the relative pose between the scans. Thus these algorithms are robust in both polygonal and non-polygonal environments.

Gutmann and Schlegel [25] compared scan matching methods such as Iterative Dual Correspondence (IDC) [41], Cox [7], and Cross Correlation Function (CCF) [67]. In CCF [67] a correlation based approach is used. The orientation of each scan point is found and the circular histogram of these is build. Then the rotation is found by correlating the normal histograms. In COX [7] data points from one scan are matched against lines, prior extracted from the other scan. The survey pointed out that IDC is less accurate than Cox and CCF. Both Cox and CCF, however, can only be used in polygonal environments.

In [26] the solution is searched by performing gradient descent on a score function. Such a function is built by convolving the reference scan with a Gaussian kernel and then correlating it with the sensor scan. This approach has been subsequently refined in [5] by defining a closed form for a potential function that can be minimized using the Newton algorithm. These approaches require a number of iterations that depends on the input configuration and the entity of the error.

In [30] global localization is performed by using a two dimensional correlation operator. This method evaluates every point in the search space and is therefore very robust to noise. It is a global, multi-modal, non-iterative matcher that can work in unstructured environments.

In [38] a method for computing the optimal transformation in a closed-form manner is described, thus eliminating any iteration. This results in a high-speed algorithm, however the laser scans must be continuously provided during the motion of the robot so that two consecutive scans are close to each other, and therefore very similar.

3.3.1 Use of stochastic optimization

The initial optimization-based techniques proposed for point matching methods where based on Iterative Closest Point algorithm (ICP) proposed by Besl and McKay in [4] and then in [12, 70], to cite just a few. Direct-descent techniques like ICP and Gradient Computation approaches [65] are based on calculating gradients of either probabilistic or squared-error [52, 20] cost functions. Among them, the ICP procedure was the most popular because it is simple and effective, thus many variants have been proposed as described in [59]. Recently, stochastic optimization approaches like evolutionary programming [1], simulated annealing [6] and Genetic Algorithms [46, 45], [63], [68] and [40] have been used. These methods introduce a stochastic component in the search for either the correspondence of points [1] or the whole pose [6, 46, 45].

In particular, Yamany et al. [68] perform a registration of partially overlapping 2D and 3D data by minimizing the mean square error cost function. While giving little details about their genetic algorithm, the Grid Closest Point transform (GCP) is introduced for speeding the search for corresponding points.

3.3.2 Iterative point-to-point correspondence algorithms

When unstructured environments are considered, the most popular algorithms are iterative point-to-point correspondence algorithms. Since these algorithms are also considered very fast they represent good candidates for implementation in applications described in the thesis. Recall, in fact, that the goal is to provide robust mapping algorithms for outdoor environments with reduced computational requirements. This section describes them in more detail.

Widely used heuristic methods for aligning 2D or 3D point sets are variations of the ICP [4]. Basically, ICP has three basic steps: first, pair each point of the first set to the second one using a corresponding criterion; second, compute the rotation and translation transformations which minimize the mean square error between the paired points and finally apply the transformation to the first set. The optimum matching is obtained by iterating the three basic steps. However, ICP has several drawbacks. First, its proper convergence is not guaranteed, as the problem is characterized by local minima. Second, ICP requires a good prealignment of the views to converge to the best global solution.

Matching points-to-points is the most general approach which does not require features to be present in the environment. Lu and Milios [42] were the first to apply this approach for localization of mobile robots. They use ICP to compute translations and propose techniques for corresponding points selection. Pfister et al. [55] developed the Weighted Range Scan Matching (WRSM) method which extends the approach of [42] considering also the uncertainty of the estimated motion in order to integrate the scan matching-based motion with odometry information.

Pfister et al. in [55] show that using a maximum log-likelihood approach, the optimal estimate of the robot's translation can be computed using the expression reported in (1),

$$p_{ij} = \frac{\sum_{k=1}^{n} \frac{p_k^i - \hat{R}_{ij} p_k^j}{P_k^{ij}}}{\sum_{k=1}^{n} \frac{1}{P_k^{ij}}} \tag{3.1}$$

where p_k^i, p_k^j are the coordinates of a couple of corresponding points to be matched, $\hat{R}_{ij}$ is an estimated rotational matrix, and P_k^{ij} is the covariance of the error derived from the corresponding points of the two scans. In [55] it is also described a second order iterative estimation of the rotational estimate, to be performed prior to the computation of the $\hat{R}_{ij}$ matrix.

The effectiveness of the algorithm, however, depends upon properly pairing corresponding scan points taken from two different poses. A poor initial displacement estimate results in a poor initial set of point correspondences which are then used to calculate the displacement estimate for that iteration. Subsequent iterations may continuously improve displacement estimates and point correspondences sufficiently such that the algorithm is able to effectively recover, or the algorithm may converge to a local minimum resulting in a localization failure.

3.3.3 Dependency on initial position estimate

A quantitative evaluation of the limits of the scan matching method of [55] has been performed. Since the algorithm is able to recover the correct displacement only if the initial displacement estimate is close to the true position, a series of tests have been executed in order to estimate the region of convergence of the algorithm. The results obtained with laser scanner data are shown in Fig. 3.4 for a medium feature rich test environment. The algorithm is robust if the initial position estimate error does not grow more than approximately 1.0 m distance and ± 0.4 rad rotation. It has been observed in the experiments that ICP derived iterative closest point algorithms typically have same or smaller local convergency area.

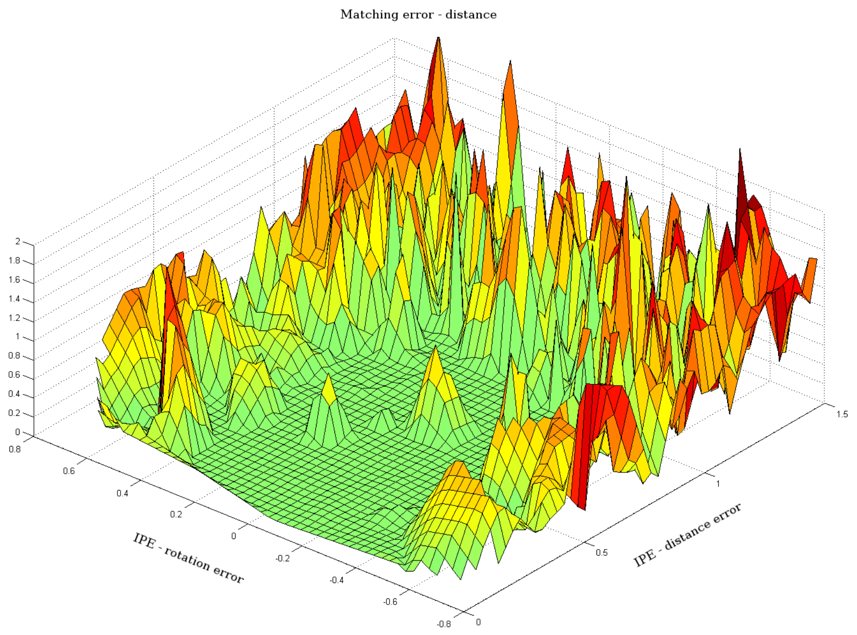
The results are shown in Fig. 3.7 where the region of convergence is the flat area close to the true position.

3.4 Implementation issues

3.4.1 Outliers removal

Consider the range scans in Fig. 3.5 from two different robot positions. The scans overlap only partially in the lower part of the environment, the other parts being occluded by obstacles. Given the two scan sets the outliers are defined as the points visible in one scan, but not in the other. In Fig. 3.5 all the points not in the lower part of the environment are outliers. If an approximate initial position estimate is given it is possible to remove the outliers from matching considerations.

The removal of outliers speeds up considerably the search for correspondence point pairs and therefore the matching process. It also increases the ability to successfully match corresponding points from the range scans since wrong correspondences in which one of the points is an outlier are avoided.



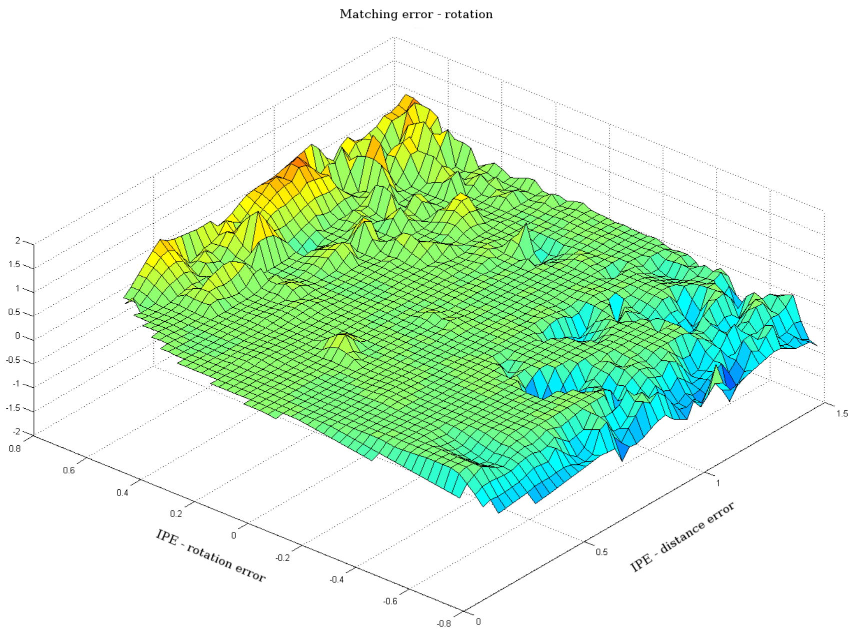


Figure 3.4: The robustness of the WRSM algorithm with increasing initial position estimate errors. Matching error in meters (Z axis) is depicted against the initial rotation displacement (X axis) and initial distance displacement (Y axis). Failures to converge increase outside of the flat area of approx < 1.0 m distance and ± 0.4 rad rotation. The flat area represents the region in which the WRSM algorithms converges most of the times.

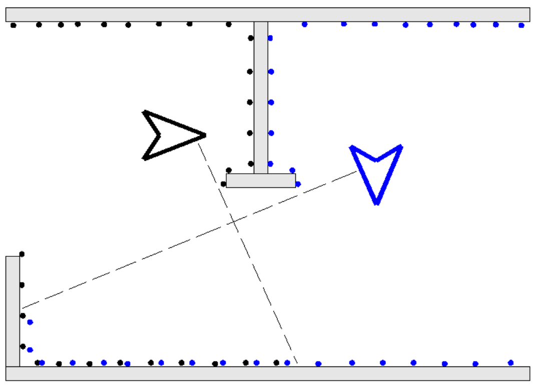


Figure 3.5: Outliers.

In [42] the authors remove the points which after the projection of the reference scan in the coordinate frame of the new scan either change orientation or become hidden by the other scan. In [55] use the same approach.

In this thesis this technique is further extended. Outliers are removed from both scans by projecting one scan in the coordinate frame of the other and then, in addition to applying the two criteria as in [42], also removing the points that after the projection result out of the reach (given the sensor range) and hidden by points of the same scan.

3.4.2 Resampling the scans

Since the scanning of the environment if usually performed by rotating the sensor at fixed step angle, the angle between scan readings is constant and the environment features that are closer to the robot will result in more dense scan points that those further away. Depending on the algorithm, the denser areas produce more correspondences and/or give a higher fitness value distorting the matching process. To avoid such an artifact the scans may be resampled with a constant resampling interval. Many authors however resample the scans with the primary goal of reducing the number of points in a scan and filtering the noise. The former significantly increases the speed of the algorithm while the ladder avoids coping with noise. The benefits of resampling come at the cost of the information loss.

3.4.3 Correspondence searching

Matching points is the most computationally intensive step in all the matching algorithms that rely on it and since the vast majority of algorithms do rely on it, it's importance cannot be stressed enough. In 3D image matching field many different approaches have been proposed together with optimization techniques in the implementation. They are best synthesized in the survey presented in [59].

In 2D scan matching literature in robotics, however, the details of how the proposed techniques were implemented in a computer program are often skipped. However minor differences in the implementation of the same technique may have a large effect on the algorithm's performance. In order to compare the algorithms to each other it is important to state clearly which building blocks were used in the experiments.

During the development of the scan matching algorithms described in this chapter, several correspondence search methods, mostly newly developed, were implemented and analyzed.

A generic correspondence search algorithm takes the points from the two scans at poses i and j respectively, $p^i_{k_i}$, $k_i = 1..N_i$ and $p^j_{k_j}$, $k_j = 1..N_j$, and selects a set of corresponding points p^i_k, p^j_k, $k = 1..N_{ij}$.

A straightforward and fast algorithm for establishing point correspondences between two scans simply considers the polar coordinates of the reference scan points and new scan points projected in the same coordinate frame of the reference scan. The scan is then traversed with increasing angle and points that belong to the same angle step which are closer than distance threshold are matched. This 'polar coordinates' approach has been used in [45] in their hybrid two phase genetic + ICP approach for the genetic phase. However the experiments have shown that using this simple approach in iterative correspondence point algorithms leads to failures to converge and poor performance.

In their original work Besl and McKay used a Closest-Point Rule and they proved that the ICP algorithm always converges monotonically to a local minimum with respect to the least squares distance function [4]. Lu and Milios use a different rule to reveal the rotation component of the scan in the hope of improving the speed of convergence, see [42] for details. In this section a faster correspondence search algorithm is proposed.

The algorithm is devised with the following properties for the selection of the corresponding points:

- given two sets of corresponding points with the same number of pairs, the one with the smaller sum of distances between corresponding points is better.

- the set of corresponding points should be statistically neutral with respect to directions of the correspondence vectors (the segment joining the two points). The experiments have shown that the slightest preference in the directions, due for example to partial search or direction of scan traversal, leads to a failure to match scans.

- without intersections of correspondence vectors

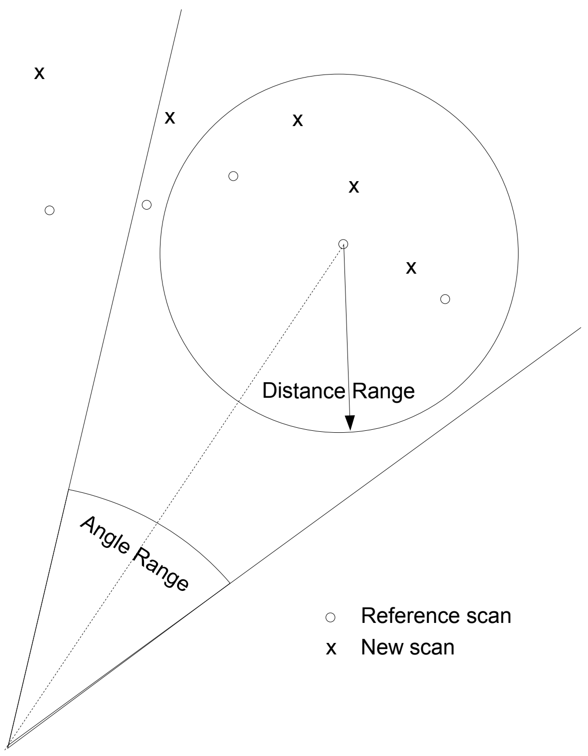


Figure 3.6: The corresponding point search. A range and angle criterion are used to reduce the search space. A corresponding point for the reference scan reading in the center of the circle is selected from the readings of the new scan. The one with the minimum distance is selected.

The optimal solution for the problem can be obtained using the following algorithm:

1. form all the possible pairs (p_k^i, p_k^j) between points ($N_i \cdot N_j$ pairs) and compute their distance;
2. sort in ascending distance order;
3. pick the corresponding points pair with minimum distance (p_k^i, p_k^j). Delete correspondence point pairs $(p_k^i, *)$ and $(*, p_k^j)$, where $*$ means any point;
4. repeat step 3 until all the corresponding pairs within the distance range have been picked.

The optimum approach is computationally very intensive, therefore a more efficient algorithm is proposed.

To speed up the search, an angle threshold during reference scan traversal is used:

```
CYCLE=1
repeat
  for each point  Pki
    compute the distance to each Pkj in Angle Range;
    select the Pkj with minimum distance not already paired in
      cycle < CYCLE and mark both points as paired;
    if Pkj was already paired in the same cycle and the new distance
      is smaller, the new pair is marked releasing the previous one;
  CYCLE++;
until (all the correspondences have been found)
```

Since each cycle decreases the number of points to consider, the speed of the next cycle increases. The majority of the correspondences are found in the first cycle so a threshold may be set on the maximum number of cycles to perform and further increase the speed. Nevertheless it is not appropriate to limit the search on the first scan only since particularly unfortunate combinations of points may occur where points already paired are released in favor of even better correspondence in the same cycle. This correspondences for this points may be found in the next cycle.

In the implementation the readings in the scan are ordered by the angle and pointers to the first and last reading in angle range are maintained, so a single fast scan is performed for each cycle.

3.5 GLASM

In this Section a new algorithm is proposed called GLASM (Genetic Look-up based Algorithm for Scan Matching). It aims at finding the (x, y) translation and the rotation φ that obtains the best alignment between two different scans. The algorithm is based on the computation of a lookup table which divides the plane of scan readings in a grid for a rough but fast reference point look-up as will be shown next in the description of the fitness function.

Each parameter of the scan position $(x,\ y,\ \varphi)$ is coded in the chromosome as a string of bits as follows: nbitx for coding x, nbity for y and nbitrot for φ. The problem space is discretized in a finite solution space with a resolution that depends on the extension of the search area and on the number of bits in each gene. The search area limits can be set based on the problem at hand. In the case of pose tracking where odometry measurement are available they are usually set on the basis of odometry error model.

The positional information is coded in the genes using the Gray code. The inverse Gray code value of the bit string is taken to obtain the position from the gene. In this way the variations in the bit string caused by the mutation or crossover operators translate in the proportional variations in the position they represent. Using a simple binary code a change in one bit may cause a significant

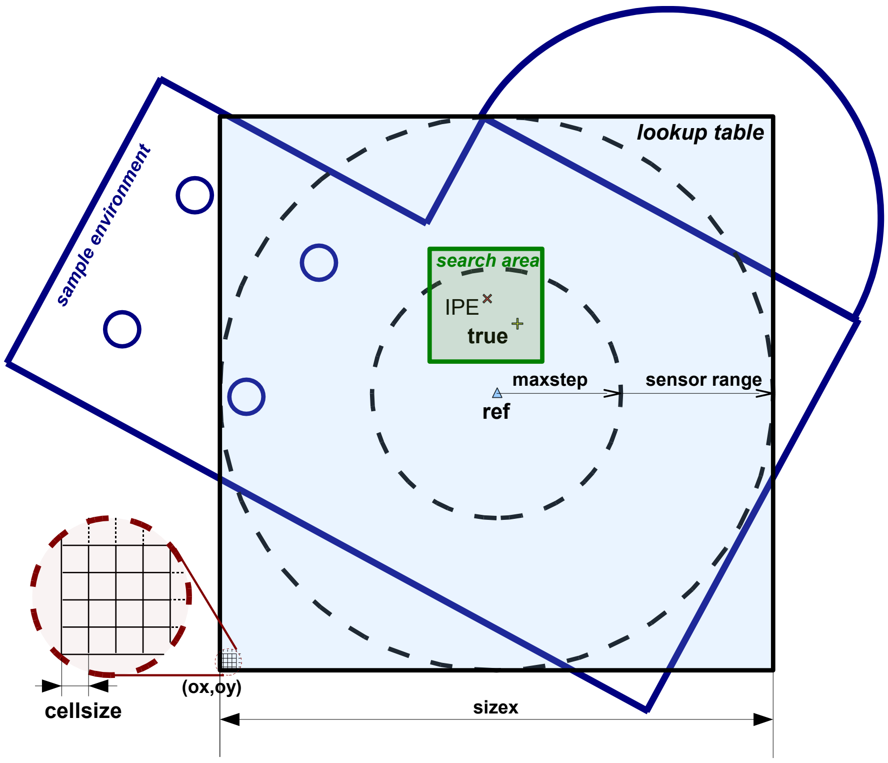


Figure 3.7: Look up table and search space size in local scan matching step.

variation in the position. It was found that for the scan matching application, simple binary code leads to a reduced efficiency of the genetic algorithm.

The genetic algorithm starts with a population of Popsize chromosomes randomly selected with uniform distribution in the search space. Each individual represents a single position of the new scan. An example of the matching process is shown in Fig. 3.8.

The goal of the scan matching is to estimate the position of the new scan relative to a reference scan or a given map which is best fitted according to a fitness value.

3.5.1 Fitness function

Fitness computation must be done very quickly and it must provide a value which is closely correlated with the degree of matching between two scans. Typically the used fitness function is formulated by accumulating matching errors and normalizing it by the number of valid corresponding points. The most fitted values, those that point to a better overlap of two scans, are the points with a smallest cumulative matching error. In GLASM a different approach has been

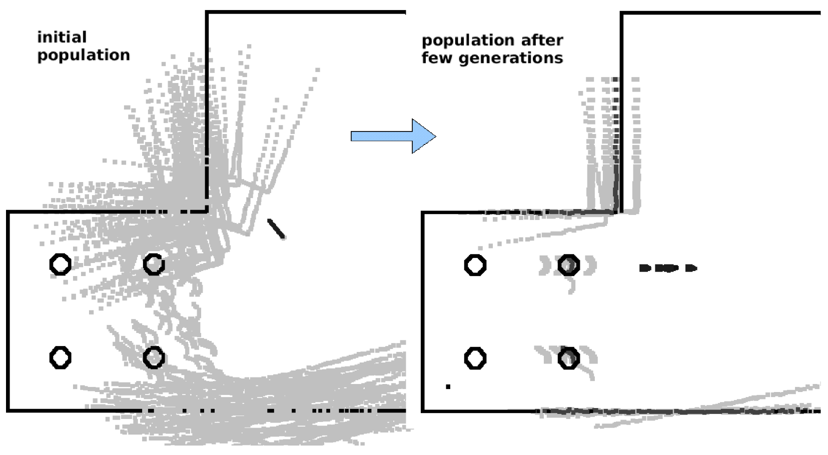


Figure 3.8: The reconstruction of the scene.

used.

As soon as the reference scan is available, a look-up table is created as depicted in Fig. 3.7 and Fig. 3.9. Each cell represents a portion of the 2D plane of scan readings. The overall dimensions of the lookup table cover all the search space.

In the case of pose tracking the lookup table is centered in the reference scan position. In order to cover all the possible sensor readings of the new scan it is at least as large as the step size plus the sensor range. For position estimation with an a-priori given map the lookup table should at least cover the map. Each cell in the table is marked with a boolean value 0 or 1. The cells of the table are initialized with 0 and only the cells close to a reference scan point are marked with 1.

The genetic algorithm evaluates the fitness value for a chromosome as follows: for each point of the new scan a direct lookup is made in the lookup table to establish if the cell corresponding to the point has 1. The number of points of a new scan having a corresponding point in a reference scan is then taken as a fitness value for this chromosome, i.e. for the new scan evaluated in the position coded in a chromosome. This in fact is directly proportional to the overlapping of two scans, i.e. the degree of matching between the two.

This way, there is no need for the correspondence function since no pairings need to be done between scan points. There is also no need for the matching error function since the fitness value is directly obtained by counting positive lookup returns. The speed-up gain of our approach comes at the cost of building the initial lookup table and of a potential reduction of matching process' accuracy posed by quantization errors introduced by the lookup table. However the ex-

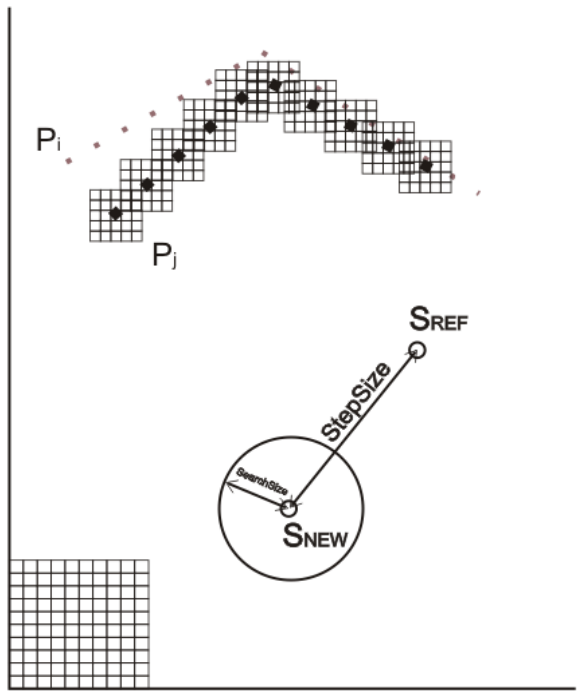


Figure 3.9: The environment is discretized and a look-up table is built. The cells near the points of the reference scan are depicted. These cells are the only ones to be marked in the look-up table.

periments presented in this paper show that both of them are negligible. In fact the creation of the initial lookup table is very fast and the accuracy reduction is lower or comparable to the resolution of the finite solution space, so the overall effect is negligible.

In order to avoid some areas carrying more points than others the scans are resampled. This is because a scan of an object close to the sensor has denser readings than one far away. The fitness function would otherwise count a higher score when overlapping dense areas containing more points. This would reduce the efficiency of the matching process.

On the basis of the above considerations, it is clear that we define our basic fitness function as the sum of the squares around each point coming from one scan that intercepts the point in the new scan:

$$f_{S_{NEW},S_{REF}} = \sum_{i=1}^{N} \rho(i) \tag{3.2}$$

where S_{NEW} and S_{REF} are the two scans to be matched, N is the number of points in S_{NEW}, and the value of ρ is as follows:

$$\rho(i) = \begin{cases} 1 & \text{if a point of } S_{NEW} \text{ lies in the square around a point in } S_{REF} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

3.6 Experimental results

The GLASM algorithm has been tested with real scan data obtained with a focused ultrasonic sensor [50]. Furthermore, for an in depth analysis of success ratio, search space size, accuracy and speed of the algorithm a scan matching simulator, expressly developed for that type of sensor, has been used. The use of a simulator is the only possible approach which allows to run a huge number of scan matching tests in a controlled testing environment with scans taken from arbitrary positions in arbitrary environments. The simulator also guarantees the knowledge of the true position from where the scan was made thus enabling the exact estimation of the accuracy and the success ratio of the matching process. With real scans the true position must be measured by hand or by a more accurate localization method and this is not a simple task and is error prone. With the use of a simulator it is also possible to save the scans and repeat the very same series of tests while varying the parameters of the algorithm and observing the effect on performance. This is especially important when performing comparisons with other scan matching methods. The algorithm's performance regarding the success ratio was investigated first.

3.6.1 Definitions

Some terms used in this chapter are defined in this section.

- Successful Matching: a matching that results in an estimated position within the ellipsoid centered in the true position $(x_{true}, y_{true}, \theta_{true})$. A successful matching yields a position estimate close to the true position.
- Success Ratio (SR): ratio between the number of successful matchings (N_{Msucc}) and total matchings (N_{Mtot}) performed in one set of localization trials:
$$SR = \frac{N_{Msucc}}{N_{Mtot}}.$$
It measures the ability of the algorithm to converge for a given environment and a set of scans.
- Maximum Success Ratio (SR_{max}): a maximum possible success ratio for a given environment and set of scan pairs. As a matter of fact, in the experiments SR is rarely equal to 1. In other words, there is always a

maximum success ratio SR_{max}, $SR_{max} \leq 1$, for a given environment and a set of scans regardless of the matching algorithm used.

- Accuracy: it is easily determined for particular test conditions if the true scan positions are known. In this case we use the average and the variance of the position estimate error for both position and rotation. In the experimental results reported further in Section 3.6, only successful matchings were used for the computation of accuracy. It is worth noting that an algorithm with a higher SR includes in the computation of accuracy cases for which the others failed to converge.

- Fitness function hit area: the area in the lookup table including cells which are less distant than the distance threshold from a reference scan point or known map. These cells are marked for a fast look-up by the fitness function.

Recall that a matching is considered successful if the estimated scan position lies within a range from the true position. The radii of the ellipsoid in the implementation were 0.1 m on the x and y axis and 0.1 rad for the rotation, i.e. the matching is to be considered successful if the resulting position is at least 10 cm near the true position and rotated less than 0.1 rad from it. In order to obtain a success ratio value which is less dependent on a particular set of scan pairs used in the tests, different matchings from many randomly chosen scan pairs were performed.

In this way the success ratio may be considered a characteristic indicator of the algorithm's matching performance for that particular environment and sensor. For each obtained pair (S^i_{REF}, S^i_{NEW}), the new scan was further randomly displaced several times from the true position. That is because the initial position estimate (IPE) is the starting point for the iterative algorithms search and since different IPEs for the same scan pair may converge or diverge differently, several randomly chosen IPEs were considered. The IPE is not used by genetic algorithms, however the center of the search space for genetic algorithms was placed in the IPE just like in real robot exploration. This implies that for different IPEs the genetic algorithm searches in different areas. A single test cycle in the experiments therefore consisted of a total of $N_{Mtot} = N_{pa} \cdot N_{ip}$ matchings, with N_{pa} the number of different randomly chosen scan matching pairs and N_{ip} different IPEs of the new scan. For each test cycle the success ratio, the accuracy and the average matching time were calculated. The accuracy of the algorithm was calculated by measuring average and variance of the displacements from the true position. The execution time was measured on a single thread of a Pentium 4, 2.8 HT enabled computer.

3.6.2 Determination of the parameters of the genetic algorithm

The parameters of the genetic algorithm were chosen with the goal of maximizing the success ratio. A new test cycle was performed for different configurations of algorithm parameters. The mutation and crossover probability were then chosen from observation. The population size and the number of generations performed should be chosen as low as possible. The lower the population size and number of generations the less computations must be performed and therefore the faster the algorithm. However the success ratio for a given environment and search space size starts to drop when Popsize and generations become too low. The algorithm's parameters are summarized in Table 3.1.

Genetic parameters		**Lookup table**	
max generations	variable with search space size	Lookup table size	variable with search space size
population size	variable with search space size	Cell size	2x2 cm
Chromosome length (X, Y, θ)	$6 + 6 + 6 = 18$ bit	Fitness function hit area (distance from ref. scan point or map)	9 cm
Crossover probability	1		
Mutation probability	0.00925926		

Table 3.1: The values of GLASM parameters used in the experiments.

3.6.3 Implementation details

Different GLASM implementations have been tested and compared with each other. A comparison between Gray and Simple Binary encoding variants was performed. In the same test conditions and for the success ratio of SR >=90% the Gray variant required less generations and a smaller population for the same performance. Moreover the repeatability of the performance is better. Furthermore there is no perceivable gain in the speed for the Simple Binary variant since the conversion in the Gray code is memory mapped with negligible computation time. Therefore a Gray variant has been chosen for the experiments presented in the following.

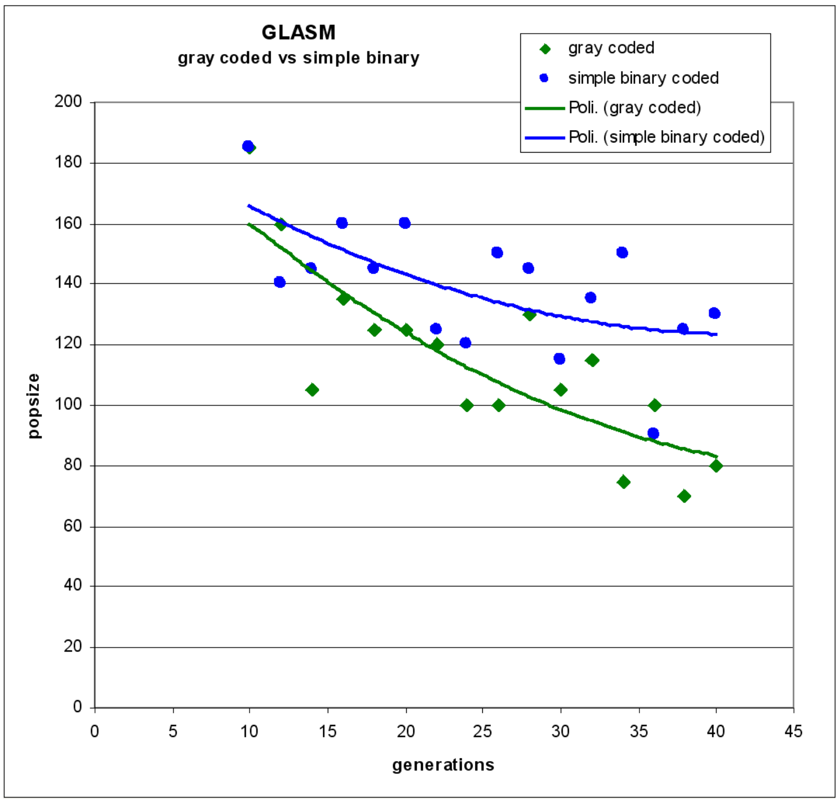


Figure 3.10: Comparison between a simple binary and Gray coded GLASM. Population size and number of generations which yielded success ratio above 90% for the same environment and set of scan pairs.

3.6.4 Comparisons

The GLASM algorithm has been compared to a pure 2D point-to-point version of the ICP algorithm (without tangent line calculations), as reported in [45]. Two versions of the most computationally expensive part, the correspondence search algorithm, were used and compared. The first one is a classical implementation which computes the squared distances for every possible combination of reference and current scan points, the second is the optimized version described earlier in 3.4.3 which yields a speedup of approx $4x$ without any noticeable reduction of the algorithm's accuracy. The speed of this faster ICP implementation is reported later in the document and compared with GLASM. Special care was taken in order to implement the best and highly optimized versions of the algorithms.

Comparison with ICP

For a given set of scan pairs and search space size, iterative correspondence point algorithms [4, 55] do not have the flexibility of varying some algorithm parameters in order to perform a more thorough search when necessary. Genetic algorithms on the other hand can accomplish better success ratio and accuracy by increasing the population size and generations. The success ratio eventually converges towards the maximum success ratio for that environment and that particular set of scans.

Genetic algorithms start the search process by distributing an initial population in a given search space. Iterative algorithms based on point correspondences on the other hand start the search process from the initial position estimate which must be available. Depending on how far and how misaligned this estimate is they converge or fail to do so. Therefore the performance of these algorithms is satisfactory only inside an area close to the true position (approx < 1.0 m distance and ± 0.4 rad rotation, see Fig. 3.4).

ICP is therefore a local matcher and is quite different from GLASM which can be used both for global and local scan matching and can scale easily with different search space sizes. Nevertheless, since ICP is wildly used in local scan matching and its performance is well known, it is interesting to compare the accuracy, speed and success ratio of the two algorithms for local scan matching with small search space size where ICP performs well.

Tab. 3.2 shows the results of a test cycle of 450 matchings performed for two different search space sizes inside the convergence region of the ICP algorithm. In the first case random positions were placed on the circle distant 0.2 m from true position and the rotation was chosen $+0.2$ rad or -0.2 rad at random. The test was then repeated for 0.4 m and 0.3 rad range.

			Position error		Rotation error	
	SR	time (ms)	avg (m)	var	avg (rad)	var
ICP	87%	103	0.01500	0.00430	0.00019	0.00002
GLASM	92%	22	0.04000	0.01100	0.00042	0.00012
ICP	79%	159	0.01900	0.00715	0.00044	0.00008
GLASM	93%	23	0.04300	0.01068	0.00045	0.00010

Table 3.2: Comparison of GLASM with ICP for local scan matching.

In both cases GLASM has a better success ratio than ICP. This is true in general and has been verified for other environments and search space sizes. ICP has a slightly better accuracy than GLASM which was expected. In small search space in fact, a reduced population size and number of generations are used. In fact for a small search space size GLASM achieves high success ratios already with a population size of 30 and after only 5 generations. Moreover GLASM is limited by the discretization of the GLASM lookup table and the size of the fitness function hit area. As for the speed, for a fixed population size and a number of generations, the genetic algorithms have constant processing time regardless of the convergence. For iterative algorithms the speed varies between iterations due to correspondence points pairings and number of performed iterations. The stop criteria is set when the convergence slows down to a point that there is little variation between successive iterations. More precisely the process stops if at least 5 initial iterations have been performed and in the last 3 iterations rotational and position displacements are less than a threshold, otherwise it continues until a maximum of 50 iterations. It is clear that ICP's failure to converge costs time. The results show that GLASM is at least $5x$ faster than ICP while still having a better success ratio. This is quite remarkable since ICP is considered a very fast algorithm.

Comparison with GCP

GLASM has also been compared to an existing 2D genetic optimization technique for scan matching, the GCP transform algorithm [68] which is classical in the area. The GCP's fitness function is based on corresponding points pairings between a reference scan and new scan point. To speed up the fitness evaluation, the Correspondence Grid is initially created with a discretization of the plain containing the scan points which calculates the displacement vectors from each cell to the nearest reference scan point. That way, instead of performing a search for correspondences for each fitness evaluation, a simple lookup in the Correspondence Grid yields the displacement vector. The initial cost of building the Correspondence Grid is small if compared to a high number of fitness evaluations performed during a matching process, which is usually the case.

The main differences of the two genetic algorithms are as follows.

1. the GCP transform calculates a displacement vector, while GLASM does not.

2. this calculation is done for each cell of the grid while GLASM only marks (without calculation) a very limited number of cells around reference scan points (model set).

3. the GCP transform requires a higher grid resolution for a good approximation of closest point. On the contrary GLASM does not calculate a displacement vector (no distances are approximated) so it does not suffer from displacement vector quantization error.

4. GLASM does not use correspondence function and matching error function, the fitness value is provided by direct count (direct lookup). A simple fitness function assures faster and straightforward calculation.

This time a medium and large search space sizes were used in the experiments (Fig. 3.11, Tab. 3.3) so a larger number of generations and a bigger population size were necessary to maintain a high success ratio for the environment.

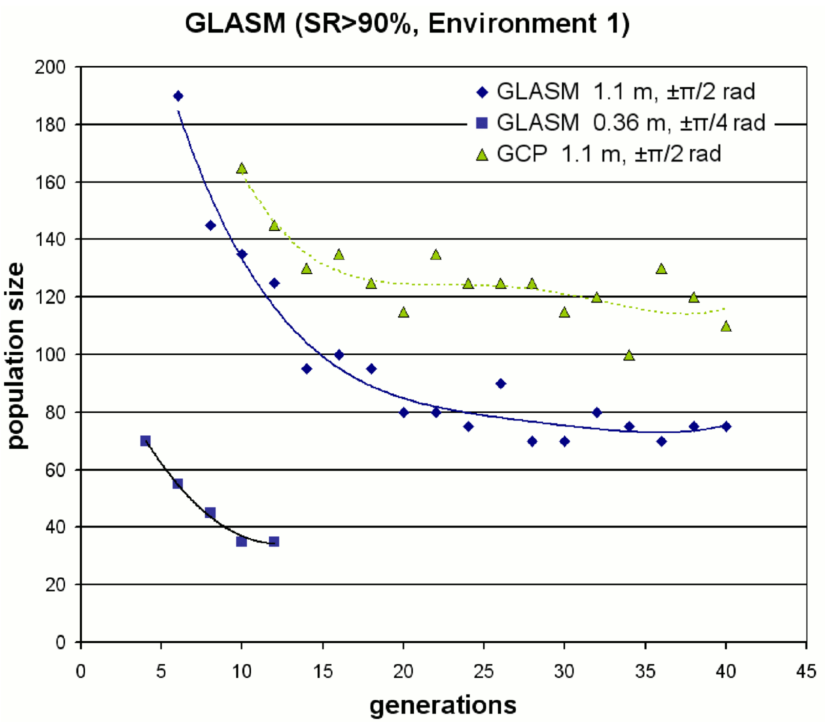


Figure 3.11: The number of generations and a population size needed to reach a success ratio above 90%. Two series of data are plotted for GLASM algorithm corresponding to a search space size of (±0.36 m,±π/4 rad) and (±1.1 m,±π/2 rad). One data series (in lighter color) is plotted for GCP algorithm for the (±1.1 m, ±π/2 rad) case.

The test results have shown that under the same test conditions (same environment, set of scan pair positions, scans) GLASM yields a higher success ratio for the same number of generations and population size, i.e. it converges in cases where GCP does not. The accuracy of the algorithms is analyzed in Fig. 3.12. Each darker point in the graphs corresponds to the average error in position (left graph) and rotation (right graph) for a series of 150 matchings (one test cycle), while the lighter points are the variance. Only successful matchings are considered in computing the averages and variances.

The average position and rotation errors are more or less constant with population size, but decrease with the number of evolved generations. For GLASM

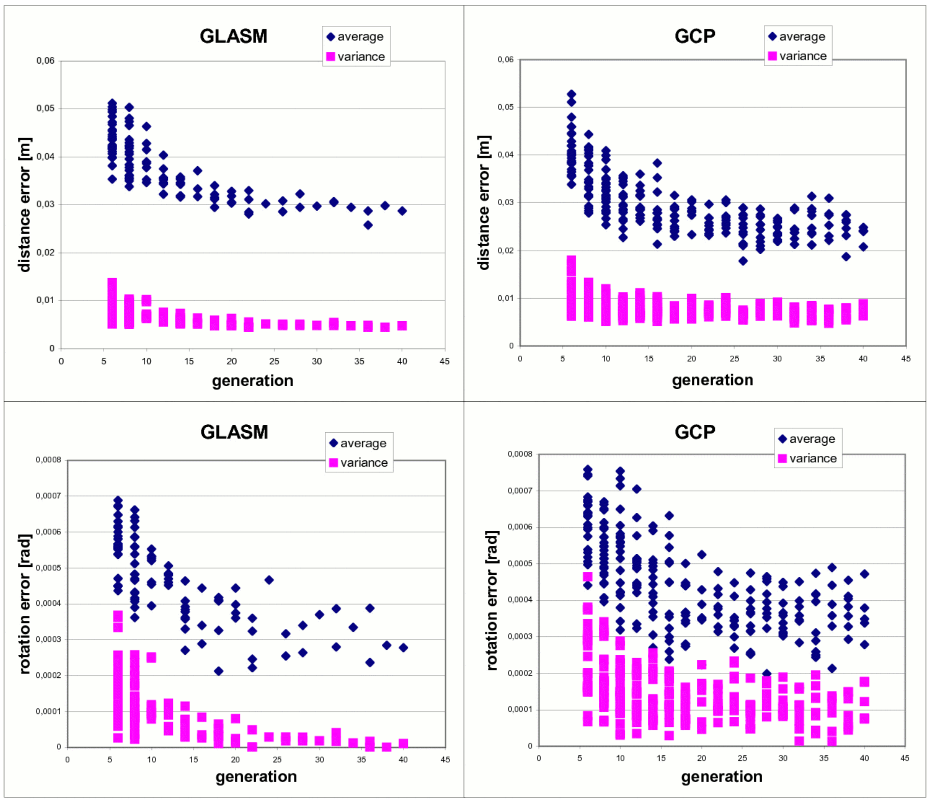


Figure 3.12: Accuracy comparison between GCP and GLASM.

the average position error finally settles to a distance of approximately 3.4 cm from the true position and rotation error to about 0.00027 rad from the true scan angle. GCP settles down to 2.9 cm and 0.00048 rad.

GLASM and GCP clearly differ as to the variance. As shown in Fig. 3.12, GLASM has a lower variance than GCP, and these experiments present a great repeatability. These characteristics lead to a feasible implementation of GLASM even with strict computational time constraints.

Let us now analyze the speed of the algorithms. Both algorithms speed up the fitness function evaluation by building a look-up based grid in memory as soon as the reference scan is known. Table 3.3 compares the time spent to prepare the grid. Two sizes of the grid are considered, one suitable for a medium size environment (up to 100 m^2), the other for a larger environment (up to 250 m^2).

The overall processing time is thus a sum of the time spent in the preparation of the correspondence grid which is performed only once for a single reference scan or known reference map and the processing time of the genetic algorithm for the matching of the new scan.

	Preparation of the correspondence grid	
	(medium size environment $< 100\ m^2$) 1000x1000 cells	(large environment $< 250\ m^2$) 5000x5000 cells
GCP	19000 ms	several minutes
GLASM	4 ms	110 ms

Table 3.3: Average time spent in the preparation of the correspondence grid (GCP) and look-up table (GLASM).

In [68] it is suggested that the speed may be improved by selecting small cell size in the region directly surrounding the model set and a slightly larger value for the rest of the grid. In typical robotic applications and with reference to Fig. 3.7 the size of the grid is just slightly bigger than the enclosed environment, so the suggested approach could result in modest improvements in speed but with greater implementation complexity.

When building a lookup table GLASM only updates a very limited number of cells around reference scan points (fitness function hit area) leaving the vast majority of cells intact. The fitness function evaluation is also much faster since GLASM does not search for correspondences and it does not compute a matching error with the set of found corresponding points.

The average execution time for a single matching (test cycles with $SR >=$ 90%) is: GLASM: 350 ms; GCP: 461 ms. This is an improvement of 24% considering only the matching phase.

3.7 Final remarks and conclusions

In this chapter we have described and discussed a Genetic based optimization algorithm for aligning two partially overlapped scans. A novel fitness function has been proposed and experimentally evaluated. The fitness function gives the genetic algorithm several important properties, namely does not require to compute any point to point or feature to feature correspondence and is very fast, because it is based on a look up table.

Instead of searching for corresponding point pairs and then computing the mean of the distances between them, as in other genetic algorithm's fitness functions, the fitness is directly evaluated by matching points which, after the projection on the same coordinate frame, fall in the search window around the previous scan. Hence, the fitness function is quicker than the usual approaches based on obtaining a point assignment between the two scans and then computing the mean of the distances between them. In fact, it has a linear computational complexity $O(N)$, whereas GCP and ICP corresponding phases have a quadratic cost of $O(N^2)$.

It might be argued that the limitation of the fitness function which counts the number of points from the new scan lying in the square around a point in the reference scan is that, for example, a candidate solution with every new scan point located very close to its corresponding reference scan point, but immediately outside their squares (i.e. a good quality solution) would have a 0 fitness value, while another solution where a single new scan point is within a square of any reference point, even not being the right one, will have a fitness of 1, even being a very bad solution.

In practice, however, this limitation does not affect the matching process for the following reasons: the candidate solution just outside of the squares is noisy so part of the readings do fall inside the squares proportionally to the distance from the reference scan[*]. Second, both the number of points in a scan and the density of population in the genetic search are very high so the effect of these unfortunate cases is vanishingly small and filtered by the selection of best fit individuals.

Moreover both the point assignment errors introduced in correspondence search and the quantized displacement vectors used in GCP are sources of error which are absent in GLASM.

The experiments have shown that the limitations to the overall accuracy posed by the finite solution space (set by a number of bits used to code a position in the genes and by the extension of the search space) and quantization of the fitness function as defined with a lookup table, are small if an appropriate size of the marked area is chosen for the problem at hand.

Let us turn now to the memory needed for the implementation of the look-up table. The look-up table is a 2D array requiring 1 bit of information per cell. For a scan matching in robotic applications with sonar and/or laser scanner, typical sensor ranges are between 5 to 50 meters. With an exploration step size of several meters a cell size of 10 cm or less should be appropriate. To cover a map of 100x100 meters a 1 million cells array is necessary which only requires 125 kb of memory.

The GLASM algorithm scales easily on multiprocessor systems. The fitness function evaluation may be easily distributed across the available CPUs to further increase the speed.

The experiments performed with the GLASM algorithm show that searching in the solution space with genetic optimization and fitness function using neither corresponding points selection nor matching error calculation, moreover based on a simple direct spatial relation is the right approach for this type of problems.

[*] just like a small addition of random noise (dither) improves the performance of analog to digital conversion.

The algorithm does not require an initial position estimate, it is fast, robust and suitable for unstructured environments.

Chapter 4

Real-time Operating Systems

4.1 Introduction

Embedded systems are hardware and software devices designed to perform a dedicated function. They usually do not allow users to build and execute their own programs. The speed, the size and the cost of the device are important factors for embedded devices controlling a physical system, such as a robot. Since a mobile robot operates in a real world, to be able to react to external events in a timely fashion, the embedded system should be operated under the control of a real time operating system. If the embedded system installed on the robot executes critical tasks like those dedicated to controlling the motion of the robot or data acquisition from the external world, hard real time operating system should be used. It guarantees that a deadline is not missed. This is the reason why mobile robotics and many other fields need a real-time operating system running on processors or controllers with limited computational power.

The motivation to carry out the research on the real time operating system as presented in this chapter stems from the fact that many research topics presented in this thesis require validation through a real application, which is based on a real world data, not limited to simulated environments. The goal is to realize small autonomous embedded system for implementing real-time algorithms for non visual robotic sensors, such as infrared, tactile, inertial devices or ultrasonic proximity sensors, as described in previous chapters and in the Appendix A (see also [50, 33])

In order to operate in unstructured environments, the raw sensorial data gathered from non visual sensors, like ultrasonic or inertial sensors, are processed to obtain a representation of the perceived environment. However, the robot is controlled by a processor with limited computational power due to the limited power supply of the mobile system. The motivation is to provide a small, autonomous embedded system which carries out the processing requested by non visual sensors without imposing a computation burden on the main processor of the robot. In particular, the embedded system described in this thesis provides the robot with the environmental map acquired with the ultrasonic sensors.

With reference to the afore stated needs, the embedded system has the following requirements: to enable real-time operation with non-preemptive scheduling, preferably with a deferred interrupt mechanism, and to have low footprint and low overhead. Furthermore, the system should be open-source and should run on the Avnet M5282 board installed in the robot which is based on a Coldfire® MCF5282 microcontroller, i.e. the operating system must support the Coldfire architecture.

The board is shown in Fig. 4.1.

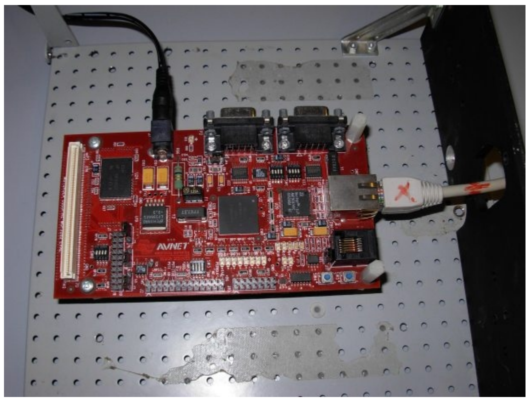


Figure 4.1: The board with a Coldfire MCF5282 microcontroller

The requirement on non preemptive scheduling policy will be thoroughly explained in the next section 4.2, whereas the other requirements are rather obvious for an embedded system. The argument that follows is intended for a uniprocessor system if not noted otherwise.

There are many different open source real time operating systems (RTlinux, RTAI, FreeRTOS), however a quick research showed that considered operating systems were in most cases unavailable for the specific target architecture and had no support for a non preemptive scheduler.

Therefore the choice was either to port one of the available systems on the Coldfire embedded platform and then to modify it to suite the stated requirements, or to develop a new real time operating system.

Since there was a tiny operating system available in Smartlab described in [51], it was decided to continue the development by modifying it's scheduling module.

The new system was called Yartek (Yet Another Real Time Embedded Kernel). It uses a small amount of resources and is suitable for running on micro-controllers. The source code is available on-line [69].

The operating system was developed in a collaborative work. My contribution consists in creating the theoretical results on non-preemptive scheduling, porting the RTAI real time system on the uClinux enabled Coldfire embedded platform, implementing the testing procedures and performing the comparison between the two systems. I also contributed in the development of the application.

This chapter is structured as follows. Section 4.2 summarizes the scheduling policies used in the kernel and proposes a design methodology useful for non-preemptive tasks. Section 4.3 describes some technical aspects in the Yartek architecture. Section 4.4 deals with the performances of this implementation. Section 4.5 reports a case study where the non-preemptive design methodology has been applied. Final remarks are discussed in Section 4.6. Some parts of source code are reported in the Appendix B.

4.2 Non-preemptive real-time scheduling

4.2.1 Preliminaries

The most basic service of a real-time operating system kernel is task management. It allows to design a software application as a number of separate "chunks" of software, called "tasks", each handling a distinct goal, usually with its own real-time deadline. Task management handles the tasks execution and assigning priorities to them. Its main service is to schedule the tasks as the embedded system is in operation in a timely and responsive way.

Preemptive vs Non-preemptive scheduling policy

The online algorithms that actually schedule a task set running in the system can use either preemptive or non-preemptive scheduling. In preemptive scheduling the algorithm is allowed to actually preempt a task that is currently being executed and allocate a processor to another task (Fig. 4.2). Non-preemptive scheduling gives a task a full quota of the requested processor time each time the task is scheduled.

Today nearly every RTOS employs a priority-based preemptive scheduler. In a priority-based preemptive scheduling each task in a software application must be assigned a priority, with higher priority values representing the need for quicker responsiveness. A quick responsiveness for high priority tasks is accomplished by using preemption.

This type of scheduler was thoroughly studied from the 1970's up to the first years of the 1990's [14, 61]. The theory was generalized to the point of being

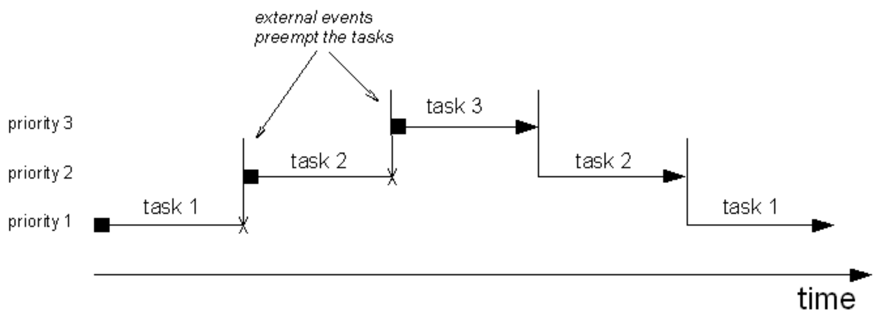


Figure 4.2: Preemptive scheduling example: the scheduler may stop any task at any point in its execution, if it determines that another task needs to run immediately. If both a low-priority task and a higher-priority task are ready to run, the scheduler runs the higher-priority task first. If low-priority task was already running, the scheduler will stop it and the higher-priority task will begin to run. The low-priority task will only get to run after the higher-priority task has finished with its current work. The external events are typically software and hardware interrupts (for example triggered by sensors).

practicable for a large range of realistic situations encountered in the design and analysis of real-time systems. A collection of quantitative methods and algorithms has been made available that allows to specify, analyze, and predict the timing behavior of real-time software systems. Nevertheless real-time systems vary in their requirements and real-time scheduling doesn't have to be so uniform. There are many applications where properties of hardware devices and software configurations make preemption impossible or expensive. In some applications, preemption also creates a number of problems for developers. Programming in such an environment necessarily creates excess complexity when the application is not well suited to being coded as a set of tasks that can preempt each other. Sometimes this added complexity results in system failures. It almost always also lengthens development and debug cycles.

In a nonpreemptive scheduling policy task instances execute without being interrupted once they have started their execution. Nonpreemptive policy is used for example for the message communication on the Feldbus CAN. The bus can not be used preemptively. Non-preemptive real-time scheduling requires less overhead, as requested by low performance micro-controllers, because both synchronization primitives and deadline sorting at each task release are not necessary, and because task switching is less frequent.

Disadvantages of non-preemptive scheduling policy are lower responsiveness and utilization of the CPU resources. Currently it can not be decided in general whether for a concrete application a preemptive or a nonpreemptive policy is more appropriate, although different authors provide guidelines for particular cases [54, 47].

The advantages of a non-preemptive scheduling are:

1. The scheduler has to intervene less often, which reduces it's overhead.

2. Tasks may need resources that can only be used in an exclusive manner. It implies that while an instance is using such a resource it may not be preempted. This condition is automatically satisfied under nonpreemptive scheduling policies. Preemptive policies on the other hand must provide synchronization primitives with added complexity.

3. The implementation of a nonpreemptive policy is simpler because during the execution of an instance the scheduler does not need to intervene.

These advantages are even more important in embedded systems. Reduced overhead reduces the power consumption which is critical in mobile robotics (and in many other fields for that matter). The reduced implementation complexity deriving from the other two advantages may be considered even more important during the development and testing phases. Whoever developed applications for embedded systems knows how precious is dealing with a simpler system. Hardware, software and operating systems issues are intricately connected making it harder to isolate problems and develop clean code. Moreover code is shorter and easier to handle without synchronization primitives and there is no need to implement careful design in order to avoid well known problems of fixed priority preemptive policies like deadlock and priority inversion [47, 31].

In the application described in this chapter, in fact, non-preemptive management of the sensors leads to a cheaper utilization of computing resources and enables to perform a more accurate response analysis. The implementation is easier, the stack memory requirements are reduced and there is no synchronization overhead.

4.2.2 Terminology and definitions

Many real situations can be modeled by a group of tasks that make repeated request for processor time. In real-time scheduling the determination if the task set is schedulable is essential. A task set is said to be schedulable if any legal set of requests has a corresponding schedule in which no deadlines are missed.

A periodic task is denoted by τ_i. A periodic task set is represented by the collection of periodic tasks, $\tau = \tau_i$. Each τ_i is associated with (p_i, D_i, C_i); period, relative deadline, and the worst case computation time respectively.

A concrete task has a specified release time, or the time of the first invocation. The difficulty of scheduling tasks can be affected by the release time. A periodic task set is said to be schedulable if and only if all concrete task sets that can be generated from the periodic task set are schedulable. We will consider only periodic task sets in this chapter.

In this chapter it is assumed that:

- There is only one processor.
- Scheduling overhead can be ignored.
- $C_i \leq D_i \leq p_i$
- Tasks are sorted in non-decreasing order by period. That is, for any pair of tasks τ_i and τ_j, if $j < i$ then $p_j \leq p_i$.
- Tasks become ready when they arrive, i.e. there is no inserted idle-time.

Remark 1. *We assume that time is discrete and it is indexed by natural numbers because it is measured in clock ticks.*

Remark 2. *A fundamental parameter in real-time scheduling of n tasks is the utilization factor U: $U = \sum_{i=1}^{n} \frac{C_i}{p_i}$.*

There are several authors who have presented some results on non-preemptive scheduling [36]. The main difficulty with non-preemptive scheduling is that it is, in general, a NP-complete problem [23] for every processor load [24]. In certain constrained cases the NP-completeness can be broken, as shown by Jeffay et al. in [28] and Georges in [24] for EDF (Earliest Deadline First) scheduling. In particular, Jeffay et al. show that necessary and sufficient scheduling conditions for a set of n non decreasing periodic tasks, i.e. $p_1 \leq p_2 \leq \cdots \leq p_n$, are the following:

$$\sum_{i=1}^{n} \frac{C_i}{p_i} \leq 1 \tag{4.1}$$

and

$$t \geq C_i + \sum_{j=1}^{i-1} \lfloor \frac{t-1}{p_j} \rfloor C_j \;\; \forall\, 1 < i \leq n,\; \forall\, t, p_1 < t < p_i. \tag{4.2}$$

In other words, informally, the EDF scheduling of the set of n periodic tasks is schedulable if, according to the first condition, there is enough computational capacity to execute all tasks while, according to the second condition, the total computational demand in a temporal interval t is lower than the length of the interval itself.

4.2.3 Non-preemptive design methodology

This section describes a design methodology which is based on the assignment of the computation times C_i of the non-preemptive tasks according to the physical requirements and subjected to suitable bounds. In other words we seek the values of the bounds B_i so that if $C_i < B_i$, $\forall i = 1...n$, the set of periodic tasks is schedulable.

Starting from the Jeffay conditions (4.1) and (4.2), we now derive the bounds for the periodic tasks executions which bring the task set to be schedulable using the EDF non-preemptive policy.

To this purpose we can easily prove the following Proposition which states a sufficient condition for a set of tasks to be schedulable.

Proposition 1. *If the computation times of a set of n non-preemptive periodic tasks are bounded by B_i:*

$$B_i = p_1 \left(1 - \sum_{j=1}^{i-1} \frac{C_j}{p_j} \right)$$

for $i = 1 \dots n$, then the set of tasks is schedulable using non-preemptive EDF.

Proof. The bounds are a direct consequence of the condition reported in eq. (4.2), which can be put in the following form.

$$C_i \leq t - \sum_{j=1}^{i-1} \lfloor \frac{t-1}{p_j} \rfloor C_j, \ \ \forall\, i = 2 \dots n, \ \forall \ \textit{integer}\ t : p_1 < t < p_i. \tag{4.3}$$

If there is only one periodic task, then we can set $B_1 = p_1$. On the other hand, if there are two tasks ($n = 2$), then the condition reported in eq. (4.3) becomes $C_2 \leq t - \lfloor \frac{t-1}{p_1} \rfloor C_1, \forall t : p_1 < t < p_2$ or, in other words, $B_2 = \min_{p_1<t<p_2}(t - \lfloor \frac{t-1}{p_1} \rfloor C_1)$. Since the possible values for t are: $p_1+1, \dots, p_2-1$, we have $B_2 = p_1 + 1 - C_1$. Consider now the situation with i tasks. As before, we have

$$U_i = \min_{p_1<t<p_i} (t - \sum_{j=1}^{i-1} \lfloor \frac{t-1}{p_j} \rfloor C_j) \tag{4.4}$$

By considering the quantity

$$\overline{U_i} = \min_{p_1<t<p_i} \left(t - \sum_{j=1}^{i-1} \frac{t-1}{p_j} C_j \right) \tag{4.5}$$

which is the same as eq. (4.4) without the floor operator, then $\overline{U_i} \leq U_i$. This means that if $C_i \leq \overline{U_i}$ the condition expressed in eq.(4.2) surely holds. Now we can easily find out that

$$\begin{aligned}\overline{U_i} &= \min_{p_1<t<p_i}\left(t-\sum_{j=1}^{i-1}\frac{t-1}{p_j}C_j\right)= \\ &= \min_{p_1<t<p_i}\left(t\left(1-\sum_{j=1}^{i-1}\frac{C_j}{p_j}\right)+\sum_{j=1}^{i-1}\frac{C_j}{p_j}\right)= \\ &= (p_1+1)\left(1-\sum_{j=1}^{i-1}\frac{C_j}{p_j}\right)+\sum_{j=1}^{i-1}\frac{C_j}{p_j}= \\ &= p_1\left(1-\sum_{j=1}^{i-1}\frac{C_j}{p_j}\right)+1 \end{aligned} \tag{4.6}$$

On the other hand, the first Jeffay condition, expressed in eq.(4.1), can be stated as

$$C_i \leq V_i = \left(1-\sum_{j=1}^{i-1}\frac{C_j}{p_j}\right)p_i$$

In conclusion, we can state that if $C_i \leq B_i$, where $B_i = \min(\overline{U_i}, V_i) = \overline{U_i}$ as p_i are in non-decreasing order and so $\overline{U_i} \leq V_i$, both the Jeffay conditions are satisfied and the task set is EDF schedulable. It is worth noting that the opposite is not true, namely the condition is only sufficient.

This derivation completes the proof.

□

From the Proposition a design methodology can immediately be derived for non-preemptive real-time scheduling, consisting in finding the bounds of each task which guarantee scheduling of the task set, and setting the duration of the tasks within the bounds and according to the physical constraints. In other words, the physical system to be controlled through the real time kernel must have time constants less than the computed bounds. Otherwise, the architecture of the real time solution must be formulated in a different way.

To show how the above conditions can be used in practice, we have worked out the following example.

Example Let us consider three tasks, with $p_1 = 4000$, $p_2 = 6000$ and $p_3 = 7000$. Then, $B_1 = 4000$, and assume that $C_1 = 1500$. Then, the bounds which guarantee the tasks to be schedulable, are: $B_2 = 4000\left(1-\frac{C_1}{p_1}\right) = 2500$. Assume then that $C_2 = 1500$. In same way, $B_3 = 4000\left(1-\frac{C_1}{p_1}-\frac{C_2}{p_2}\right) = 1500$. Assume then that $C_3 = 1500$.

This scheduling is outlined in Fig. 4.3.

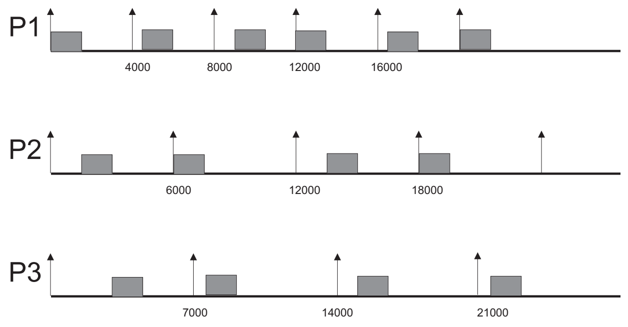


Figure 4.3: Scheduling example

Remark 3. *The algorithm has a complexity of $O(n^2)$ divisions, where n is the number of tasks.*

In fact, for $i = 2$ we have to compute one division, for $i = 3$ we have two divisions, and for the generic $i = n$, we have one product and $n - 1$ divisions. It is worth noting that complexity is not a critical problem, because the scheduling is statically designed.

4.3 Yartek architecture

Yartek has been designed according to the following characteristics:

- running on the Freescale MCF5282 Coldfire micro-controller [21];
- non-preemptive EDF scheduling of periodic real-time threads;
- background scheduling of non real-time threads;
- sensor data acquisition with a polling mechanism;
- deferred interrupt mechanism;
- contiguous stack and data memory management using first-fit policy;
- RAM-disk management;
- system call primitives for thread, memory and file management;
- general purpose I/O management;
- communication with the external world via serial port.

Yartek allows the creation and running of threads for fast context switch and it is based on a contiguous memory; moreover it offers a dynamic memory management using a first-fit criterion. The threads can be real-time periodic scheduled with non-preemptive EDF [28], or non real-time. In order to improve the usability of the system, a RAM-disk is included: it is actually an array defined in the main memory and managed using pointers, therefore its operation is very fast. The RAM-disk offers a file system structure for storing temporary data and executable code to enrich the amount of real-time applications which the kernel can run.

YARTEK has added flexibility of switching/implementing, in addition to non-preemption, also non real-time preemptive tasks. For example tasks used for communication with external devices, for example through a serial port which is managed using interrupts, can be served by non real-time tasks.

4.3.1 Task scheduling

Task scheduling is one of the main activities of the operating system. All the scheduling operations are performed on the basis of a real-time clock, called RT-Clock, which is generated by an internal timer. Each task is represented using a data structure called Thread Control Block (TCB), which is reported in Appendix A. TCB contains the name, type and priority of the process, its allocated memory and fields used to store the processor's state during task execution. For real-time processes the TCB also contains *Start*, *Dline*, and *Period* fields to store the time when the process starts, its deadline ant its period. Scheduling is managed with a linked list of TCBs with 3 priority levels, as shown in Fig. 4.4. There is one more queue used for storing free TCBs.

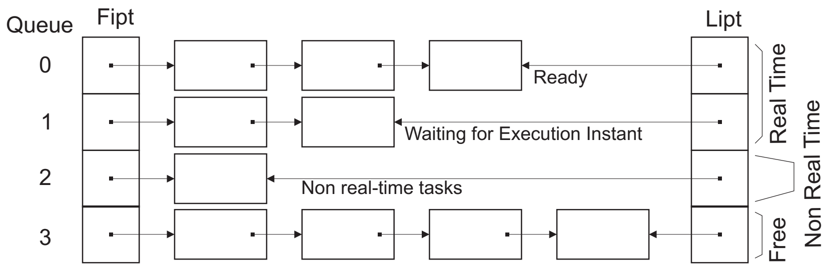


Figure 4.4: Yartek queues used for scheduling.

The periodic real-time threads are managed using non-preemptive EDF scheduling. The TCB queue 0 and 1 are used as follows: queue 0 contains the TCBs of active threads, i.e. ready to be scheduled, and is ordered by non-decreasing deadline; hence these TCBs are executed according to the EDF policy. In the queue 1 are inserted, ordered by start time, the periodic threads awaiting to be activated. As time lasts, some tasks can become active and the corresponding TCBs shall be removed from queue 1 and inserted into queue 0. This operation is performed by the ServiceTaskQueue routine, which analyzes the TCBs on queue 1 to seek start times less than or equal to RTClock, i.e. threads to be activated. Moreover, in queue 2 are stored the TCBs of non real-time threads. The queue 2 is managed using a FIFO policy.

The entry point of the kernel is an infinite loop where the interrupt table and the task queue are examined, as described in the following pseudo-code:

```
MainLoop() {
    while(true){
        if(InterruptTable is not empty)
            ServiceInterruptTable();
        else
            ServicetaskQueue();
    }
}
```

Interrupts are served using a deferred mechanism: each interrupt raises a flag on an interrupt table, reported in the Appendix, and the ServiceInterruptTable routine checks the interrupt table to verify if a pending interrupt flag is set. In this case it activates the suitable non real-time thread for serving that interrupt.

The movement of a TCB from a queue to another at a given priority level is performed with a procedure which inserts the task in the task queue.

4.3.2 Process States

When a real-time thread is created (Fig. 4.5), it is in *Ready* state when $Start\ time \leq RTClock$ (TCB inside queue 0), it is in *Waiting to be activated* state when $Start\ time > RTClock$ (TCB inside queue 1). The first *Ready* thread shall then

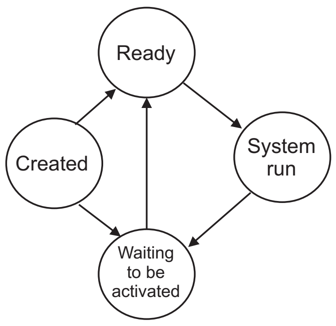


Figure 4.5: State diagram of a real-time thread in Yartek.

be selected for execution and will go into *System run* state. When the execution stops, the process will become *Waiting to be activated* as it is periodic, and the scheduler updates its start time and its deadline adding them the thread period.

The states of a non real-time thread are similar, and are reported in Fig. 4.6. The main difference is that a non real-time thread is preemptive, and it can be interrupted by real-time threads.

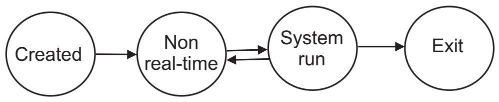


Figure 4.6: State diagram of a non real-time thread in Yartek.

To create a process, a TCB is taken with GetTCB() function and filled with process information. The QueueTCB function then inserts the TCB in the requested queue. For queue 0 the TCB is inserted in ascending order according to deadlines, while for the queue 1 the TCB is inserted in ascending order according to activation time. For the other two queues it is simply enqueued at the end.

4.3.3 Memory management

An amount of stack and data memory, containing thread-related information such as a local file table and information needed for thread management and user variables, is assigned to each process; furthermore dynamic memory is also available when requested by system calls. Stack, data and heap memory are organized in a sequence of blocks managed with first-fit policy.

4.3.4 System calls

A number of system calls has been implemented using the exception mechanism based on the trap instruction. The system calls are divided into file system management (open, read, write, close, unlink, rewind, chname), process management (exec, kill, exit), heap management (alloc, free) and thread management functions (suspend, resume).

4.3.5 Timer and interrupts

The micro-controller MCF5282 [21] has 4 programmable timers: one is used as the system's time reference, and it is used as RTClock, and the other timers are used for the measurement of time intervals. The timer is composed of a 16-bit register and a frequency divider. The first timer is used as the system's time reference, and it is used as RTClock. Since four interrupts are used for the timers, there are three interrupt levels for application code. The routines activated by interrupts set a single flag in the interrupt table. Later, the scheduler activates a process to actually manage the request, i.e. in deferred mode. The pseudo-code of an interrupt service routine is illustrated as follows:

```
Interrupt_handler:
        set the flag in the operating system table;
        return from interrupt;
```

4.4 Performance evaluation

Generally speaking, as noted in [60], measuring real-time operating system performance and comparing a real time system to other real-time operating systems is a difficult task. The first problem is the fact that different systems can have different functionalities, and the second concerns the method used to perform the actual measurements. Many features are worth to be measured: for example, Sacha [60] measures the speed of inter-task communication, speed of context switch and speed of interrupt handling, while Garcia-Martinez et al. [22] reported measurements of responses to external events, inter-task synchronization and resource sharing, and inter-task data transferring. Finally, Baynes et al. [3] considered what happens when a real-time operating system is pushed beyond its limits; they also report real-time operating system power consumption measurements.

This Section reports some measures used to describe the performance of Yartek, namely context switch time, jitter time, interrupt latency time, kernel stability and kernel overhead. Yartek has been implemented on the AVNET board [2] part number `ADS-MOT-5282-EVL`, based on the Freescale MCF5282 ColdFire Processor running at 29.5 MHz. It is equipped with BDM/JTAG interface and has 16 MB SDRAM and 8 MB Flash. The communications are based on general purpouse I/O (GPIO) on AvBus expansion connector, two RS-232 serial ports, 10/100 Ethernet. The performances obtained with Yartek have been compared to the performance of RTAI operating system, ported to this board.

4.4.1 RTAI

The RTAI project [44, 58, 62] began at the Dipartimento di Ingegneria Aerospaziale del Politecnico di Milano (DIAPM) as a plug-in which permits Linux to fulfil some real-time constraints. RTAI allows real-time tasks to run concurrently with Linux processes and offers some services related to hardware management layer dealing with peripherals, scheduling and communication means among tasks and Linux processes. In particular, RTAI port for Coldfire micro-controller is a uCLinux [66] kernel extension that allows to preempt the kernel at any time to perform real-time operations. Unlike the implementations of RTAI for x86, PPC and MIPS, the Coldfire version is not based on the "deferred interrupt mechanism" but uses the capability of the MC68000 architecture to priorities interrupts in hardware by the interrupt controller. In the current implementation, Linux interrupts are assigned lower priority interrupt levels than RTAI interrupts. However, RTAI presents drawbacks which restrict the fields of integration:

1. the preemptive scheduler works with static priorities and there is no built-in non-preemptive EDF scheduler,

2. both aperiodic and periodic tasks can be used but the priority of a periodic task bears no relation to its period,
3. deadlines are not used,
4. it is a hybrid system with high footprint.

4.4.2 Performance measures and comparisons

Experimental results regarding kernel performances are based on the following parameters:

context switch time: it is the time spent during the execution of the context switch routine of the scheduler.

jitter time: it is the time delay between the activation time of a periodic real-time process and the actual time in which the process starts.

deferred interrupt latency: it is the time delay between an interrupt event and the execution of the first instruction of the deferred task scheduling its service routine.

kernel stability: it establishes the robustness of the kernel.

kernel overhead: it represents the time the kernel spends for its functioning.

Test for context switch time

The test program used for measuring the context switch time is shown in Fig. 4.7.

```
void test_context_switch( ) {
  int i;
  CreateRTtask( Thread, thread_code );
  ActivateTask( thread );
  t0 = StartTimer( );
  for ( i = 0; i < LOOPS; ++i ) {
    Resume( thread );
  }
  t1 = StopTimer( );
}

void thread_code( ) {
  while (1) {
    Suspend();
  }
```

Figure 4.7: Pseudo-code of the context-switch test.

The test program creates one thread and measures the time which lasts from the thread execution to subsequent suspension repeating the process *LOOPS* times. More precisely, in Yartek the only thing the thread does is to suspend itself, that is to put its TCB on the queue 1 while the loop calls the resume *LOOPS* times. *Resume*() moves the TCB on the queue 0 so it will be immediately scheduled. It is worth noting that *Resume*() is a blocking primitives, i.e. it exits only when the TCB is put indeed in queue 0.

In RTAI, *rt_task_suspend*() suspends the execution of the task given as the argument, *rt_task_resume*() resumes execution of the task indicated as argument previously suspended by *rt_task_suspend*().

In Yartek the average context switch time is 130 μs, while in RTAI is 124 μs.

Test for jitter time

It is worth recalling the detailed operation of Yartek to manage periodic real-time tasks. The TCB queue 1 contains the periodic threads, ordered by activation time, whose activation time is in the future. The starting time of the periodic tasks that lie on queue 1 are tested to detect the scheduling condition, i.e. the starting time in TCB is greater than current time. In this case, the TCB is moved to the queue 0 which is sorted by deadline.

The test program used for measuring jitter time is shown in Fig. 4.8.

```
void test_jitter_time( ) {
  CreateRTtask( thread, thread_code );
  ActivateTask( thread );
}

void thread_code( ) {
  int loops=LOOPS;
  while (loops--) {
    t1[loops] = RTClock();
    t2[loops] = CurrentTCB->StartTime;
  }
}
```

Figure 4.8: Pseudo-code of the jitter time test.

In Yartek, TCB is the data structure containing the data related to threads, and CurrentTCB is the pointer to the current thread. The system call RT-Clock() returns the value of the real-time clock used by Yartek for scheduling real-time tasks. However, in Yartek, real-time threads are activated by the ServiceTaskQueue routine and therefore the exact starting time does not necessarily correspond to the start time written in the TCB. The delay can vary depending on the state of the routine. As a consequence, the uncertainty of activation time can be quite high. In fact, testing results show that Yartek has an average jitter time of 750 μs, while RTAI has an average jitter time of 182 μs.

Test for deferred interrupt latency

Yartek serves the interrupt using a deferred mechanism, i.e. only a flag is raised immediately. A non real-time thread is activate only when the scheduler processes the Interrupt Table.

As RTAI does not implement on the Coldfire porting any interrupt service, the time needed to schedule a task has been estimated, thus implementing a deferred interrupt service.

```
void test_deferred_interrupt_latency( ) {
  CreateRTtask( thread1, thread_code1 );
  ActivateTask( thread1 );
}

void thread_code1( ) {
  CreateRTtask( thread2, thread_code2 );
  t1 = StartTimer( );
  ActivateTask( thread2 );
}

void thread_code2( ) {
  t2 = StopTimer( );
}
```

Figure 4.9: Pseudo-code of the deferred interrupt latency test.

So, the test program used for measuring deferred interrupt latency is schematically presented in Fig. 4.9. There are no periodic real-time threads in execution during the tests.

By using Yartek an average deferred interrupt latency of 780 μs is obtained, while using RTAI 17 ms is obtained.

Kernel stability and overhead

Although no specific stability test has been performed, Yartek has shown a good robustness since it run for several hours both for performing applications and for evaluating the performance tests.

The time the kernel spends for itself and not for the application is mainly divided in scheduling time, context switch time and memory management time. The essential code of scheduling is reported in the Appendix.

In summary, the worst case complexity for interrupt management is about 110 assembler instructions to deferred schedule a non-real time thread for serving one interrupt. The management of the real-time task queues, under the hypothesis of one real-time TCB to be activated, requires about 150 assembler instructions. Regarding memory management, the alloc system call requires about 300 assembler instructions for allocating one block of contiguous memory using first fit and the free system call about 270 assembler instructions. These overheads expressed in time depend on the actual CPU frequency and for this reason has been left in number of instructions.

As previously computed, the overhead for context switch is 130 μs.

4.4.3 Discussion

The first important difference between RTAI and Yartek is the operating system footprint. The footprint of an operating system concerns the usage of RAM and flash memory resources. As noted in Section 4.4.1, the RTAI plug-in works with a Linux kernel. In the tests, uCLinux has been used which has a minimum kernel size of 829 kBytes. The RTAI modules have a size of 97 kBytes, so the whole image is of about 900 kBytes. Instead, the footprint of Yartek is about 120 kBytes. This big difference in size is due to the fact that Linux plus RTAI brings some of the powerful tools and features of Unix. However, these tools are not necessary for an embedded system. The second difference between RTAI and Yartek is the non-preemptive scheduling. It is worth remarking the adequacy of non-preemptive scheduling in real-time embedded systems on low power micro-controllers. RTAI does not offer non-preemptive scheduling. Of course, it could be introduced by coding a new scheduler and integrating it in RTAI, but it was instead decided to modify the previous available kernel [51] for footprint reasons.

In conclusion, it was decided to use Yartek for developing embedded solutions. The performance evaluation tests show that the time performances of Yartek are similar to RTAI for the context switch time and that the time for task creation is much lower for Yartek than RTAI. However, the jitter time for Yartek is much worse than RTAI, due to the Yartek architecture. It has to be noted, however, that the performance of an embedded systems for non visual sensors is not critical with respect to the jitter time, due to the time constants of such sensors. Yartek allows to manage task allocation that would be much more complex on a non-real time or a sequential system providing periodic threads scheduled with a non-preemptive EDF policy; the schedulability can be tested using the simple methodology presented in Section 4.2.

4.5 Application: embedded map building system for mobile robots

In mobile robotics several tasks require the strict satisfaction of time constraints, so real-time systems are needed. The sensors generally used for mobile robot navigation, such as inertial navigation systems, sonar sensor array, GPS, laser beacons, should be processed considering real-time constraints. As map building is a fundamental task in mobile robotics, an application of this type was considered. Its design is described and its implementation using Yartek is outlined in the following.

4.5.1 Previous work in map building for robot navigation

The goal of robotic map building is that of acquiring a spatial model of a robot's environment. The problems inherent in mapping the environment were described in the introduction of this thesis § 1.1.1.

In the experiments described in the thesis metric maps in the form of spatial occupancy grids are used. These maps attempt to represent a world without trying to identify any individual object.

The occupancy grid method [49, 17, 18] provides a probabilistic framework for target detection, that is, determining whether a region of space is occupied or not.

Elfes [17, 18] reformulated the problem as a probabilistic Bayesian updating problem using gaussian noise with a very large variance to account for the gross errors entailed by multiple reflections. He also addressed the problem of geometric uncertainty associated with sensor beam width by considering target detection under all possible configurations of the environment. In practice, given the overwhelming combinatorics of keeping track of data associations for each reading, independence and other simplifying assumptions are made to reduce the computational complexity of Bayesian update. That is, each cell of space is treated as an independent target in the presence of the geometric uncertainties induced by the beam width. This leads to unrealistic estimates for target map updates, e.g., all the cells at the leading edge of the beam have their probabilities raised, when in fact usually only one cell is responsible for the echo.

Borenstein and Koren [10, 9] introduced the Vector Field Histogramm (VFH) method. They use a spatial histogram of sonar points, along the axis of the sonar beam, to identify areas that are likely to contain obstacles. The histogram is updated rapidly as new returns come in, and older ones are abandoned. The VFH method has the advantage that it can deal with dynamic and noisy environments. Since it is based on a continuous update of the map, this method is particularly suitable for mobile robots. The updating of the map can in fact be performed during the robot movement.

4.5.2 Implementation

Yartek has been installed on an embedded system composed by the Coldfire micro-controller board connected to an array of 6 sonar sensors placed in front of a mobile robot, and a PC-104 board on top of the robot, as shown in Fig. 4.10.

The embedded system controls the ultrasonic sensors, calculates the robot position from the odometry readings and updates the internal map. The last N_c sensor readings are stored internally in a circular list and used for updating the map with N_c set according to the odometry error model. The map is a histogram grid [10, 9] which is quite simple for a computational point of view. It is suitable

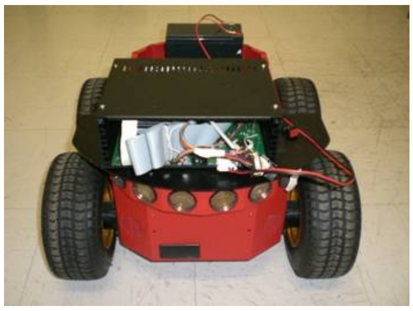


Figure 4.10: Mobile robot, type PIONEER 3-AT, equipped with the embedded system.

for rapid in-motion sampling of onboard range sensors and modeling of inaccurate and noisy range-sensor data, such as that produced by ultrasonic sensors. The system provides the internal map through a serial port and receives commands such as clear the map, compute the map and send the map.

The application is designed as follows:

Sensor acquisition tasks. There are 6 real-time periodic tasks that perform the acquisition of the data from the sonar sensor and 2 real-time periodic tasks that read the odometers.

Map updating tasks. There is a real-time periodic task that updates the map according to the acquired sensorial data provided by the sensor acquisition task and Antisensor real-time periodic task with larger period for filtering erroneous readings.

Map requests. These are aperiodic tasks scheduled upon requests. There is an aperiodic task that allows to obtain the complete map from the embedded system using serial line connection. There is also a task for clearing the map and resetting the robot's position.

There is one real-time task for each ultrasonic sensor that controls the firing of the sensor and waits for the reflected ultrasonic burst. This is compatible with the sonar array operation since only one sensor fires the ultrasonic burst at a time in order to avoid crosstalk. Crosstalk is a common problem occurring with arrays of multiple ultrasonic sensors when echo emitted from one sensor is received by another sensor leading to erroneous reading. In order to avoid crosstalk only one sensor emits the ultrasonic burst and listens for the echo. Only after the echo has been received or a maximum allowed time has elapsed in the case there is no obstacle in front of the sensor, the process can be repeated for the next sensor, and so on.

The range distance of the sensor is 3 m which is equivalent to a time of flight (TOF) of 17 ms. It is assumed that the set of sensors is read every 500 ms. This fact poses a physical limit on the maximum robot speed to approximately 0.5 m/s (if the robot is not to bump into obstacles and considering that minimum detectable distance is set to 20 cm). This is more than enough for the application. Finally, note that in this application the interrupts are generated only by the serial port.

The design methodology described in Section 4.2.3 has driven the development of this application. We call:
getSonar1, getSonar2, ..., getSonar6 as the periodic tasks which read the sensors with period $p_1 = p_2 = \cdots = p_6 = 500$ ms;
updateMap a periodic task with period $p_7 = 500$ ms;
getOdo1, getOdo2 periodic tasks with period $p8 = p9 = 1200$ ms;
antiSensor a periodic task with period $p_{10} = 2000$ ms.
We choose worst case task duration $C_1 = 20$ ms ($B_1 = 500$ ms), then $B_2 = 480$ ms.
Choosing $C_2 = 20$ ms, then $B_3 = 460$ ms.
Choosing $C_3 = 20$ ms, then $B_4 = 440$ ms.
Choosing $C_4 = 20$ ms, then $B_5 = 420$ ms.
Choosing $C_5 = 20$ ms, then $B_6 = 400$ ms.
Choosing $C_6 = 20$ ms, then $B_7 = 380$ ms.
Choosing $C_7 = 100$ ms, then $B_8 = 280$ ms.
Choosing $C_8 = 20$ ms, then $B_9 = 500(1 - 220/500 - 20/1200) = 271$ ms.
Choosing $C_9 = 20$ ms, then $B_{10} = 500(1 - 220/500 - 40/1200) = 263$ ms.
Finally we choose $C_{10} = 20$ ms.
As the condition holds, then the set of task can be scheduled using non-preemptive scheduling. In Fig. 4.11 the corresponding time diagram of the non-preemptive scheduling designed for the map buiding application is reported. In black are shown the part of times which can be dedicated to the aperiodic tasks.

The pseudo-code for sonar real-time tasks is:

```
robot.getSonarN()
{
    fire ultrasonic sensor N (set logical 1 on digital pin)
    while time < maxtime and echo not detected
    {
        listen for echo (check the I/O pin of sensor N for logical level 1)
        time++
    }
    calculate distance from time of flight
}
```

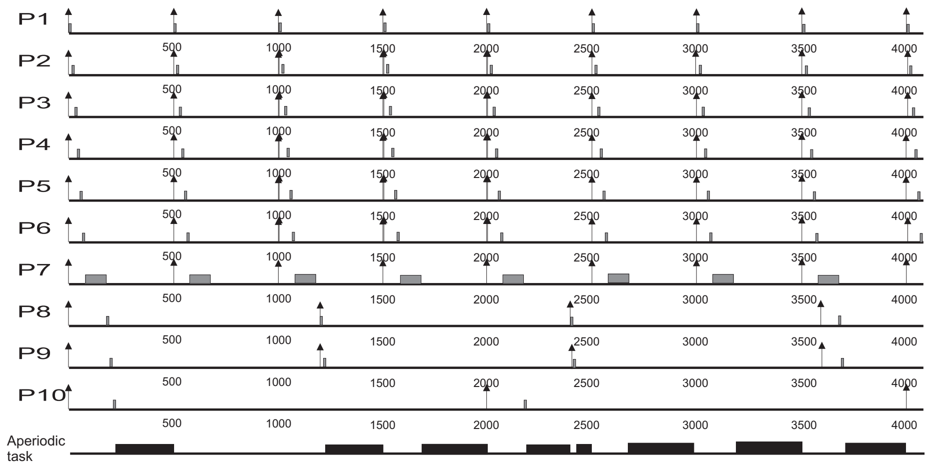


Figure 4.11: Time diagram of real time scheduling for the map building application.

Two other real-time periodic tasks read the odometers: the odometer increments the counter register for each tick of the wheel encoder. The task reads the register and compares it with the previous reading to find a traversed distance. The wrapping of the counter when passing from maximum value to 0 is easily handled since max wheel speed is such that relative distance to previous reading is limited.

```
robot.getOdoN
{
 read OdoN counter;
 diff=counter-previous;
 if diff<0 diff=maxcounter-previous+counter;
 calculate traversed distance from counter value;
 previous=counter;
}
```

The real-time periodic process that updates the map reads the sensor readings stored in the internal circular list, calculates the robot's position and updates the cells in the histogram grid corresponding to sensor readings. The most recent readings increment the histogram grid cells while the oldest ones decrement the cells, i.e. only the last Nc readings compare in the map. In this way the portion of the map with revealed obstacles moves with the robot.

```
robot.updateMap();
{
    getLastOdoReadings();        // the most recent odometry
                                 //  readings in circular list
    calculateRobotPosition();
    getLastSonarReadings();     // the most recent sonar readings
    calculateSonarReadings();   // polar to cartesian based
                                //  on relative sensor position
    updateMapPositive();        // increments histogram map cells

    getFirstOdoReadings();      // the oldest odometry readings
    calculateRobotPosition();
    getFirstSonarReadings();    // the oldest sonar readings
    calculateSonarReadings();
    updateMapNegative();        // decrements histogram map cells
}
```

As opposed to the updateMap task a second real-time periodic task called Antisensor decrements all cells in a specified area around a robot (an area in front of the robot with range 1,5 m in the implementation). This mechanism is used in order to eliminate moving obstacles and to correct erroneous sonar readings. The Antisensor task is scheduled with much larger period than the map updating task allowing for the cells in which real obstacles are present to reach high values before they get decremented.

The map transfer task is a real-time aperiodic task scheduled when a request for a map arrives. The array containing the map is then compressed with a simple (value, count) scheme and transmitted to a serial port running at 9600 bps. The number of bytes to be transmitted does not grow with the array size due to the updating mechanism discussed earlier where only the last readings compare in the map.

```
robot.sendMap()
{
    compressMatrixData();       // the map stored in memory array
    transferMatrixData();       // is transmitted to serial port
}
```

4.5.3 Experimental results

The bitmap estimated with the *updateMap()* method is referred to absolute coordinates and requires absolute localization results obtained with odometry. In other words, the map is an array of bytes where each element represents a cell of the grid by which the environment is divided. In our case, each cell represents a square of 50 mm long sides.

In Fig. 4.12 some examples of maps obtained with the described embedded system are reported: on the left the actual map of the environment and on the right the perceived map obtained from the embedded device.

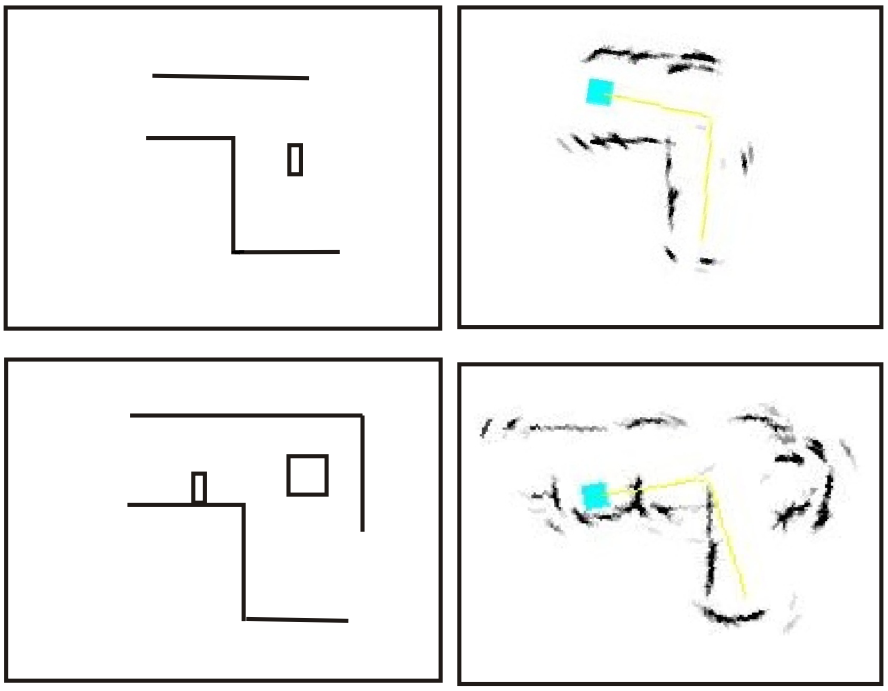


Figure 4.12: Map building results. On the left the real maps, on the right the maps estimated with the embedded system.

4.6 Concluding remarks

Theoretical results have been obtained in the field of non-preemptive scheduling of periodic real time processes. The general non-preemptive scheduling has been seen to be NP-complete problem, so sub-optimal scheduling policies must be used in realistic hard real-time systems. One of such policies has been proposed in this thesis. Bounds on maximum task execution time are introduced that bring the task set to be feasible, i.e. sufficient conditions for the schedulability of a task set in non-idling context are proposed and a design methodology is derived.

Further work could extend the bounds in the case of fixed-priority scheduling policy for non preemptive tasks with deadline smaller than period.

Using the proposed design methodology a robot mapping application has been developed. The embedded system computes an environmental map using the ultrasonic sensors. The created grid based map is compressed and transferred to the robot upon request without imposing any computational cost on the robot's main processor.

A real-time kernel called Yartek has been described. The development of Yartek was motivated by the impossibility to find an open source real-time kernel for the Coldfire micro-controller using non-preemptive scheduling. The main features include the usage of a non preemptive EDF scheduler and a small kernel which can easily be ported to other micro-controller architectures. A deferred interrupt mechanism is also implemented; using this mechanism, interrupts are served by non real-time, aperiodic tasks, which are scheduled in background. Other features of the kernel include a thread management mechanism, dynamic memory management using first-fit, availability of a serial port driver which allows the connection of an external terminal for data exchange and system monitoring purposes, and availability of a RAM-disk which provides a convenient data structure for temporary storage of information and for easily extending the operating system features. Applications must be compiled inside the operating system using cross-development tools.

Yartek has been developed for the Coldfire micro-controller family; if a porting to different architecture is to be performed, there are some short pieces of assembler code, mainly for timers management, which must be rewritten. It should be pointed out that Yartek could be simply modified for working both with preemptive and non-preemptive scheduling. In the preemptive case, another table is required to contain the TCB ID, the next activation time and the task period. Before executing a TCB code, a timer is set in order to execute the scheduling when a periodic task arrives. The interrupt routine first updates this table, adding the period to the next activation time of the current task, it sorts the table by the next activation time in the ascending order, and then the scheduler is called.

Appendix B reports some steps that should be done to adapt Yartek to another application.

The system is highly reconfigurable. Almost all the kernel has been written in C language, and the source code is downloadable [69]. The executable image takes less than 120 kBytes.

There is, however, room for further development: for the moment Yartek operating system runs only the described target board, it has a limited set of features, lacks a consistent documentation and some routines may be further optimized. In the future the development of Yartek could continue on its own, adding new features and porting to other boards, or, it could be merged in the development tree of a similar open-source RTOS.

The main contributions of this chapter are the following.

Theoretical results regarding the scheduling conditions for periodic tasks in non preemptive scheduling are presented. A simple design methodology for non-preemptive EDF (Earliest Deadline First) scheduling is proposed based on bounds on the duration of non-preemptive tasks. The Yartek embedded kernel is introduced, and its performances and comparisons to a different real-time operating system are reported. Finally, a real-time application is described, in the field of non visual sensor perception in robotics.

Chapter 5

Conclusions

Robotics is inherently a multi-disciplinary field. The research carried out in this thesis reflects that.

Robotic perception was dealt with in the first part of the thesis. A sensing device was developed by focusing the ultrasonic signal in order to increase the spatial resolution of the sensor and provide a detailed contour of the environment surrounding the robot. A method for the registration of the acquired raw ultrasonic signal was presented for reliable mapping of the environment. The method overcomes inherent problems of ultrasonic sensing in case of high irregularities and missing reflections. It is suitable for map building during mobile robot exploration missions. It's main limitation is small coverage area. This area however increases during exploration as more scans are processed from different positions. Moreover, the areas in which the algorithm does not fully perform may be addressed by moving the robot towards these areas for further exploration. Further work on the prototype of a focused sonar used in the experiments should consider developing a more compact and rugged device to employ the sensor in outdoor conditions.

Localization and mapping problems were addressed in the second part of the thesis. In particular scan matching techniques are used to correct the accumulated positional error using dead reckoning sensors like odometry and inertial sensors and thus cancel out the effects of noise on localization and mapping. Genetic based optimization algorithm for aligning two partially overlapped scans called GLASM has been described and discussed. It is suitable for implementation in feature-poor environments and robust to high sensor noise, as is the case with the sonar readings used in this thesis which are much noisier than laser scanners. The algorithm does not place a high computational burden on the processor which is important for real world applications where the power consumption is a concern. GLASM scales easily on multiprocessor systems since the fitness function evaluation may be easily distributed across the available CPUs. The algorithm does not require an initial position estimate, it is fast, robust and suitable for unstructured environments.

As for the future work, one of the problems that could be addressed stems from the fact that scan matching is used in a vast range of operative conditions. It is difficult to compare the strengths and weaknesses of different techniques that have been proposed over the years. Currently we are working on presenting a uniform framework for comparison of scan matching algorithms suitable for unstructured environments and real-time implementation. Another work in progress which is related to GLASM is the integration of the technique in a SLAM framework. This would enable prolonged continuous mapping explorations of real world environments.

Preliminary results on global scan matching in large environments using GLASM pointed out that the performance could be further improved with a two stage process. The first stage would provide a rough estimate for a non-uniform probability distribution for the initial population of the genetic algorithm used in the second stage. This should be further explored experimentally. Furthermore, the experiments performed with binary, Gray coded and real value coded robot poses and different genetic parameters provided some insight on the effect of the robot pose coding combined with the genetic operators of mutation and crossover on the performance of genetic algorithms. In a further research we intend to provide sound theoretical principles for coping with some of the artifacts that different codings introduce.

Finally the GLASM algorithm could be extended in the field of computer vision and compared to existing 3D image matching algorithms.

In the third part of the thesis theoretical principles regarding real-time scheduling are considered and joined in the real application where implementation issues regarding embedded systems were tackled. Some new theoretical results are derived concerning open problems in non-preemptive scheduling of periodic tasks on a uniprocessor. In mobile robotics it is critical to evaluate the above mentioned methods and devices in real world applications on systems with limited power and computational resources. This results are then used to propose a design methodology which is used in an application on a mobile robot. The mobile robot is equipped with an embedded system running a new real-time kernel with a non-preemptive scheduler of periodic tasks. The application is described and some preliminary mapping results are presented.

Thesis contributions:

The contributions of this thesis include the presentation of new algorithms and devices, their applications and also some theoretical results. The techniques are compared with the closest rivals in the state of the art.

To summarize:

- A focused ultrasonic sensing device is developed and used in mapping applications.
- An algorithm that processes the ultrasonic readings in order to develop a reliable map of the environment is presented.
- A new genetic algorithm for scan matching called GLASM that outperforms the closest rivals is proposed.
- Schedulability conditions for non-preemptive scheduling in a hard real-time operating system are introduced and a design methodology is proposed.
- A real-time kernel for embedded systems in mobile robotics is presented.
- A practical robotic application is described and implementation details and trade-offs are explained.

Bibliography

[1] A. Agrawal, N. Ansari, and E.S.H. Hou. Evolutionary programming for fast and robust point pattern matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS1994*, page 17771782, 1994. [cited at p. 38]

[2] AVNET Home page. http://www.em.avnet.com. [cited at p. 71]

[3] K. Baynes, C. Collins, E. Filterman, B. Ganesh, P. Kohout, C. Smit, T. Zhang, and B. Jacob. The performance and energy consumption of embedded real-time operating systems. *IEEE Transactions on Computers*, 52(11):1454–1469, 2003. [cited at p. 71]

[4] P.J. Besl and N.D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, pages 239–256, 1992. [cited at p. 33, 37, 38, 42, 52]

[5] Peter Biber. The normal distributions transform: A new approach to laser scan matching. In *Proceedings of the 2003 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2003. [cited at p. 36, 37]

[6] G. Blais and M.D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Trans. Pattern Anal Machine Intelligence*, 17:820824, 1995. [cited at p. 38]

[7] I.J. Cox Blanche. An experiment in guidance and navigation of an autonomous robot vehicle. *J. IEEE Trans. Robot. Automat. (TRA91)*, pages 193–203, 1991. [cited at p. 36, 37]

[8] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996. [cited at p. 11]

[9] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transaction on Robotics and Automation*, 7(4):535–539, 1991. [cited at p. 76]

[10] J. Borenstein and Y. Koren. The vector field histogram fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991. [cited at p. 76]

[11] A. Cao and J. Borenstein. Experimental Characterisation of Polaroid Ultrasonic Sensors in Single and Phased Array Configuration. In *UGV Technology Conference, SPIE AeroSense Symposium*, 2002. [cited at p. 14, 16, 105]

[12] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10:145–155, 1992. [cited at p. 37]

[13] C. Chiaruttini, P. L. Bragato, and G. Cassiani. Seismic reflection interpretation as an image understanding problem. *J. Seismic Exploration*, 3:53–68, 1994. [cited at p. 20]

[14] C.L.Liu and J.W.Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of ACM*, 20(1), Jan 1973. [cited at p. 61]

[15] J.L. Crowley. World modeling and position estimation for a mobile robot using ultra-sonic ranging. In *IEEE International Conference on Robotics and Automation*, pages 674–681, 1989. [cited at p. 14]

[16] Gregory Dudek and Michail Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press. [cited at p. 5]

[17] A. Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In S. S. Iyengar and eds. A. Elfes, editors, *Autonomous Mobile Robots: Perception, Mapping, and Navigation*, pages 60–71. IEEE Computer Society Press, 1991. [cited at p. 76]

[18] A. Elfes. Dynamic control of robot perception using multiproperty inference grids. In *IEEE International Conference on Robotics and Automation*, pages 2561–2567, 1992. [cited at p. 76]

[19] U. La Fata, K. Lenac, E. Mumolo, and M. Nolich. Exploration of Unknown Environment with Focused Sonar Sensor and Fuzzy control. In *5th WSEAS International Conference on Fuzzy Sets and Fuzzy Systems*, 2004. [cited at p. 98]

[20] A.W. Fitzgibbon. Robust registration of 2d and 3d point sets. In *Conference on British Machine Vision*, page 411420, 2001. [cited at p. 37]

[21] Freescale. MCF5282 ColdFire Microcontroller User Manual. http://www.freescale.com/files/32bit/doc/ref_manual/MCF5282UM.pdf, Nov. 2004. [cited at p. 67, 70]

[22] A. Garcia-Martinez, J.F. Conde, and A. Vina. A comprehensive approach in performance evaluation for modern real-time operating systems. In *IEEE Proocedings of EUROMICRO-22*, pages 61–68, 1996. [cited at p. 71]

[23] M.R. Garey and D.S.Johnson. *Computer and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman Company, 1979. San Francisco. [cited at p. 64]

[24] L. Georges, P. Muehlethaler, and N. Rivierre. A few results on non-preemptive real-time scheduling. Research Report 3926, INRIA, 2000. [cited at p. 64]

[25] S. Gutmann and C. Schlegel. Amos: comparison of scan matching approaches for self-localization in indoor environments. In *11th European Workshop on Advanced Mobile Robots (EUROBOT96), Kaiserslautern, Germany*, page 6167, 1996. [cited at p. 37]

[26] D. Hahnel, D. Schulz, and W. Burgard. Map building with mobile robot in populated environment. In *IROS 2002*, 2002. [cited at p. 37]

[27] R. M. Haralick. Statistical and structural approaches to texture. In *Proc. IEEE*, volume 67, pages 787–804, 1979. [cited at p. 20]

[28] K. Jeffay, D.F. Stanat, and C.U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. In *IEEE Real-Time System Symposium*, pages 129–139, 1991. [cited at p. 64, 68]

[29] K. H. Kim and H. S. Cho. Range and contour fused environment recognition for mobile robot. In *Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems MFI2001*, pages 183–188, 2001. [cited at p. 15]

[30] K. Konolige and K. Chou. Markov localization using correlation. In *IJCAI1999*, 1999. [cited at p. 37]

[31] J. Labrosse and M. Barr. Introduction to Preemptive Multitasking. *Embedded Systems Programming*, April 2003. Accepted for future publication. [cited at p. 63]

[32] D. Lee and M. Recce. Quantitative evaluation of the exploration strategies of a mobile robot. *International Journal of Robotics Research*, 16(4):413–447, August 1997. [cited at p. 95, 97]

[33] K. Lenac, E. Mumolo, and M. Nolich. Fast genetic scan matching using corresponding point measurements in mobile robotics. In *Proc. EVOIASP'07, Lecture Notes in Computer Science*, pages 125–129. Springer, 2007. [cited at p. 59]

[34] J. J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE-RSJ International Workshop on Intelligent Robots and Systems IROS-91*, pages 1442–1447, November 1991. [cited at p. 11, 14]

[35] J. J. Leonard and H. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, 1992. Boston. [cited at p. 14]

[36] W. Li, K. Kavi, and R. Akl. A non-preemptive scheduling algorithm for soft real-time systems. *Computers and Electrical Engineering*, 33(1):12–19, January 2007. [cited at p. 64]

[37] J. H. Lim and J. Leonard. Mobile robot relocation from echolocation constraints. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1035–1041, September 2000. [cited at p. 11, 15]

[38] Kai Lingemann, Andreas Nuchter, Joachim Hertzberg, and Hartmut Surmann. High-speed laser localization for mobile robots. *Robotics and Autonomous Systems*, 51:275296, 2005. [cited at p. 37]

[39] Kai Lingemann, Hartmut Surmann, Andreas Nuchter, and Joachim Hertzberg. Indoor and outdoor localization for fast mobile robots. In *Proceedings of the 2004 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2004. [cited at p. 36]

[40] E. Lomonosov, D. Chetverikov, and A. Ekart. Pre-registration of arbitrarily oriented 3d surfaces using a genetic algorithm. *Pattern Recognition Letters, Special Issue on Evolutionary Computer Vision and Image Understanding*, 27:1201–1208, 2006. [cited at p. 38]

[41] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference (CVPR94)*, pages 935–938, 1994. [cited at p. 37]

[42] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. of Intelligent and Robotic Systems*, 18:249–275, 1997. [cited at p. 34, 35, 38, 41, 42]

[43] D. Maksarov and H. Durrant-Whyte. Mobile vehicle navigation in unknown environments: a multiple hypothesis approach. In *IEE Proc. Control Theory Applications*, volume 142, pages 385–400, July 1995. [cited at p. 11, 14]

[44] P. Mantegazza. Diapm rtai for linux: Why's, what's and how's. In *Real Time Linux Workshop*. University of Technology of Vienna, 1999. [cited at p. 71]

[45] Martinez, Gozales, Morales, Mandow, and Garcia-cerezo. Mobile robot motion estimation by 2d scan matching with genetic and iterative closest point algorithms. *Journal of Field Robotics*, 23(1):21–34, 2006. [cited at p. 38, 42, 51]

[46] J.L. Martnez. Mobile robot self-localization by matching successive laser scans via genetic algorithms. In *5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications*, pages 1–6, 2003. [cited at p. 38]

[47] Jorn Martin Migge and Alain Jean-Marie. Real Time Scheduling: Non-preemption, Critical Sections and Round Robin. Research Report 3678, INRIA, 1999. [cited at p. 63]

[48] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem, 2002. [cited at p. 5]

[49] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*, pages 116–121, 1985. [cited at p. 76]

[50] E. Mumolo, K. Lenac, and M. Nolich. Spatial map building using fast texture analysis of rotating sonar sensor data for mobile robots. *IJPRAI*, 19(1):1–20, 2005. [cited at p. 35, 36, 48, 59]

[51] Enzo Mumolo, Massimiliano Nolich, and Massimo OssNoser. A Hard Real-Time Kernel for Motorola Microcontrollers. *Journal of Computing and Information Technology*, 9(3):247–252, 2001. [cited at p. 60, 75]

[52] P.J. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *IEEE International Conference on Shape Modeling and Applications*, page 130139, 1997. [cited at p. 37]

[53] S. Noykov and C. Roumenin. Calibration and interface of a polaroid ultrasonic sensor for mobile robots. *Sensors and actuators*, 135(1):169–178, 2007. [cited at p. 105]

[54] Moonju Park. Non-preemptive Fixed Priority Scheduling of Hard Real-Time Periodic Tasks. In Springer-Verlag, editor, *ICCS 2007, Part IV*, pages 881–888, 2007. [cited at p. 63]

[55] S. Pfister, K. Kriechbaum, S. Roumeliotis, and J. Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Proceedings of the 2002 IEEE Int. Conference on Robotics and Automation (ICRA)*, page 16671674, 2002. [cited at p. 35, 38, 39, 41, 52]

[56] T. Risse. Hough transform for line recognition. *Computer Vision and Image Processing*, 46:327–345, 1989. [cited at p. 19]

[57] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. In *IEEE Trans. System Man Cybernetics*, volume 6, pages 420–433, 1976. [cited at p. 21]

[58] RTAI Home Page. http://www.rtai.org. [cited at p. 71]

[59] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3rd International Conference on 3D Digital Imaging and Modelling*, page 145152, 2001. [cited at p. 37, 41]

[60] K.M. Sacha. Measuring the Real-Time Operating System Performance. In *Euromicro Workshop on Real-Time Systems*, pages 34–40, June 1995. [cited at p. 71]

[61] Sha, Klein, and Goodenough. *Rate Monotonic Analysis for Real-Time Systems.* MA: Kluwer Academic Publishers, 1991. [cited at p. 61]

[62] M. Silly-Chetto, T Garcia-Fernandez, and A. Marchand. Cleopatre: Open-source operating system facilities for real-time embedded applications. *Journal of Computing and Information Technology*, 15(2):131–142, 2007. [cited at p. 71]

[63] Silva, Bellon, and Boyer. Enhanced, robust genetic algoritm for multiview range image registration. In *Int.Conf. on 3D digital imaging and modeling*, 2004. [cited at p. 38]

[64] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002. [cited at p. 2, 3, 4]

[65] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *IEEE International Conference on Robotics and Automation ICRA2000*, page 321328, 2000. [cited at p. 37]

[66] uClinux Embedded Linux Microcontroller Project. http://www.uclinux.org. [cited at p. 71]

[67] G. Weiss, G. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlatin of range-finder scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS94), Munich, Germany*, page 595601, 1994. [cited at p. 37]

[68] S. Yamany, M. Ahmed, and A. Farag. A new genetic based technique for matching 3d curves and surfaces. *Pattern Recognition*, 32:1817–1820, 1999. [cited at p. 33, 38, 53, 56]

[69] Yartek Home Page. http://www.units.it/smartlab/yartek.html. [cited at p. 61, 82]

[70] Z. Zhang. Iterative point matching for registration of free-form curves. Technical report, INRIA Tech. Rep. 1658, 1992. [cited at p. 37]

Appendices

Appendix A

Exploration strategy with sonar and scan matching

This appendix briefly describes an application of the developed techniques for obtaining the map of the environment with a real mobile robot.

An autonomous mobile robot is capable of moving around on his own. When placed in a completely unknown environment it faces the problem of how to gain some initial knowledge of it. Apart from sensing the environment and controlling the motors, it must have some rules and/or a strategy of choosing where to go.

Research in robotics has offered many approaches and solutions to similar problems. Some exploration schemes base the exploration on randomness either selecting every action with uniform probability distribution or, in more complex works, using for example neural networks to change that probability distribution according to on-line acquired knowledge. Directed exploration techniques use exploration-specific knowledge usually combined with heuristics to guide the robot, mostly because in an unknown environment it is difficult to understand in advance how an action will affect the exploration. Most techniques build a representation of the environment and then use that map to plan the next positions where the robot will be directed. In [32] some well known algorithms are compared in relation to the quality of the map they produce.

In order to apply the focused ultrasonic sensor in a real exploration using scan matching to correct localization errors a simple approach to robot exploration was developed during the research. The focus of the work was not the proposal and study of new exploration methods but implementation of developed techniques to produce a real map of the environment. Nevertheless a simple algorithm that was developed enabled efficient exploration when the robot is placed in an unknown environment with the only task to explore it. The algorithm is a directed technique that allows the building of a map during the progress of the exploration, but it does not rely on it to function efficiently therefore it requires little resources.

It works in the following way: the robot uses a single rotating sensor derived from low-cost ultrasonic sensor and fuzzy logic for simple and smooth control of the motors. It scans the environment with the sensor and directs itself in the largest open direction, that is with no sonar returns. It then performs another scan and so on until the open directions become sufficiently small or exploration area boundaries are reached. The algorithm then returns, backtracking, to eventually explore other open directions.

The main loop of the algorithm is summarized in the following steps:

```
1.  doScan();

2.  currPos.S.pos=currPos.believedPos;

3.  currPos.matchedPos=match(prevPos.S,currPos.S);

4.  currPos.believedPos=currPos.S.pos=currPos.matchedPos;

5.  storePosition(currPos);

6.  updateMap(currPos.S);

7.  selectNextPos();

8.  move();

9.  prevPos.S=currPos.S;
```

1. doScan() performs a scan of the environment. In simulations it is performed from the real robot position, as opposed to the believed position i.e. where the robot thinks it is;

2. all the calculations are performed from the believed position;

3. the matching is performed against the previous scan (pose tracking). On the long run this may lead to accumulation of localization error. A different approach would perform the matching against the created global map (position estimation);

4. setting the believed position to the result of the matching process means that we are completely confident in matching! On the long run this is a wrong approach, but usually works for short explorations. See the discussion on SLAM in §1.1.1;

5. the scan is saved for future use. This enables different strategies for updating the global map. For example the solutions may be refined later when more scans become available or when we backtrack on already explored positions;

6. the map is updated according to the chosen updating strategy.

7. selectNextPos() selects the next exploration position from a set of candidate positions as explained in A.1.

8. move() calls the navigation procedure to control the robot movement and bring the robot near the selected destination A.2.

9. we are ready for the next step!

A.1 Selection of the next exploration position

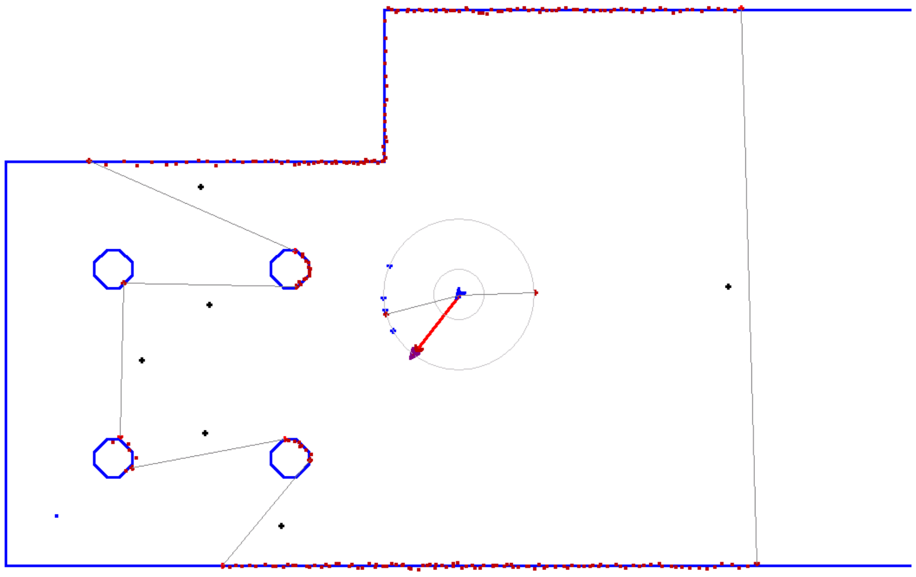


Figure A.1: The next position to direct the robot in is on the circle of radius maximum step distance from the robot. The direction is towards the position chosen between candidates depicted as black crosses. These are chosen slightly in front of open areas to guarantee some readings for the scan matching to process.

The candidate positions are chosen with the goal of maximizing the exploration area gain. This approach is similar to the longest lines approach [32]. The input of the algorithm are the sizes and directions of unexplored arcs as seen from the current position. Other optional inputs that may steer the choice are the existence of previous scans close by, the vicinity of the search area boundary and the confidence value of the last performed scan matching process. The output is the direction and range of the chosen candidate position. However every candidate position is also memorized to be revaluated during a backtracking algorithm steps. When there is nothing left to explore in one area, either because the map has no areas large enough left to be considered interesting, or when boundary of the search area is reached, then the robot moves toward other candidates left from previous positions, if they are still interesting (maybe the area has been explored from new positions).

A.2 Fuzzy control

The fuzzy system controls the movement of the robot in order to take it from Start position, which is the robot's believed position in the moment it receives the command, to or near the End position set by the exploration algorithm after each scan of the environment.

Fuzzy control has been chosen for the characteristic smooth movement of the robot that it produces and for the ease of the development using rules that are simple to describe.

For a description of the used fuzzy system refer to the [19].

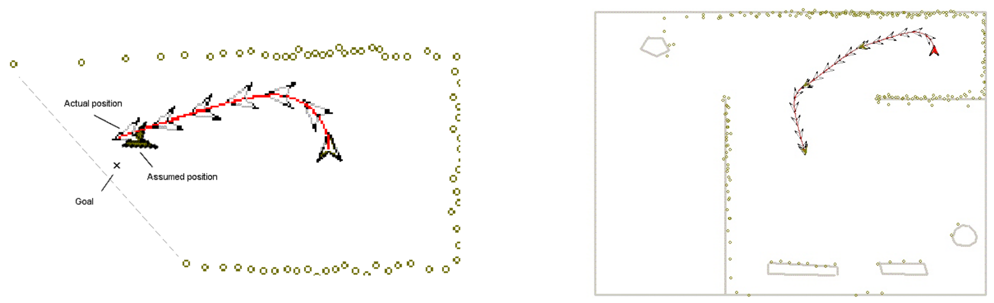


Figure A.2: The left figure shows how fuzzy control brings the robot near the designated position. On the right the robot trajectory after a few steps.

The scans collected after a few steps of exploration are shown in figure A.3.

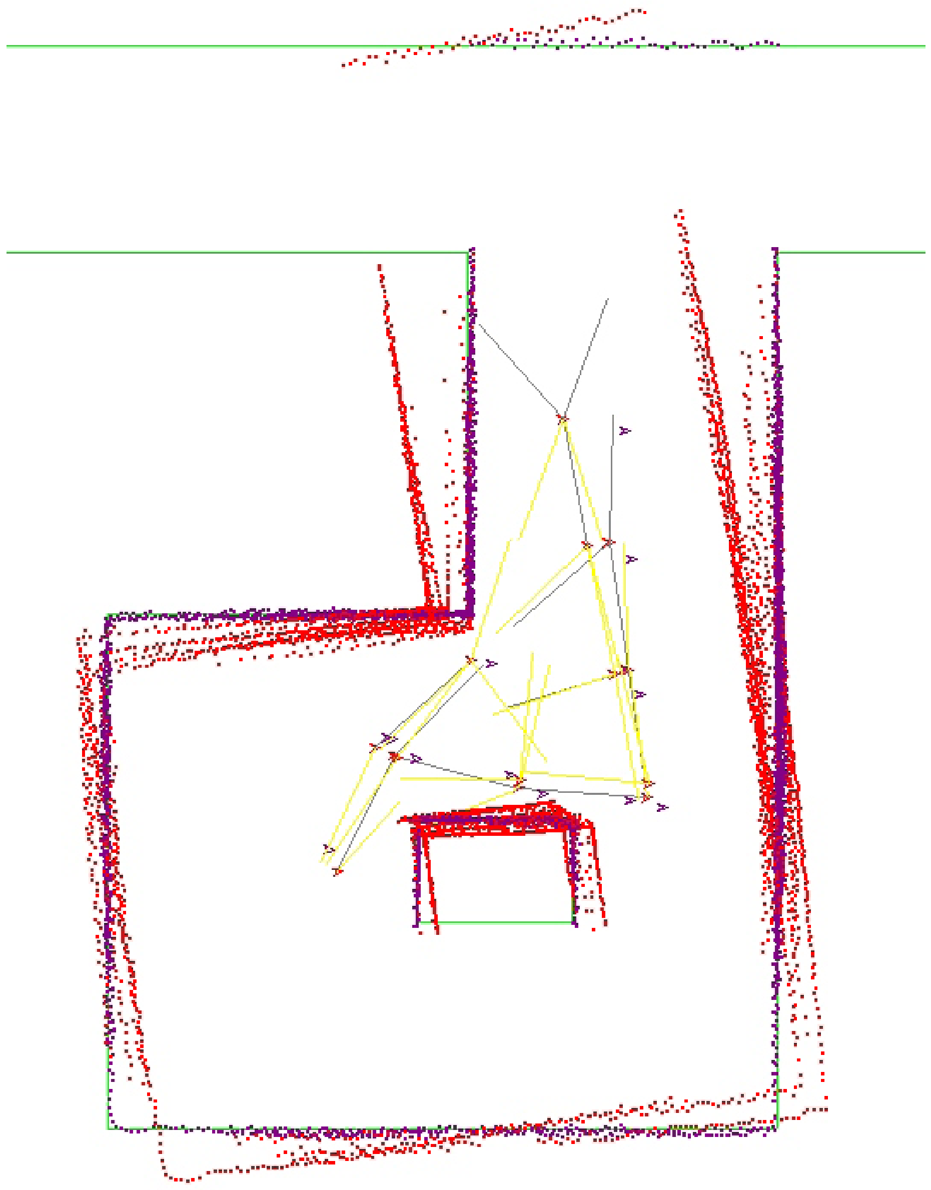


Figure A.3: Exploration in a cyclic environment: the scans not corrected by scan matching are shown (red). With scan matching the robot successfully corrects the errors and manages to complete the exploration.

Appendix B

YARTEK internals

Data structures

In the following are briefly summarized the data structures and the scheduler. To compile the application, a programmer needs to include headers of the required components.

```
/* Components interfaces */
#include <TCB.h>    /* Thread Control Block */
#include <yartek.h> /* Scheduler */
#include <irq.h>    /* Interrupt management */
```

The main data structure in Yartek is called TCB, and is defined as follows:

```
/* the TCB (Thread Control Block) type */
struct Tcb {char    SMI,        /* sending module */
            RMI,                /* receiving module */
            PR,                 /* task priority */
            PX,                 /* process index */
            DATA[4],            /* short data field */
            SR[2],              /* Cpu condition codes (SR) */
            *a[8],              /* Cpu address registers : A0-A7 */
            *d[8],              /* Cpu data registers : D0-D7 */
            *PC;                /* Cpu Program Counter */
    short   Stack,              /* Assigned Stack */
            Heap;               /* Assigned Heap */
    int     BornAbsTime,        /* Time of birth of process */
            Mem;                /* Memory address */
    char    ST_PR;              /* Starting task priority */
    void    (* fun)();          /* Pointer to the function to execute */
    unsigned int
            PID,PPID;           /* Process ID, Parent Process ID */
    CommandLine Command;        /* Record field for longer data */
    char    Name[NameDim];      /* Name of command or process */
    struct Tcb  *CP;            /* Chain pointer */
    char   Type;
    unsigned int
            Born,               /* Born time */
            Start,              /* Next start time */
            Dline,              /* Deadline */
            Period,             /* Period of the thread */
            Maxtime,            /* Maximum duration of the thread */
            Stat;               /* Duration of the previous execution */
};
typedef struct Tcb TCB;
```

The Interrupt Table is defined as follows:

```
int InterruptTable[3];          /* Interrupt Table */
```

Scheduling algorithm

The scheduling algorithm of Yartek implements an EDF non-preemptive scheduling. A representative C code of this routine for interrupt and real-time threads follows.

```
void MainLoop( )
{
  bool Found;
  register int i,j;
  TCB *p, *p1, *p2;

mainloop:

    /* ServiceInterruptTable */
    /* deferred mechanism, non real-time threads are scheduled using Call */
    /* Call takes a TCB from the free list and put it on queue 2 */
    if (InterruptTable[0]!=0)
    {
        Call( ServiceInterrupt0 );
    }
    if (InterruptTable[1]!=0)
    {
        Call( ServiceInterrupt1 );
    }
    if (InterruptTable[2]!=0)
    {
        Call( ServiceInterrupt2 );
    }

    /* ServiceTaskQueue */
    /* if a task to be activated is found on queue 1
       then concatenate task on queue 0 (using concatTCB function) */
    p=fipt[1];
    while (p!=NULL && p->Start<=RTClock)
    {
        fipt[1] = p->CP;    /* fipt: root of the queue 1 */
        if (fipt[1]==NULL) lipt[1]=NULL;
        p->PR = 0;
        p->ST_PR = 0;
        concatTCB(p);
        p = fipt[1];
    }
    if (p!=NULL)
      while (p->CP!=NULL)
    if (p->CP->Start<=RTClock)
    {
        p1=p->CP;
        p->CP=p1->CP;
        if (p->CP==NULL) lipt[1]=p;
        p1->PR = 0;
        p1->ST_PR = 0;
        concatTCB(p1);
    } else p=p->CP;

if (Found==TRUE)
{
  savedTCB=fipt[0];
}
if (savedTCB->CP == NULL)
    lipt[0] = NULL;   /* Updates the last element queue pointer */

/* EXE execute the real-time thread */
asm{
  jmp EXE;
}

/* When the real-time periodic thread has completed its period,
  then it is enqueued on queue 1 */
concatTCB(savedTCB);

goto mainloop;
```

Hints for application developer

A Yartek application developer should:

- write the code for the interrupt service routines *ServiceInterrupt0*, *ServiceInterrupt1* and *ServiceInterrupt2*;
- write the *Init( )* routine that schedules the periodic real-time threads;
- write the code of the periodic real-time threads.

Appendix C

Polaroid ultrasonic ranging module

The Polaroid 6500 ranging module is an active time-of-flight (TOF) device widely used in mobile robotics and is representative of the general characteristics of such ranging devices. Borenstein studied the device thoroughly [11] and referenced many other works in robotics using the sensor. A more recent characterization of the sensor can be found in [53].

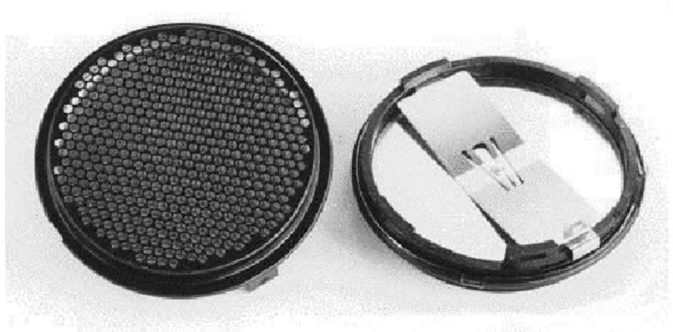


Figure C.1: Instrument-grade Polaroid electrostatic transducer

The Polaroid system uses a single transducer both for transmitting the ultrasonic burst and for listening to the echo (monostatic transceiver mode).

Recall that in the TOF method the distance is determined by multiplying the velocity of the ultrasonic burst by the time required to travel to the obstacle and back:

$$d = \frac{vt}{2}.$$

where

- d = distance
- v = speed of sound in air
- t = elapsed time.

The speed of sound is influenced by temperature changes, and to a lesser extent by humidity. Under normal conditions, the following relation may be used:

$$v = 331.4\sqrt{\frac{T}{273}} \qquad \text{m/s}$$

The system configuration consists of two components: 1) the ultrasonic transducer, and 2) the ranging module electronics.

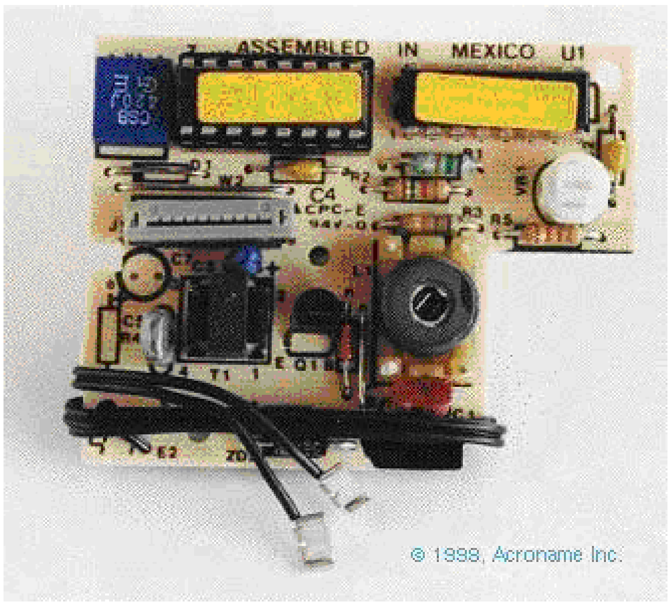


Figure C.2: Polaroid electronic interface board

The 6500 series, used in this work, is a third-generation board introduced in 1990 which provides a reduction in interface circuitry, with the ability to detect and report multiple echoes. It starts the measurement by feeding sixteen voltage transitions to the transducer at the frequency of 49.1 kHz and then remains listening for echoes until the start of the next measurement. In the experiments carried out in the thesis the minimum and maximum measured range were between 10 cm and approximately 5 m. The measured beam dispersion angle was approximately 30 degrees. A typical operating cycle is as follows.

1. The control circuitry fires the transducer and waits for indication that transmission has begun.

2. The receiver is blanked for a short period of time to prevent false detection due to ringing from residual transmit signals in the transducer.

3. The received signals are amplified with increased gain over time to compensate for the decrease in sound intensity with distance.

4. Returning echoes that exceed a fixed threshold value are recorded and the associated distances calculated from elapsed time.

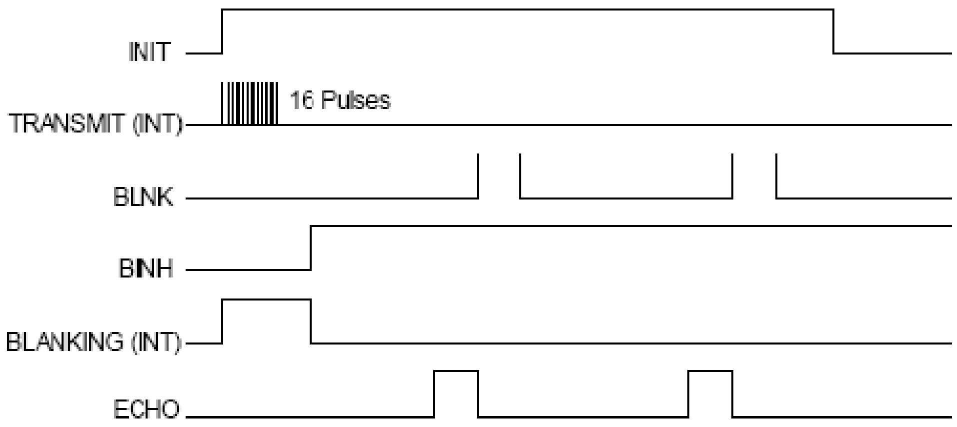


Figure C.3: timing diagram for the 6500 Sonar Ranging Module executing a multiple-echo cycle with blanking input [Polaroid, 1990]

The figure C.3 taken from the modules datasheet illustrates the operation of the sensor in a timing diagram. In the single-echo mode of operation for the 6500-series module, the blank (BLNK) and blank-inhibit (BINH) lines are held low as the initiate (INIT) line goes high to trigger the outgoing pulse train. The internal blanking (BLANKING) signal automatically goes high for 2.38 milliseconds to prevent transducer ringing from being misinterpreted as a returned echo. Once a valid return is received, the echo (ECHO) output will latch high until reset by a high-to-low transition on INIT. For multiple-echo processing, the blanking (BLNK) input must be toggled high for at least 0.44 milliseconds after detection of the first return signal to reset the echo output for the next return.

In this thesis all the digital processing of the 6500 module was ignored acquiring the signal directly on the appropriate pin of the analog processing chip on the module.

Publications by the Candidate Relevant to the Thesis

1. **E. Mumolo, M. Nolich, K. Lenac, A Real-Time Embedded Kernel for Non Visual Robotic Sensors, *EURASIP Journal on Embedded Systems*, October 2007.**

   In this paper the real-time kernel Yartek and the robotic application from the Chapter 4 are described.

2. **K. Lenac, E. Mumolo, M. Nolich, Fast Genetic Scan Matching using Corresponding Point Measurements in Mobile Robotics, *Proc. EvoIASP 2007*, 2007**

   In this paper the GLASM algorithm is introduced.

3. **E. Mumolo, K. Lenac, M. Nolich, Heuristic Optimization for Global Scan Matching with Focused Sonar on a Mobile Robot, *ITI 2006*, June 19-22, Cavtat, Croatia.**

   This paper proposed a hybrid approach for solving the problem of global scan matching (not presented in this thesis). The algorithm is composed of an estimation phase performed by exhaustive or heuristic optimizations and by a final optimization phase using a closed form expression. The result of the first phase is to obtain a sub-optimal solution by reaching a local minimum of the objective function, and the result of the closed-form expression is to improve the local minimum. The exhaustive approach for the first phase is used as a benchmark and compared to the proposed heuristic optimization using a genetic algorithm. It is shown that the different solutions bring to a trade-off in performance and computation time.

4. **E. Mumolo, K. Lenac, M. Nolich, M. OssNoser, Development of Embedded Devices in Real-Time Autonomous Robots, *ITI 2006*, June 19-22, Cavtat, Croatia.**

   In this work a development of embedded systems suited for the implementation of robotic perceptions is described. Applications are scheduled by a realtime micro-kernel, called YARTOS (Yet Another Real-Time Operating System) which was ported to the Motorola Coldfire micro-controller. This micro-kernel is the previous version of the system which was later extended with a new non-preemptive real-time scheduler as described in this thesis.

5. **E. Mumolo, K. Lenac, M. Nolich, Spatial Maps Building using Fast Texture Analysis of Rotating Sonar Sensor Data for Mobile Robot, in *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 2005.**

   This article presents the algorithm described in the Chapter 2.

6. **U. La Fata, K. Lenac, E. Mumolo, M. Nolich, Exploration of Unknown Environment with Focused Sonar Sensor and Fuzzy control, *5th WSEAS International Conference on Fuzzy Sets and Fuzzy Systems (FSFS '04)***

   In this paper the focused ultrasonic sensor is introduced. The algorithm mentioned in the Appendix A is used for the exploration of the unknown environment using the focused ultrasonic sensor, scan matching and fuzzy control for the control of the robot. An improved version of the WRSM scan matching algorithm was used for position tracking. Fuzzy control produced smooth robot trajectories.

7. **U. La Fata, K. Lenac, E. Mumolo, M. Nolich, Fuzzy Coordination of a Multi-Robot System for Audio Surveillance, *5th WSEAS International Conference on Fuzzy Sets and Fuzzy Systems (FSFS '04)***

   In this paper a multi robot system suited for acoustic surveillance of environments is presented (not presented in this thesis). The system is composed of robots with an acoustic emitter and a microphone array for detecting and localizing acoustic sources. As the robots move according to a given reconnaissance task, it is necessary to avoid collisions among them. To this end an acoustic communication protocol is implemented to transmit the robot positions. When each robot acquires the knowledge of the trajectory performed by the others, a fuzzy coordination among robots is activated to avoid collisions. Experimental results in the coordination task are reported.

The following pubblication is currently under second review:

8. **E. Mumolo, K. Lenac, M. Nolich, Fast Genetic Scan Matching in Mobile Robotics, invited chapter for a book Evolutionary Image Analysis and Signal Processing in Springer Verlag's "Studies in Computational Intelligence" series.**

   Topics from the Chapter 3 are covered.

Publications in preparation

The publications currently in preparation that originated from the thesis include:

1. **K. Lenac, E. Mumolo, M. Nolich, "On Sufficient Conditions for EDF and RM Non-preemptive Scheduling"**, written for the submission to the *Transactions on Software Engineering.*

   This paper presents a unified view of non preemptive scheduling offering further schedulability bounds (not presented in this thesis) that enable both EDF and RM (Rate Monotonic) algorithms to be used for on-line scheduling. The conditions may be implemented in an on-line scheduler for a fast (O(1)) feasibility check in response to changing environment conditions or local overload.

2. **K. Lenac, E. Mumolo, M. Nolich, "2D Dense Scan Matching Algorithms for Real-time Applications"**, written for the submission to the *Journal of Systems and Software Engineering.*

   In this paper a framework for comparison of different genetic and iterative correspondence point algorithms which are suitable for real time applications is proposed. The comparison results are presented and discussed.

3. **M. Nolich, K. Lenac, E. Mumolo, "Slam algorithm for indoor robotic navigation using genetic scan matching"**, written for the submission to the *RAS.*

   In this paper the GLASM algorithm is integrated in a SLAM framework for exploration of unknown environments.

List of Symbols and Abbreviations

Abbreviation	Description	Definition
DMA	Direct Memory Access	page 6
EDF	Earliest Deadline First	page 64
GCP	Grid Closest Point transform	page 53
GLASM	Genetic Lookup based Scan Matching Algorithm	page 44
GPS	Global Positioning System	page 4
IDC	Iterative Dual Correspondences	page 37
CCF	Cross Correlation Function	page 37
ICP	Iterative Closest Point	page 37
IPE	Initial Position Estimate	page 32
LRF	Laser Range Finder	page 32
RCD	Region of Constant Depth	page 14
RTAI	Real-Time Application Interface	page 71
RTOS	Real-Time Operating System	page 6
SLAM	Simultaneous Localization and Map building	page 5
SR	Success Ratio	page 48
TOF	Time Of Flight	page 105
VFH	Vector Field Histogramm	page 76
WRSM	Weighted Range Scan Matching	page 38

List of Figures

List of Tables

Summary

The work described in this thesis has been carried out in the context of the exploration of an unknown environment by an autonomous mobile robot. It is rather difficult to imagine a robot that is truly autonomous without being capable of acquiring a model of its environment. This model can be built by the robot exploring the environment and registering the data collected with the sensors over time. In the last decades a lot of progress has been made regarding techniques focused on environments which posses a lot of structure. This thesis contributes to the goal of extending existing techniques to unstructured environments by proposing new methods and devices for mapping in real-time.

The first part of the thesis addresses some of the problems of ultrasonic sensors which are widely used in mobile robotics for mapping and obstacle detection during exploration. Ultrasonic sensors have two main shortcomings leading to disappointing performance: uncertainty in target location and multiple reflections. The former is caused by wide beam width and the latter gives erroneous distance measurements because of the insertion of spikes not directly connected to the target. With the aim of registering a detailed contour of the environment surrounding the robot, a sensing device was developed by focusing the ultrasonic beam of the most common ultrasonic sensor to extend its range and improve the spatial resolution. Extended range makes this sensor much more suitable for mapping of outdoor environments which are typically larger. Improved spatial resolution enables the usage of recent laser scan matching techniques on the sonar scans of the environment collected with the sensor. Furthermore, an algorithm is proposed to mitigate some undesirable effects and problems of the ultrasonic sensor. The method registers the acquired raw ultrasonic signal in order to obtain a reliable mapping of the environment. A single sonar measurement consists of a number of pulses reflected by an obstacle. From a series of sensor readings at different sonar angles the sequence of pulses reflected by the environment changes according to the distance between the sensor and the environment. This results in an image of sonar reflections that can be built by representing the reading angle on the horizontal axis and the echoes acquired by the sensor on the vertical

one. The characteristics of a sonar emission result in a texture embedded in the image. The algorithm performs a 2D texture analysis of the sonar reflections image in such a way that the texture continuity is analyzed at the overall image scale, thus enabling the correction of the texture continuity by restoring weak or missing reflections. The first part of the algorithm extracts geometric semantic attributes from the image in order to enhance and correct the signal. The second part of the algorithm applies heuristic rules to find the leading pulse of the echo and to estimate the obstacle location in points where otherwise it would not be possible due to noise or lack of signal. The method overcomes inherent problems of ultrasonic sensing in case of high irregularities and missing reflections. It is suitable for map building during mobile robot exploration missions. It's main limitation is small coverage area. This area however increases during exploration as more scans are processed from different positions.

Localization and mapping problems were addressed in the second part of the thesis. The main issue in robot self-localization is how to match sensed data, acquired with devices such as laser range finders or ultrasonic range sensors, against reference map information. In particular scan matching techniques are used to correct the accumulated positional error using dead reckoning sensors like odometry and inertial sensors and thus cancel out the effects of noise on localization and mapping. Given the reference scan from a known position and the new scan in unknown or approximately known position, the scan matching algorithm should provide a position estimate which is close to the true robot position from which the new scan was acquired. A genetic based optimization algorithm that solves this problem called GLASM is proposed. It uses a novel fitness function which is based on a look up table requiring little memory to speed the search. Instead of searching for corresponding point pairs and then computing the mean of the distances between them, as in other algorithms, the fitness is directly evaluated by matching points which, after the projection on the same coordinate frame, fall in the search window around the previous scan. It has a linear computational complexity $O(N)$, whereas the algorithms based on correspondences have a quadratic cost of $O(N^2)$. The GLASM algorithm has been compared to it's closest rivals. The results of comparison are reported in the thesis and show, to summarize, that GLASM outperforms them both in speed and in matching success ratio. Glasm is suitable for implementation in feature-poor environments and robust to high sensor noise, as is the case with the sonar readings used in this thesis which are much noisier than laser scanners. The algorithm does not place a high computational burden on the processor, which is important for real world applications where the power consumption is a concern, and scales easily on multiprocessor systems. The algorithm does not require an initial position estimate and is suitable for unstructured environments.

In mobile robotics it is critical to evaluate the above mentioned methods and devices in real world applications on systems with limited power and computational resources. In the third part of the thesis some new theoretical results are derived concerning open problems in non-preemptive scheduling of periodic tasks on a uniprocessor.

This results are then used to propose a design methodology which is used in an application on a mobile robot. The mobile robot is equipped with an embedded system running a new real-time kernel called Yartek with a non-preemptive scheduler of periodic tasks. The application is described and some preliminary mapping results are presented. The real-time operating system has been developed in a collaborative work for an embedded platform based on a Coldfire microcontroller. The operating system allows the creation and running of tasks and offers a dynamic management of a contiguous memory using a first-fit criterion. The tasks can be real-time periodic scheduled with non-preemptive EDF, or non real-time. In order to improve the usability of the system, a RAM-disk is included: it is actually an array defined in the main memory and managed using pointers, therefore its operation is very fast. The goal was to realize small autonomous embedded system for implementing real-time algorithms for non visual robotic sensors, such as infrared, tactile, inertial devices or ultrasonic proximity sensors. The system provides the processing requested by non visual sensors without imposing a computation burden on the main processor of the robot. In particular, the embedded system described in this thesis provides the robot with the environmental map acquired with the ultrasonic sensors. Yartek has low footprint and low overhead. In order to compare Yartek with another operating system a porting of RTAI for Linux has been performed on the Avnet M5282EVB board and testing procedures were implemented. Tests regarding context switch time, jitter time and interrupt latency time are reported to describe the performance of Yartek.

The contributions of this thesis include the presentation of new algorithms and devices, their applications and also some theoretical results.

They are briefly summarized as:

- A focused ultrasonic sensing device is developed and used in mapping applications.
- An algorithm that processes the ultrasonic readings in order to develop a reliable map of the environment is presented.
- A new genetic algorithm for scan matching called GLASM is proposed.
- Schedulability conditions for non-preemptive scheduling in a hard real-time operating system are introduced and a design methodology is proposed.
- A real-time kernel for embedded systems in mobile robotics is presented.

- A practical robotic application is described and implementation details and trade-offs are explained.