Control of a hierarchical team of robots for Urban Search and Rescue

by

Thomas Mirko Roehr

A thesis
submitted to the Victoria University of Wellington
in partial fulfilment of the
requirements for the degree of
Master of Science
in Computer Science.

Victoria University of Wellington
2008

Abstract

Research teams worldwide are researching the application of robots for Urban Search and Rescue (USAR) operations and some are using teams of robots. The Mechatronics Research Group of Victoria University of Wellington is developing a low cost architecture of a team of USAR robots that is hierarchically structured and can operate autonomously. The objective of this thesis is to design the autonomous control system for the proposed architecture.

The overall system design and combination of hardware and software solutions needs to be evaluated in a realistic environment. The project could not perform tests in a real environment and developed a realistic simulation environment instead to allow the evaluation of hardware and software constraints.

This project successfully developed an incremental mapping algorithm which served as foundation for distributed path planning, and modified an existing navigation approach to cope with the main challenges of 3D operation environments. In order to deal with multiple robots, this thesis applied a centralised control mechanism and a combination of a global and local exploration strategy.

This thesis contributes software solutions to operate the low cost robot architecture and identified weaknesses in the design of the middle tier of robots. The individual algorithms, and their combination in a major control system proved to be effective, but not without limitations. Consequently, this thesis suggests solutions to overcome some of these limitations.

Acknowledgements

This thesis has taken me on a journey into the world of mobile robotics. This would not have been possible without the support of my supervisors Peter Andreae and Dale Carnegie. My gratitude goes to both for cooperating with each other and offering me this unique opportunity. Specifically I would like to express my sincere appreciation to Peter Andreae for his continuous support and encouragement throughout this thesis, and to Dale Carnegie for offering a second point of view and help with clarifying my ideas.

I want to thank my fellow students Aleks, Adrian and Caleb for help and distraction at the right time. I am also grateful to Chris for sharing his ideas, and Praneel for stimulating discussions.

Many thanks to Jason, Scotty, Johnny and all the other people of the Mechatronics team who offered me a friendly and entertaining work environment, and my appreciation to those not named here and who showed interest in my work.

Finally, I take this opportunity to express my profound gratitude to my beloved parents, brother and sister for their support from half around the globe.

Contents

List of Figures

List of Tables

The journey is the reward

Chapter 1

Introduction

1.1 General research objective

New Zealanders live under the constant threat of natural hazards [EG08] and especially earthquakes. The reason is their country's unique location at the boundary of the Australian and the Pacific tectonic plates. Although most of the earthquakes are rather small, it has been estimated that the chance of a rupture of the Alpine Fault within the next 50 years lies at 65% [Min08]. The risk for the populated region of Wellington is only 10% and it has been estimated that about 750 people might be trapped in such an event. However, recent events in China [Wik08] have shown that the number of victims can be significantly higher in more densely populated areas. Rescue specialist teams will therefore be required at a number of affected sites to find victims in the rubble, rescue them and give initial medical treatment.

In order to support such teams and international equivalents, the Mechatronics Research Group of Victoria University of Wellington is currently developing a low cost multi-tier architecture for robots involved with Urban Search and Rescue (USAR) [WC06]. Though the robots cannot replace human rescuers, they are able to provide valuable information about a disaster aftermath area. Even more, they allow early penetration of the disaster site without putting human rescuers at risk, until the site is prepared for the rescue teams. Thus, the primary goal of this USAR team of robots is to locate human victims in a disaster site.

In order to achieve its goal, the team of robots has to successfully explore and map the destruction area. Furthermore, the team of robots needs to be manageable with minimum effort and number of operators, so that it has to perform its tasks autonomously in order to provide a real value for an USAR operation.

Efforts to build robot rescue teams are being undertaken world wide, but financial considerations can play an important role even in catastrophic events. Hence, for the proposed architecture the aspect of low cost is of major importance, because the application of low cost devices makes a potential loss of individual robots affordable. But this aspect will also have a significant impact on the capabilities of the individual robots.

The proposed robot architecture is based on three different types of robots: grandmother, mothers and daughters. Each is designed to fulfil a special task in a search and rescue scenario: the single grandmother for instance, allows high level and centralised control close to the operating area of the mother robots. Meanwhile, multiple mother robots are intended to penetrate into and explore the operation area. Areas inaccessible due to size constraints of the mother robot can then be investigated by daughter robots deployed by the mother robots — specifically areas under the main surface.

A design study has been completed for the real mother robot and currently a single proof-of-concept of this unit exists. The mother robot is equipped with a basic set of sensors to allow self-localisation and the sensing of obstacles in its close environment, and therefore fulfils minimum requirements for autonomous operation. The design for the daughter robots has not been finalised yet.

The Mechatronics Research Group of Victoria University of Wellington has recently completed related projects which deal with the investigation of emotion based navigational control and efficient multi-task allocation for a group of robots [LJCC07] [LJC07]. This project will embed some results of this research.

1.2 Thesis objective

The objective of this thesis is to advance the current state of the team of robots for USAR and develop autonomous control for the search team down to the mother robots. Currently, only a low level control system exists for the mother robots which allows the manual operation of the robots. Hence, the primary objective of this thesis is to design a high level control system for the hierarchy of robots. This control system should allow for autonomous operation of an individual mother robot and cooperating ones in an operation area similar to a field of rubble. The construction of such a system also involves the design of a communication structure between grandmother and mother robots.

It is critical for the success of this project to deal with the following challenges:

- Limited (and also unique) navigational capabilities of the mother robots which heavily influence the selection of current algorithms and the design of new ones.

- Mother robots will operate in a complex 3D environment with arbitrary shaped objects, ramps and drops, while many common solutions for mapping and navigation only address planar environments.

- An exploring unit will initially have no previous information about the environment and has to work autonomously. Thus, it has to build its own map of the operation environment.

- With the lack of real world experiments and only one real mother robot, a simulation is required which needs to be as realistic as possible to be a valuable development, analysis and evaluation tool.

- Multiple robots will explore at the same time, so that a coordinated exploration strategy is required.

Autonomous operation is a fundamental problem in the field of robotics and incorporates a range of research fields. Common and recent advances in different areas of robotics will be the foundation of this project. Modifications to existing approaches will focus on advancing the proposed architecture, but will not aim at completely generalisable solutions.

This project represents the first holistic approach towards autonomous control for the presented robot architecture and will therefore heavily influence the direction of future research projects. Thus, its design decisions have to be made carefully to avoid unnecessary and early restrictions for future projects.

1.3 Contributions of this thesis

This thesis makes following major contributions:

Control

A control system for the hierarchical group of robots consisting of the grandmother robot and multiple mother robots has been designed, implemented and successfully evaluated. The control system includes new aspects of area mapping, navigation and exploration in order to address the special requirements of the robot hierarchy and a cluttered 3D operational environment. It also presents an approach to including the role of an operator to monitor, give advice and eventually improve the performance of a multi robot USAR operation without breaking the idea of an autonomously operating team of robots.

Simulation

This project built a reusable and extensible simulation for a multi robot architecture. This simulation allowed preliminary testing of control algorithms as well as the examination of physical design properties. This project used the simulation successfully to develop the control system 'ExploreTM' and was also able to point out limitations of the current robot hardware design. Subsequently, improvements have been suggested to the hardware design which can be realised with minimal effort.

General

This thesis represents an holistic approach to building autonomous control for the hierarchy of robots and provides substantial modules for mapping, path planning, exploration and navigation. Thus, this project builds the foundation and context for subsequent and incremental improvements of this newly built control architecture, as well as an environment to study alternative hardware designs.

1.4 Thesis structure

Chapter 2 offers an introduction into the background and main techniques involved in this project. A system overview and essential context for subsequent chapters is given in Chapter 3 and Chapter 4 explains the setup of a simulation environment. This project justifies its assumption about localisation in Chapter 5 and introduces an incremental 3D mapping approach in Chapter 6. This approach serves as a foundation for the concept of distributed path planning explained in Chapter 7. A description of the navigational approach EmotioNav and the new modifications are given in Chapter 8. The final building block is presented in Chapter 9, where the design of the exploration strategy is outlined. The final conclusions are presented in Chapter 10.

Chapter 2

Background

This thesis deals with a range of research fields and bases several of its ideas upon these or thereof derived methods. The following chapter introduces concepts, ideas and specific techniques which build the foundation of this thesis or have been a major source of inspiration.

2.1 Search and Rescue

Search and rescue deals with the location, rescue and the initial medical treatment of victims, who are trapped in areas with limited accessibility. Most often this involves buildings with structural collapse [FEM08]. Search and rescue is applicable to multiple hazard scenarios not limited to earth and land slips, earthquakes and floods.

2.1.1 Different types of search and rescue operations

Search and rescue has come to a new focus of interest in the past few years, mainly due to the effect of natural disasters and acts of terrorism. Some of the first systematic approaches toward search and rescue were developed during the second World War, mainly to find hostile submarines. In later years the application of search theory resulted in the successful spotting of a H-Bomb previously lost in the ocean near Spain [Sto75]. These examples show search and rescue operations with two main characteristics:

1. run on the open sea, i.e. an ”even” surface

2. large and remote search area

The scanning of ocean ground areas is majorly difficult regarding the sensing devices, which are affected by scattering and water as dynamic propagation medium. The application of a systematic search, however, can be done more easily - considering favourable weather. The reason for that is an even surface to travel on - the ocean - and a mainly unobstructed field of search.

Search and rescue operation on land are quite different in that sense, e.g. finding lost trampers would be a typical example for New Zealand, because areas can be inaccessible or significantly obstructed. The main characteristics for these type of land operations are:

1. run in obstructed, often forested areas
2. large and remote search areas

Areas need to be scanned with a large number of human helpers to achieve a desirable coverage, because many areas cannot be searched with the help of motorised vehicles or are too forested to be searched with aerial support. For urban search and rescue operations, in comparison with these scenarios, the matter of obstruction is also one of the main issues. Buildings with structural collapse pose a threat to searchers and support equipment, due to the possible effects of loose components especially in the likely case of aftershocks following a major earthquake. A smaller area of search can be seen as advantage, though it usually has the drawback of greater complexity considering the level of obstruction. Even more, the area of search can vary significantly from the footprint of a small house, to the collapsed site of a skyscraper.

2.1.2 Strategies for search and rescue

One of the initial thoughts for large scale search and rescue operations is the application of an effective strategy to narrow the area of search. For example, but without scientific foundations, one of the basic rules in search and rescue on land is the thoroughness of search, i.e. coverage has highest priority while revisiting of places does not happen [Gal81].

Theory of search [Sto75], however, provides a mathematical base to develop strategies for searching large scale areas and as such tries to increase the probability of a successful finding.

In order to build an optimized strategy, information about the last known location of the item of interest is gathered and modelled in a probability distribution over the search area. Under consideration of the available amount of time, a strategy is developed to control the optimal effort.

The findings here will serve as a motivation for a partially informed exploration strategy as discussed in Chapter 9.

2.1.3 Standards for urban search and rescue robots

As part of the efforts to construct urban search and rescue robots, the US Department of Homeland Security has developed a catalogue of criteria [Dep08] for this kind of robot. This catalogue allows the comparison of robot performances in competitions and at the same time, gives a brief insight into the motivations and real challenges of robotic search and rescue. The tasks of teams of robots involved in search and rescue defined by the National Institute of Standards and Technology as part of the Department of Homeland Security and relevant for this project are:

- negotiating compromised and collapsed structures autonomously
- finding victims (and reporting their condition)
- produce a human readable (practical) map of victim locations
- identify hazards

The aspects mentioned here are mainly part of a pure search task, while the catalogue definitions are broader and cover the topic of physical rescue as well.

The Department also stresses that currently performance metrics exist only for the above mentioned aspects, but future versions of the catalogue will incorporate metrics for rescue. The following list is an extract of the performance metrics list [Dep08] considered to be relevant for this project:

(a) In the category of mobility and locomotion:
Sustained speed, endurance and tumble recovery (considering none, self-righting and invertible continuous operations) are the main metrics. They are measured during the traversal of different terrain types, e.g. plane but obstructed areas or an inclined terrain.

(b) For the design quality of human-system interaction:
The time for initial training to control the robots and a proficiency education are measured in hours and years. Furthermore the usability (e.g. remotely controlling the robot) is a metric based on an effectiveness evaluation, but assistive elements such as automatic notifications or path tracing are simply tested on their existence.

(c) Sensing metrics:
Performance evaluation in this category concentrate on the fact, that the robot provides spatial modeling and allows or provides a waypoint annotation.

For performance evaluation, some of these aspects are applied to different types of environments or specific situations, e.g. measuring sustained speed while moving over a step or alternatively over rough, inclined terrain.

The given evaluation gives the basic motivation and criteria to enhance, develop and measure robotic control and also suggest a basic operator interface. This project will not be able to extensively measure its approach against all the given criteria. However, it will provide the simulation environment to do so.

2.2 Robot perception

Perception is of major importance for a robot to gain information about its environment. Only once information has been perceived, can it be processed and analysed to take appropriate actions.

What a robot can perceive is defined by its set of sensors and this project deals with a robot type which is equipped with infrared range finders, an inertial measurement unit, wheel encoders, a single camera and a localisation unit. Note that out of these sensors, this project will mainly take the infrared range finders into consideration, while making assumptions for localisation in general and leaving the integration of the camera unit completely to other projects. Clearly, augmenting perception with camera input would be a benefit. Nonetheless, a single camera does not easily generate depth information and therefore range finders are of greater importance in the first place.

Robot perception is critical when a robot has to act autonomously, considering that every action is based on the information perceived. Therefore, autonomous mobile robots are a great challenge, even more when operating in complex environments. The robot type in this project has a limited and unique field of perception, which requires special techniques for compensation to allow its operation in a cluttered environment. The project implements rather specific techniques to deal with the limitations of the current hardware platform. That means, that generalisation cannot be guaranteed and a transfer of these techniques needs thorough evaluation. This project also tries to keep changes to the actual hardware configuration to a minimum, with the purpose to keep to overall cost of the units, which is a major design criteria, down.

2.3 Mapping

Robot mapping describes the process of storing spatial information about the environment while a robot travels through it. Different techniques exist for map building. Distance information is necessary and range finder devices can provide information, so that free space and obstacles can be detected and classified. In order to store spatial information the robot constructs a map. Two different types of map representations exist [Mur00]:

(1) Topological or landmark based map

(2) Metric or continuous-valued map

A topological or landmark based map stores information about significant points in the environment, i.e. so-called landmarks and keeps track of their relation to each other. Topological maps are therefore a graph-like representation of the world, consisting of nodes and connecting arcs. Topological maps are also known as the feature-based approach. This method relies on identification of recognisable landmarks and because no metric data is stored a unique representation for each landmark has to be found to allow recognition.

In contrast, the iconic approach works with metric or continuous-valued maps. These maps offer an exact representation of the environment often down to a certain resolution. The closed-world assumption [SN04] applies here, saying that everything not defined as obstacle is free space. This assumption makes this type of representation more memory efficient.

In addition, grid cell decomposition is widely applied to metric maps, though by using fixed cell decomposition the advantage of memory efficiency is given away, i.e. larger maps require more cells to be stored in memory. Also by definition, cell decomposition represents details only down to a predefined resolution. This can cause details such as narrow pathways to become lost in this representation, depending on the degree of resolution.

To reduce memory constraints and allow a human readable map at the same time, the mapping strategy of this project relies on a metric map with a customisable resolution (see Chapter 6).

2.3.1 Occupancy grid

Occupancy grids [Elf89] (same meaning as certainty or evidence grids) belonging to the iconic approach are widely used [LAJ04] [BMF+00]. An occupancy grid represents the environment with (uniform) grid cells with a binary state. A value 0 represents the state of an unoccupied cell, while a value of 1 is translated into an occupied cell, i.e. the cell represents (part of) an obstacle.

Occupancy grids gained high popularity in the representation of two dimensional environments. However, they can be expanded to represent three dimensions. In that case a three dimensional grid is used, where each cell represents a cube. According to the two dimensional version, the cell state describes the occupancy.

The approach of this thesis towards mapping will maintain the idea of occupancy grids by integrating a representation for free and occupied space.

2.3.2 Elevation maps

An elevation or height map stores height values for a given area and hence, allows the computation of surfaces (usually after triangulation). Although an elevation map does not need to be based on a grid, an underlying grid simplifies computation. Hence considering an underlying uniform grid, each cell stores a single height value and achieves a two and a half dimensional representation. Compared to a three dimensional representation, the third dimension is only represented by a single height value. Effectively, the elevation map can only describe a three dimensional surface, but not a complete three dimensional structure like a building.

Though measurements of occupied space and free space do not instantly give information about total object heights, a bounded height measure can be produced. This measure represents a range, e.g. $[1.5, 2]$ represents knowledge of minimum height of 1.5 m due to occupied space measurements up to this height and a maximum object height of 2 m due to free space measurements above 2 m. [MPV00] shows how to build the final elevation map with this bounded error.

The idea of combining knowledge about occupied space and free space is the main motivation to make changes to the map representation presented in the following section.

2.3.3 Multi-Level-Surface map

In order to represent 3D space efficiently, Triebel suggests the Multi-Level Surface Map (MLS map) [Tri07][TPB06]. A MLS map can represent a three dimensional space with reduced memory constraints using uniform grid cells in two dimensions. The third dimension is defined by so called surface patches. Each patch describes a surface at a certain height accompanied by a factor of uncertainty. Triebel classifies patches into horizontal and vertical patches in order to distinguish between plain surfaces and patches with a certain depth - effectively blocks. The depth value is associated with both patch types, but equals 0 for horizontal patches. Figure 2.1 illustrates this mapping model. The most important feature of this representation is, that a single cell can contain multiple surface patches.

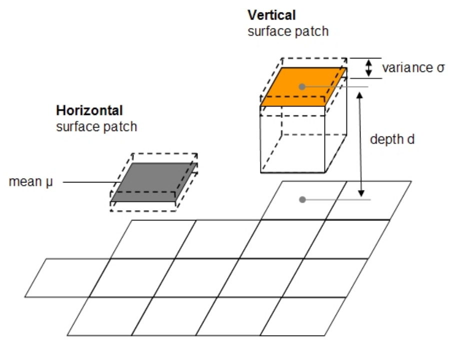


Figure 2.1: Horizontal and vertical surface patches according to [TPB06]

This grid representation is the base of the mapping algorithm in this project, but due to some specific constraints this approach will be modified as discussed in Chapter 6.

2.4 Uncertainty

Imperfect sensors and approximate knowledge about the current sensor position, e.g. when mounted on a mobile robot leads to measurement uncertainties. Different compensation techniques exist for dealing with uncertainty in this context. One commonly used method is the Bayesian approach. For example the occupancy grid is commonly used in combination with a binary Bayes filter [TBF05]. To reflect uncertainties each cell of the occupancy grid is mathematically described by a probability variable in the range of $[0, 1]$. Free and occupied cells are represented by a given single probability value $p(x)$ as follows:

$$occupied(x) = p(x)$$

$$free(x) = 1 - p(x)$$

The log odds representation simplifies computation, translating the value from range $[0, 1]$ into range $[-\infty, +\infty]$:

$$l(x) = log \frac{p(x)}{1 - p(x)}$$

$$occupied(x) = 1 - \frac{1}{1 + \exp(l(x))}$$

An occupancy grid integrates measurement data z_t of one cell over time. With each new sensor measurement comes a probability value $p_t(x)$ (or $l_t(x)$ in log odds representation) describing the belief that a specific cell occupied. Each cell's occupancy probability value is then updated on basis of the Bayes rule [RN03] and using the log odds representation:

$$l_t(x) = l_{t-1}(x) + log \frac{p(x|z_t)}{1 - p(x|z_t)} - l_0(x)$$

The probability of a measurement value can be calculated from an inverse sensor model. This sensor model describes the likelihood of measurements after the measurement has taken place. Each sensor has its own characteristics, so that this model is unique for each hardware. The inverse sensor model can be approximated using neural networks [TBF05].

Though this project does not implement an inverse sensor model, the implementation includes the above model to provide a measure a certainty based on default belief about the measurement quality in Chapter 6.

2.5 Simultaneous localisation and mapping

Without the availability of a map a robot will not have any knowledge about the environment. Furthermore, it cannot know where it is in the map and eventually has to build its own map, while keeping track of its relative position. Different approaches have been made to solve the problem as a whole under the name of simultaneous localization and mapping (SLAM)[DWB06][BDW06]. In SLAM a robot tries to solve two tasks at the same time:

(a) find the current position (and orientation) in the environment

(b) build a map of the environment

Solving the two tasks simultaneously is not trivial, because the environment is initially unknown, so that probabilistic techniques are essential for SLAM algorithms [TBF05]. The most common technique applied is Kalman Filtering [DJ00] or extended versions of it [GL06]. Particle filters for FastSLAM [TBF05] or information filters offer further alternatives [FKK+06]. Part of open challenges in this area as defined by Castellanos *et al.* [GL06] are large environments and multi-vehicle SLAM. Because of the complexity of this issue, this project does not address SLAM. The localisation problem is assumed to be solved by any of the methods given in Chapter 5 and efforts focus on mapping, path planning and navigation.

2.6 Noisy sensor measurements

Sensing devices do not operate perfectly, i.e. produced measurements do not always reflect the real state of the world, due to noise in the environment and also due to certain characteristics of the sensing technology. Erroneous measurements of an active range sensor (IR) can be caused by reflective surfaces or ambient lighting conditions and also depend on the distance to an object [Jon04]. The real world will not deliver perfect measurements as a simulation might do. Therefore the integration of an error model into the simulation is required for a realistic scenario. Here, the term noise will refer to noise and sensor error.

Modelling sensor noise

Different ways to simulate noise exists. The most simplistic way is to base noise simply on random numbers. But usually, when noise cannot be classified, a specific noise distribution is assumed. For a situation with no additional information, a Gaussian distribution (normal distribution) is the common choice. Only looking at the data sheet error specification of the mother robots hardware devices, supports the application of Gaussian error distribution. The latter one is clearly not the most supportive argument, because it only characterises the sensor error itself and the noisy environment needs to be still accounted for.

Thrun *et al.* [TBF05], however, suggest a model for range finders, which takes the existence of four different types of measurement errors into account:

- small measurement noise
- errors cause by unexpected objects
- errors based on the failure to detect objects
- random noise

They create a probability distribution for each of the mentioned situations and mix those distributions, to finally reproduce the most likely measurement. This means that the measurement model is based on four different densities. So instead of using a pure Gaussian distribution, this offers a better representation of world noise and sensor errors.

In order to produce a noisy measurement, the process takes an actual real measurement (from the simulation) to create the appropriate mixed distribution and sample from it. The sample is then the final measurement.

Previous experimentation results from [Wil07] and device specifications allow the definition of the individual distributions in this project.

2.7 Path planning

As soon as a map of the environment exists a path can be planned to allow a robot to fulfil specific tasks. Path finding is a well researched area and a wide selection of search algorithms [RN03] is available.

In general, path planning algorithms can operate with different space representations. Most representations use cell decomposition with a fixed number of cells to divide the search space. Most obvious is a decomposition into uniform grid cells. Skeletonisation of an existing map, can serve as a preprocessing step and means the reduction of the search space for the planning algorithm. Clearly, a reduced search space has performance advantages and offers an easier application of existing path planning algorithms. Visibility graphs, Voronoi diagrams and probabilistic roadmaps belong to this category [RN03] and generate road maps [SN04] through free space. This effort reduces the complexity to find a path so that graph search algorithms can be easily applied [GMAM06].

2.7.1 A*

A* is the generalisation of the shortest path algorithm or Dijkstra's algorithm [Mil06]. Also called best first search, A* is a complete algorithm, i.e. if there is a solution the algorithm will find it. The key part of the A* algorithm is an estimation function based on an heuristic which is also admissible. An heuristic estimates the cost of travelling between nodes and it is admissible if it does not overestimate the costs. The estimation function f subsequently estimates the total path costs:

$$\begin{aligned} f(n_{start}, n_{current}, n_{goal}) = \quad & c(n_{start}, n_{current}) + \\ & h(n_{curr}, n_{goal}) \end{aligned}$$

where

$f(x)$	estimation function
$c(x)$	cost function
$h(x)$	heuristic, i.e. cost estimation function
n_x	a graph node

In order to be a so-called optimal algorithm, the overall path costs of a solution have to be minimal. To achieve this, the algorithm keeps track of the path cost always knowing the actual costs up to the current node. The heuristic then estimates the travel costs from the current node to the goal, i.e. the last and yet unknown path section. Different heuristics can be applied [Mil06]:

- euclidean distance: most common and either produces the exact or underestimated cost
- cluster heuristic: region based distance calculation
- null heuristic (always return zero costs) results in the Dijkstra algorithm

- custom heuristic: application/domain dependant heuristic

Only an underestimating heuristic is guaranteed to provide an optimal solution. However, overestimating heuristics can be used, but they corrupt the optimality of A* algorithm. They will not guarantee to find the lowest-cost path and are likely to create significantly worse results if the heuristic is not close to optimal [Mil06].

2.7.2 Dynamic A*

Despite its popularity A* has some drawbacks. One of the major disadvantages is its dependency on a static environment. But mobile robots often have to operate in dynamic environments. In addition imperfect sensors cause information about the environment to vary. A* is a graph-based algorithm, i.e. it uses nodes and links with associated costs to find a solution, but, as previously discussed, arc costs can change and any change or new information can be critical for a planned path requiring replanning. To cut down the computational effort for rebuilding the whole graph and replanning each time a change of costs occurs, dynamic A* (D*) [Ste95] allows the propagation of changes using the old graph.

A number of extensions to D* exist to address further issues with cell decomposition, e.g. generate a smoother path in 3D by using interpolation [CFS06] or using multiple resolutions [FS06] to minimise memory requirements.

D* has been part of the framework GESTALT (Grid-based Estimation of Surface Traversability Applied to Local Terrain) [MBT+06] to give real time replanning capability to the Mars rovers. With the same functionality of D* but with a slightly higher efficiency and a simpler algorithm, Focussed Dynamic A* Lite (D* Lite) [KL02] is based on lifelong planning A* [KLF04]. A subsequent development is Anytime D* [LFG+05], which combines anytime and incremental planners, generating suboptimal results under time constraints.

This project used A* and D* Lite for the current implementation. A* applies to the centralised planning on the grandmother robot and D* Lite to the mother robots. Details are provided in Chapter 7.

2.7.3 Others

Wavefront planner

Wavefront planners are a group of search algorithms which can be easily applied to regular grids. A wavefront planner such as Trulla [Mur00] searches the configuration space based on the idea of how heat propagates through a material. Different material properties may allow a faster progression of heat or might slow it down. Material properties in a path planner for example, can be a traversability factor, which describes the difficulty of terrain. Wavefront planners can be computationally expensive, as they will search in all directions in a similar way to breadth first search. Breadth first search is a standard graph search algorithm. It does an exhaustive search and creating a set a unexplored nodes from all neighbours of the currently explored nodes, which are then explored before repeating this process.

Path planning with potential fields

Based on the idea of a potential field in electromagnetism, an artificial potential field represents a map as a field of force vectors which influence the path finding decisions of the robot. Forces depend on the robot's current position in the map [SN04], so that for each map position an influence vector steers the robot in a certain direction. In order to compute a potential field, existing obstacles and goal are known influences associated with values of attraction. The goal point for example, has the highest attraction, while obstacles will have negative attraction to create repulsion. Potential fields can be used to create smooth trajectories, but they suffer from local minima, oscillations and the requirement that behaviours have to be encoded as potential fields as well.

2.7.4 Traversability

A traversability analysis can add additional information to a map, which can then subsequently be used for generating more secure paths. Different methods exists to determine this value: Joho et. al. [JSPB07] are using slope, roughness and a value to account for neighbouring obstacles. Using a

Multi-Level-Surface-Grid the slope is calculated by fitting a plane to a (3x3) section of patches and comparing it to a horizontal plane without elevation. Roughness takes the differences between the neighbouring cells in this field of cells into account and uses the averaged sum of squared height distances between neighbour patches to calculate a numeric measure. The final traversability value is then computed using linear interpolation and lies in the range of [0,1]:

$$traversability = roughness * slope * obstacle$$

Joho et. al. further convolve the result with a Gaussian kernel while giving non existing neighbours a traversability value of 0.5. The convolution causes obstacle growing when a single patch has an assigned traversability value of 0. Obstacle growing means, that the area of cells occupied by an obstacle is enlarged by filtering, due to the mathematical properties of the filter algorithm.

The GESTALT planner [MBT+06] works with similar attributes as the mentioned approach and fits a plane to an area roughly the size of the robot determining tilt and roughness of this plane. It is based on the generic framework described in [USN03]. Additionally, a comparison with the best plane fit is done, where the best plane is calculated from the weighted squared means of measurement data in this area. This finally classifies the area in terms of being impassable or passable with fuzzy degrees of difficulty. The GESTALT planner uses the traversability analysis as an additional criteria to decide between a number of generated possible paths.

2.8 Exploration

Exploration of areas can be performed in a number of different ways. The group of deterministic approaches toward exploration contains strategies such as seedspreader, star or concentric exploration [SD03]. Nonetheless, the application of these strategies is constrained to easily traversable terrain, e.g. a plane office space. In rough terrain frequent replanning will be necessary, because a predetermined path based on partial knowledge might be too hard to follow or it turns out to be not traversable at all. Enforcing any deterministic strategy would therefore slow down the exploration time significantly. Furthermore, the mentioned deterministic approaches work completely uninformed, so that they cannot be effectively applied to a group of robots.

One of the most popular strategies, frontier based exploration [Yam97], is based on the analysis of a grid map to find unexplored but accessible areas. A robot extracts cells of its internal grid map representation, which connect to at least one unexplored cell. It then marks each of these cells as frontier cells, as long as the current cell is not occupied, i.e. is not part of an obstacle. The result is an unsorted collection of frontier cells, which then can be grouped into frontier regions. Aiming to reduce the total number of regions, a threshold (domain dependant and empirical) is applied to the size of frontier regions. This leaves a set of selectable frontier regions, i.e. new promising target areas for the robot.

This approach has proven to be effective for a single robot and can be applied to collaborating robots as well. For a multi robot scenario some modifications to the frontier based approach can be made to improve results. For example, each robot will be greedily assigned a frontier region with the best utility cost tradeoff, so that each robot has a different target [BMF+00]. A utility cost function can incorporate the euclidean distances between robot and frontiers, cost from a path planner to reach the frontiers or the amount of newly explored terrain upon arrival at the assigned frontier region. Joho *et. al.*[JSPB07] for example use an approach calculating the benefit of a number of new viewpoints (frontier regions) and choose the most promising one.

In general the frontier based approach allows utility measures, but it needs to associate this measure with frontier cells. Meanwhile a more general technique is to attach utility to cells and regions, whether they are part of a frontier or not. On the basis of this utility, path planners can generate optimised paths to subsequently optimise the exploration of a region. To compute a combined utility, i.e. one which takes different criteria into account, influence maps offer a good solution.

2.8.1 Influence Maps

A powerful tool for real time strategy games are influence maps [Mil06]. Influence maps allow the quick analysis of regional influences and thereby provide important data for strategic or tactical decisions. Games use influence maps usually to calculate military strength in a specific area. The computation of influence is mainly based on a distance measure from the influencing item:

$$\Phi_{cell} = \frac{\phi_n}{1 + d}$$

where

Φ_{cell}	Influence on the current cell
ϕ_n	Influence of the (source) object n
d	Distance between current cell and object

The given case presents an influence with non-linear degradation over distance. A limited radius of effect can also be taken into consideration to set a maximum distance for influence. Alternatively to computation by distance, convolution filters such as Gaussian filters can calculate the cell influence on states of neighbour cells. But depending on the number of influences and the map size, convolution can be computationally inefficient. In general, influence maps in games are seen to be task dependent and need to be tuned to give a benefit to the solution. However, influence maps offer a simple way to combine different parameters for tactical and strategic decisions and ease spatial analysis. More advanced versions of influence maps use value propagation or cellular automata, i.e. cells influence their neighbours based on a predefined propagation type. This propagation type defines a rule how a cell is updated, based on its neighbour cell values.

Influence maps allow an operator to provide additional information to the central planner in this project. The operator information is combined with other measurement types into a single utility value and influences the path planning (see Chapter 7 and Chapter 9).

2.8.2 Multi-robot collaboration and exploration strategies

Many problems in multi-robot collaboration arise from its distributed nature. But rather than exploring robot cooperation in detail, this project will use a simpler approach, because the robotic architecture uses a central computing unit. The grandmother robot will be the central computing unit and will allow the effective computation of tasks for the group of robots. This way, a highly sophisticated coordination technique for a fully distributed system is not required.

The aspect of robot collaboration has its own field of research and the reviewed papers show some methods addressing simplified coordination. Fox *et al.* [BMF+00] use a simple structured approach integrating a cost function for robots to explore dedicated target areas. This way, the costs can be optimized by evaluating all robots which are taking part in the exploration task.

Leung and Al-Jumaily [LAJ04] rely on a similar approach, although they use a bidding scheme in order to coordinate cooperative behaviour. Furthermore, a map building process can be supported by actively enforcing an overlapping search of areas for multiple robots, while estimating relative positions of those robots [FKK+06].

While both methods address fully distributed system, this thesis uses the given ideas in a system with a central computing unit. The details of this system are provided in Chapter 9.

2.9 Navigation

Different paradigms for navigation and collision avoidance have been successfully applied to robotics. In general, it can be placed on a spectrum from purely reactive strategies through to deliberative navigation schemes.

2.9.1 Behaviour based reactive methods

One of the main characteristics of purely reactive systems is the direct interpretation of sensor input. An intermediate knowledge representation is avoided, so that current sensor data provides the knowledge about the environment. Actuation is therefore solely based on current perception and does not involve any memory. Data processing is effectively done in the most direct way possible, showing the tight coupling of sensing and acting.

Behaviours form another important building block of a reactive navigation system. Mainly inspired by animal behaviour, a single behaviour is a simple rule of action, which is triggered under a certain condition or stimulus [Ark98]. The advantages of a behavior based system concern:

1. application
2. development

Primarily important is the application of behaviour, which allows action in dynamic and potentially hazardous environments without any global knowledge, due to fast response times. Secondly important, behaviour based systems ease development as they inherently support a modular design. A modular design facilitates reuse and maintenance, because functional blocks are easier to identify, allowing faster addition, redesign and discarding of individual behaviours.

However, overall behaviour is a combination of the different behaviours involved and as such emergent. Depending on the architecture, it has to be differentiated between primitive (low-level) and more complex (high-level) behaviours. Complex behaviours are often built from different primitive behaviours. Behaviours are often independent and concurrent [Mur00] and issues such as dominance and cancellation of behaviours have to be addressed.

Relying on the current local sensing only with avoidance of any intermediate representations is the pure behavioural approach. Hybrid solutions are common and this project adapts a hybrid solution previously developed in this department.

Subsumption architecture and potential fields

Reactive methods are commonly implemented in two different ways:

(i) Subsumption architecture [Bro86]

(ii) Potential field methods [Ark98]

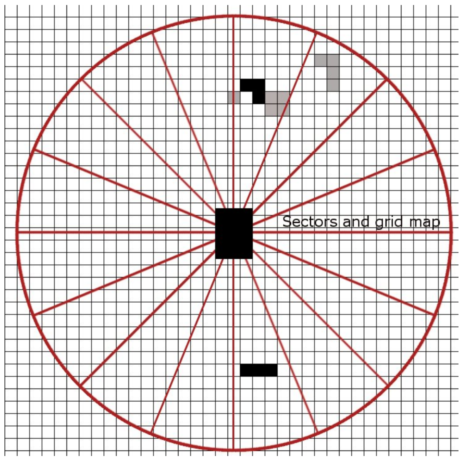
Within the subsumption architecture behaviours are structured in different layers, where each layer tries to complete a certain goal. Active behaviours can use inhibition of stimulus and responses to control other behaviours. Potential fields meanwhile employ vectors to represent the environment and behaviours with a magnitude and direction of force. These behaviour vectors are combined by summing, eventually leading to new behaviours. Comparing both approaches, they perform equivalently in practice and neither of these approaches is a completely robust solution [Mur00]. However, the subsumption architecture is closer to a hardware implementation allowing parallel processing of individual behaviours, while potential fields are closer to software implementation [Ark98]. Nevertheless, potential fields have limitations by becoming stuck in local minima or causing unstable motion in narrow passages [KB91].

2.9.2 A list of alternative navigation methods

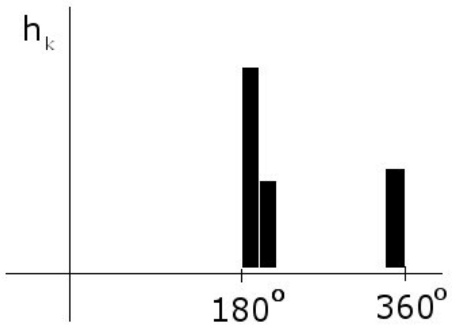
The following section provides a list of alternative navigation methods, which are part of the implementation in this project in one way or another. Navigation is based on a method developed in [LJC07]. The method combines different of the listed navigation methods, but this project also needs to extend a given navigation method into three dimensions. For that reason, this project looks at two of the few existing methods operating in three dimensional concise environments.

Vector field histogram

The Vector Field Histogram (VFH) [BK91] provides a mechanism for obstacle avoidance and goal finding at the same time. It will be one of the elements of the navigation algorithm for the mother robots. VFH uses knowledge about the current environment of the robot, where the current environment is defined by a moving section with the robot in its centre. An underlying occupancy grid allows the generation of a local occupancy grid around the robot's current position (Figure 2.2a). Based on the information contained in this local grid, a one dimensional polar histogram of



(a) Sectors in the grid map



(b) Polar histogram for densities

Figure 2.2: Building a polar histogram from sectors

the obstacle densities around the robot centre is generated in a next step (Figure 2.2b). This polar histogram has a predefined resolution resulting in sectors. An obstacle density value describes each sector. The obstacle density value is based on a combination of certainty values from the occupancy grid and a distance measure. Each cell provides a measure:

$$m_{i,j} = (c^*_{i,j})^2(a - bd_{i,j})$$

where

a, b	are positive constants
$c^*_{i,j}$	is the certainty of the cell i,j being occupied
$d_{i,j}$	is the distance to cell i,j
$m_{i,j}$	is the magnitude of obstacle influence on cell i,j

The obstacle density for a sector s is then calculated as follows:

$$h_k = \sum_{i,j} m_{i,j}$$

where

C_s	describes the set of cells in sector s
$m_{i,j}$	is computed from $c_{i,j} \epsilon C_s$

After smoothing the polar histogram to compensate for discreteness, it is analysed to find sections with a low obstacle density to finally process the direction of travel for the robot. To identify low obstacle density sections a threshold is used. One of the drawbacks of VFH is the need to fine-tune this threshold in challenging environments, e.g. those which are densely cluttered, otherwise it cannot pass through narrow passages.

The more advanced version VFH+ [UB98] reduces the tuning requirements of the initial approach and compensates for kinematic constraints by introducing further data reduction steps. Overall the algorithm consists of four main data reduction steps:

1. Generate a primary polar histogram.
2. Translate the primary into a binary polar histogram, where a sector can be either free or occupied.
3. Mask the histogram where the robot cannot travel due to kinematic constraints.
4. Select the steering direction based on a cost measure, which takes the chosen direction and the current position as inputs.

The translation of the primary into the binary polar histogram addresses the problem of oscillating in narrow pathways and relies on the use of two threshold parameters:

$$\begin{aligned} h^b_{k,t} &= 1 && \text{if } h_k > \tau_{high} \\ h^b_{k,t} &= 0 && \text{if } h_k > \tau_{low} \\ h^b_{k,t} &= h_{k,t-1} && \text{otherwise} \end{aligned}$$

where

$h^b_{k,t}$	binary density value, of section k at time t
τ_{high}, τ_{low}	upper/lower density threshold

The effects of this reduction step are a smoother and more reliable trajectory as claimed by the authors. Note, that the cost function of the last step facilitates the integration of behaviours by allowing the properties of this cost function to change.

In a comparative study with a fuzzy behavioural approach [SDC07] the authors have found that a vector field (or vector polar histogram) requires less programming and implementation effort than the behavioural method, which needs behaviours explicitly added to the system via individual rules. Nevertheless, with a laser range finder as a sensing device, the behavioural approach generated a smoother path and proved therefore better for energy saving and comfort of travel.

Nearness diagram

Nearness diagram navigation [MM04] is a recent approach to allow goal driven reactive navigation in cluttered and dense areas. It is one of the few approaches which addresses the operation in complex three dimensional environments. The approach builds upon a behavioural design. Minguez *et. al.* represent a set of situations in a decision tree and build it upon two main safety areas around the robot: high safety and low safety. Depending on the sensing of obstacles in each of these areas further criteria apply. Criterions are descriptions of specific situations and are differentiated in high safety and low safety situations, requiring specific actions to safely reach a goal. The approach claims to be general, but also provides a specific application based on sensor information provided as a depth point map generated by a ring of ultrasonic sensors.

The authors show that the idea of security situations and the use of a rule based system are capable of dealing with complex environments. Though this thesis uses reactive navigation, the presented techniques illustrate the possibility to extend it with a rule based system.

Obstacle avoidance in 3D workspaces

Motivated by robotic rescue in confined areas, [VM06] presents an obstacle avoidance methods for 3D by extending an existing validated approach from two to three dimensions. The general idea is to split three dimensional space around the robot into quadrants. These quadrants depend on the current orientation of the robot and the target location. After definition of the quadrants, each is analysed to find movement constraints, i.e. obstacles. The method uses a rule based system to find the single best single direction. The rules are based on a small predefined set of situations and the computation of direction is based on geometrical information only.

The authors show, that the method works in simulation, but it has not been tested in a real world scenario.

The current project requires the transfer of a two dimensional navigation method into 3D and also builds a geometric analysis for navigational control as described in Chapter 8.

Dynamic window approach

The presented examples so far, focus on the computation of a movement direction. Nevertheless, velocity control of is of significant importance to a operation as well. For that purpose this subsection looks at the well established dynamic window approach [FBT97]. The dynamic window approach uses the current robot velocity, i.e. linear velocity and rotational velocity, to compute a set of reachable velocities. The idea behind it, is that each system has a control loop and therefore a certain amount of time that a motion command can be applied. Depending on the maximum achievable acceleration/deceleration and the current velocity, the set of reachable velocities is limited. The formal constraints are described in following formulation of the dynamic window V_d:

$$V_d = (v, \omega) \mid v \epsilon [v_a - \dot{v}t, v_a + \dot{v}t] \vee \omega \epsilon [\omega_a - \dot{\omega}t, \omega_a + \dot{\omega}t]$$

where

v_a	actual linear velocity
ω_a	actual angular velocity
$\dot{v}, \dot{\omega}$	linear and angular accelerations

This set of reachable velocities is further analysed to find velocities for which a collision free trajectory can be predicted for the next control cycle. After this reduction step an objective function is applied to find the best velocity, in order to direct the robot to its goal at maximum speed, while avoiding any detected obstacles.

One of the main elements of this approach is that trajectories are always curvatures with a radius of:

$$R = \frac{v}{\omega}$$

where

v linear velocity

ω angular velocity and $\omega \neq 0$

Obstacles on the trajectory of a curvature define, whether the velocity is admissible or not.

Global dynamic window approach

As an extension to the dynamic window approach the global dynamic window approach includes a path planner called NF1 [BK99]. The planner computes the distance to a goal location for every single cell in the local environment. This information is then used in combination with the dynamic window to find the best direction and velocity. As soon a path planning algorithm is in place, it is a straight forward matter to embed it into the dynamic window approach. Thus, in order to enhance the navigation algorithm with existing map data, this project aims at integrating a (short range) path planning capability.

2.10 Simulation frameworks

The limited availability of robots often requires a valid simulation to be used instead during the development phase. Nevertheless, it would also require significant effort to test all development stages in a real environment. This is neither cost nor time effective, although it is well known that better and more realistic results can be expected from a test in real environment. However, a simulation should be able to produce valid preliminary results to overcome the first development stages. For that reason this project evaluated a number of different candidate simulation frameworks.

Most of the simulation frameworks come as an additional package for complete robotic control packages. Some of the currently available robotic frameworks which enable high level control are: CARMEN [MRH+02], Player/Stage [GHH+07], Webots [Cyb07] and Microsoft Robotics Toolkit (MRT) [Mic06]. CARMEN, Player and Webots provide frameworks with out of the box solutions for a number of robotic control problems, including path planning, localisation, mapping and communication. Each framework also supports a range of different hardware devices.

All these frameworks come with a simulation environment. In this environment a number of already simulated robots exist, e.g. the popular Pioneer [MOB07], which can directly be used in simulation. MRT offers a rather plain framework, i.e. no modules for common robotic tasks like path planning and map building are provided. Implementing solutions for task specific problems remain the developer's responsibility.

Most task definitions only require a 2D simulation environment. Operation in a simulated office space is one example. In this area CARMEN and Player/Stage compete. But a search and rescue operation requires robots to move in a more challenging environment, i.e. a heterogeneous, cluttered environment has to be expected and hence, is not comparable to any office space scenario.

For that reason, having a 3D simulation environment is essential. Such a 3D simulation environment including physics simulation is already part of Webots, MRT and is available as additional project for Player. In order to allow as much realism as possible, external physics packages are included in the given frameworks for that purpose, i.e. Ageia in case of MRT and Open Dynamics Engine (ODE) for Player/Stage and Webots.

A survey of commercial open source simulators [CMBG07] analysed a large range of simulators by four major criteria:

(a) physical fidelity

(b) functional fidelity

(c) ease of development

(d) cost

Physical fidelity defines how good the simulation can actually present the real operation environment and mainly concerns the presentation of rendering capabilities. Functional fidelity describes the degree in which physics are calculated considering forces such as gravity, motor acceleration and collision and how these affect the movement of a vehicle or parts of a vehicle. In order to take into account how much effort has to be spent to create a feasible operation environment and how easily modification can be made, ease of development is a further criteria. The last criteria is cost and it incorporates the time and money necessary for acquiring, installing and running the simulation.

Overall Player and Webots are considered to be in the midrange of all four criteria. Player is characterised with low functional fidelity. In contrast Craighead *et. al.* describes the physics engine Ageia as ″one of the highest fidelity engines available″. Therefore MRT shows high ratings for physical and functional fidelity. Only three available simulators provide a high level of fidelity, whereas MRT is the only one under consideration for this project. MRT is also a more resource friendly and well documented 3D simulation. Consequently, the current implementation is based on MRT.

Chapter 3

System overview

This chapter presents the layout of the centralised control system 'ExploreTM', which guides a team of robots through a search and rescue operation. This chapter provides a general introduction into individual components and is followed by a detailed discussion of the individual elements in subsequent chapters.

3.1 General design considerations and underlying assumptions

The control system applies to a hierarchical team of robots, that consists of a single grandmother robot and multiple mother robots. The system centralises high level control on the grandmother robot to give guidance to the exploring mother robots.

The centralisation of control has several advantages over a fully distributed system. Firstly, compared to a fully distributed system less communication effort has to be spent to organise the group of robots and achieve decisions, which affect the group as a whole. Secondly, computational intensive tasks can also be delegated from other robots to the grandmother robot, thereby increasing the capacity of the individual exploring units.

Clearly though, the grandmother robot is a single point of failure in the present architecture. Therefore, in order to reduce the overall risk of physical breakdown, the robot will be positioned almost stationary but close to the operation area. The chosen location has to guarantee communication with all mother robots in the exploration area during the operation (see Figure 3.1).

In order to test the system in a simulation environment two major assumptions are made, based on the given constraints:

(1) The grandmother robot does not penetrate the field of rubble for search or rescue and therefore does not need a physical simulation.

(2) The grandmother and the mother robots communicate within a wireless network with full coverage of the operation area, and communication between grandmother and mother robots is assumed to be perfect[1].

An additional task of the grandmother robot can be the relaying of messages between mother robots. However, this is not an element of the current design. Instead, this project expects more sophisticated wireless solutions with better hardware and intelligent relaying strategies in the future, so that transparent and continuous communication can be achieved in a real scenario.

The grandmother robot is the general high level manager of the team of robots in the architecture and it collects data from mother robots to direct them to new exploration areas. The mother robots are part of the active search group and explore the operation area autonomously, but based on the general directives provided by the grandmother robot — referred to as 'missions'.

3.2 Centralised control to command a team of explorers

To control the team of mother robots from a central device the following major requirements have to be fulfilled:

- The mother robots need fundamental navigational control to operate autonomously.

[1]The actual simulation of multiple robots will run on a single computer.

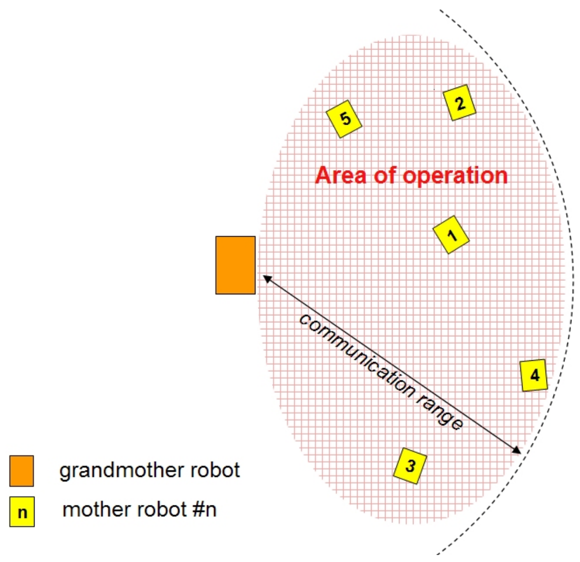


Figure 3.1: Assumptions for the operation

- The grandmother robot needs to compute a global exploration strategy for the group of mother robots with individual exploration tasks for each mother robot.
- Each mother robot has to execute the assigned exploration task autonomously based on a possibly individual local exploration strategy

While the grandmother robot is responsible for computing the global exploration strategy, mother robots will have a local exploration strategy which is inherent to the navigational control. The navigational control is based on Nehmzow's description of robot navigation [Neh03] and the ideas of Bekey [BT90] with the following low level control blocks:

- self-localisation
- map building
- path planning

Path planning in this description accounts also for a local (reactive) navigation scheme.

This project does not implement the self-localisation module (refer Chapter 5) for the control architecture (see Figure 3.2), but focusses on map building, path planning and navigation and finally applies an exploration strategy.

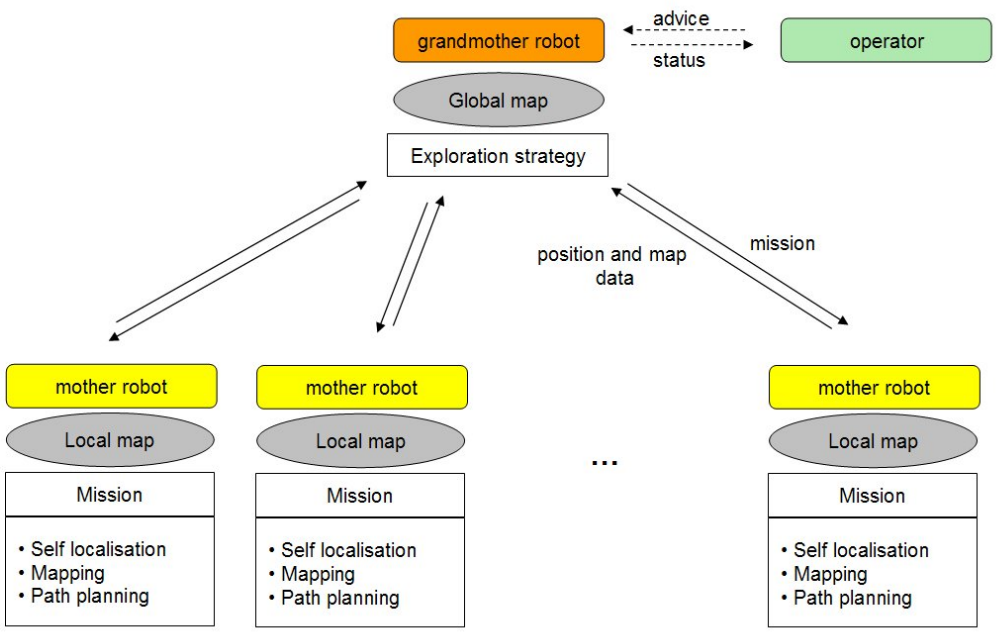


Figure 3.2: Control architecture

3.2.1 Tasks of the grandmother robot

The grandmother robot itself does not penetrate into the operation area and thus, does not have a perceptual unit. It therefore serves primarily as a data integration platform and maintains a global map of the environment by integrating data from the mother robots. It does so, by tracking the mother robots' positions and states.

The combination of global map and knowledge about the current mother robot positions enables the grandmother robot to compute an optimised set of missions for the group of mother robots. Each mission defines a target area for a mother robot to explore and the grandmother robot also provides the currently known best path to reach this area. This path is computed on the basis of the accumulated data and thus, can only be a rather loose guidance for the mother robot — taking into account that the environment is highly unpredictable and gives higher importance to current or recent perception than to accumulated and possibly outdated knowledge. This also incorporates inaccurate localisation, a (low) resolution based map and a changing, noisy environment; factors which all affect the validity of the long term map representation (refer Chapter 6).

3.2.2 Tasks of the mother robots

The main task of the mother robots is exploration and mapping of their environment. Therefore perception and subsequent map building are of special importance for the mother robots (Chapter 6).

The mother robots mainly use a reactive navigation scheme to allow for their operation in cluttered environments. To take advantage of available map data and planning capabilities, each mother robot can also plan a path through its local environment. Path planning and the integration into the navigation module are presented in Chapter 7 and 8.

The global exploration strategy for the group of mother robots is defined by missions received from the grandmother robots. These missions are provided by the grandmother robot usually on request — only the very first mission will be an unrequested assignment. A mission defines an overall area to explore (currently a circular region), but to explore this area mother robots apply an individual strategy; the frontier-based exploration will be used in this project (refer Chapters 2 and 9)

3.3 Embedding the control system into a simulation

Due to a number of reasons, which are outlined in Chapter 4, this project requires a simulation of the mobile robots. The simulation and its integration with the control elements of the system are an essential part of this project, which allows the evaluation of ExploreTM. The relationship between control and simulation is illustrated in Figure 3.3.

The simulation is based on two major functional elements: physical simulation of the mother robot and simulation of the current low level software stack[2]. Each mother robot needs to be simulated with a device model — instances are illustrated as 'mr device <id>' in Figure 3.3. To simulate the low level software stack, this project reconstructs a reduced

[2]This term refers to the layer of software that directly connects to the robot hardware.

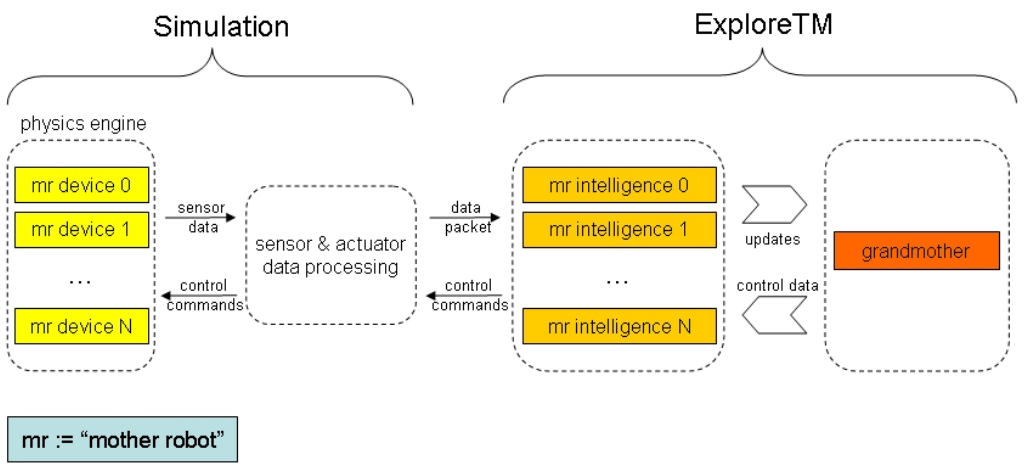


Figure 3.3: Relationship between control and simulation

version of the software stack of the actual mother robot — protocol specific communication details are ignored. This actual software stack operates on a layer close to the hardware and combines all raw sensor data into a single packet and also allows the mother robot to perform incoming control commands. The 'sensor & actuator data processing' element as shown in Figure 3.3 simulates this component and also defines the communication interface for ExploreTM, which is a close match to the real interface.

3.3.1 Simulation as part of the system design

MRT provides the simulation environment in this project (refer Chapter 2) and allows the substitution of simulated devices for real devices. It also allows modularisation using loosely coupled modules and thus, communication between modules is message based. Figure 3.4 separates elements into simulation and control and gives an insight into the actual implementation, where services and the MRT specific physical simulation are separate. This project uses services to implement generic robot control, but also to enhance the simulation. Services that are a part of the control of the mother robots combine the functionality of the developed control algorithms in this project, and in addition provide communication and synchronisation mechanisms (refer Chapter 4) — each of the illustrated rectangular elements represents a service instance.

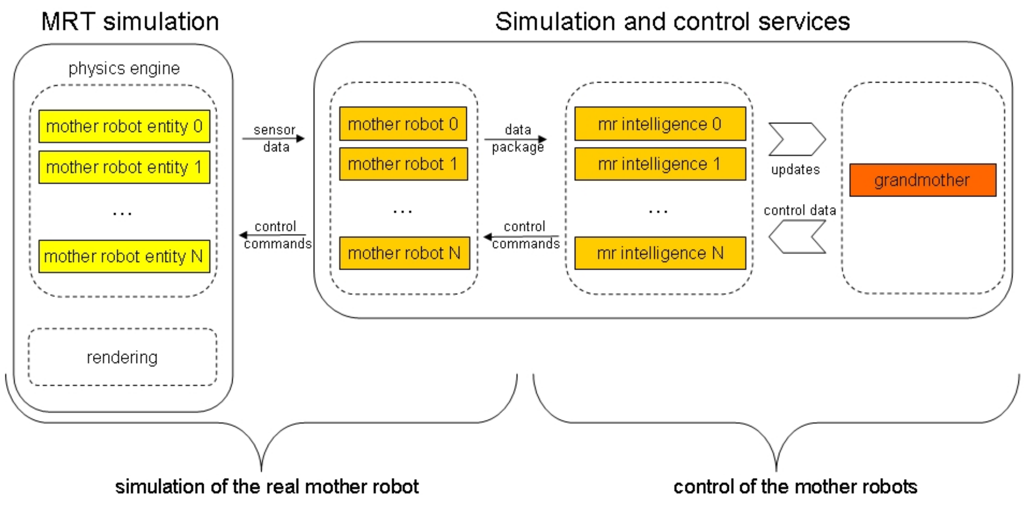


Figure 3.4: Simulation and control of the robots

Each mother robot is represented by a physical entity in a physics engine within the simulation. In order to fully model the mother robot as developed in [Wil07], an additional 'mother robot <id>' service performs the mentioned 'sensor & actuator data processing' (see Figure 3.3), handles simulation specific details and communicates with the simulated mother robot entity. Thus, this service mediates between the control services and the simulated entity, and finally allows the control services to work independently from the robot simulation. The control and its implementation are therefore generalisable — only the message types passed between the control and hardware devices need to match for a transfer to the real device,.

In order to ease this transfer, the implementation follows closely the existing hardware communication protocol definition [Wil07]. This protocol mainly defines the sensor data the mother robot provides. Nevertheless, the current implementation contains (temporary) extensions, where necessary; changes are required for additional package content after introducing new sensors or in order to forward debug information from the simulation environment.

The choice of a robotic framework requires some commitments in terms of the technologies to use. The general intent of a robotic framework is to ease the developer's life and MRT does so. Nevertheless, it has to be clear that the usage of such a framework also raises constraints and enforces the application of specific techniques. For this implementation it has implications on the communication and the way of dealing with concurrency, which are not easily transferable to other platforms (operation systems apart from Microsoft Windows). These implications are discussed in Chapter 4.

However, the framework allows a faster implementation in general and matches the design approach. The service oriented nature of the MRT framework supports the idea of a centralised operating grandmother robot, which can be seen as a service provider for the group of mother robots. Eventually, the design of an effective and clear communication is facilitated.

3.4 Message based communication

Although control is centralised, the team of robots is still distributed and thus relies on the exchange of messages to transmit information. Three main communication nodes exist:

1. an operator
2. the grandmother robot
3. the mother robot

Future designs will need to expand this model to include the daughter robots as further communication nodes.

Figure 3.5 visualises the message types including their direction of flow. The diagram abstracts the most important communication elements, while the implementation actually splits some message types into more specialised types.

- *influence map*: allows operators to influence and monitor operations
- *state update*: provides the known position and orientation of the mother robots, based on the best knowledge of the grandmother robot

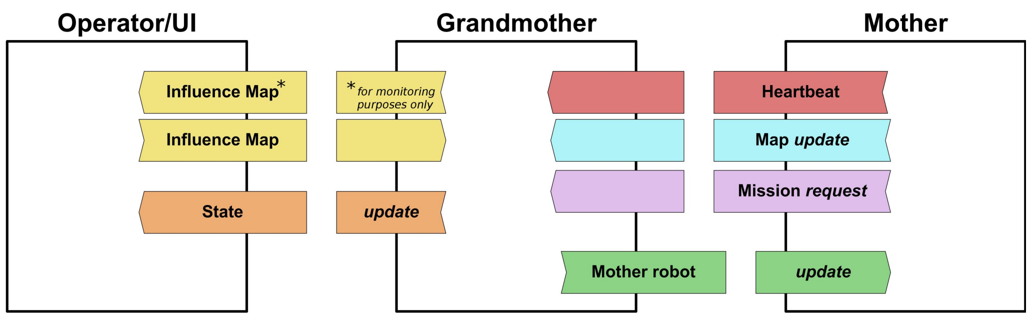


Figure 3.5: Message types

- *heartbeat*: contains information about the current position and orientation of the mother robots
- *map update*: transfers an update (subsection) of a larger map
- *mission request*: triggers a mission update for the group of robot. This is a functional message type and contains no payload.
- *mother robot update*: transfers the current mission to the individual mother robot

The focus of ExploreTM is autonomous control, so that an operator should be mainly observing. Messaging between operator and grandmother robot is therefore limited. The grandmother also serves as a buffer and relay to direct information from the mother robots to the operator, because direct communication between mother robots and the operator cannot be guaranteed. The *State update* message, for example, is forwarded to the operator and contains information about the current positions of mother robots in the field.

Operators can provide information via an influence map, which is only sent to the grandmother robot. On the grandmother robot, this influence map is then augmented with map information from the mother robots (refer Chapter 7 and 9 for details). The operator will regularly receive an enriched map and more frequently a general state update from the grandmother robot to monitor progress of the operation.

The outlined communication between the grandmother robot and mother robots characterises the grandmother robot's initial more passive, observing role. Nonetheless, it provides additional functionality to the mother robots. So the mother robots will actively send timestamped heartbeat messages, containing information such as their current position and orientation. The grandmother, however, will send updates to the mother robots usually only on request. A single request of one mother robot will trigger a mission update for the whole group. This is necessary in order to maintain an optimised global strategy. Influenced by the current mother robot positions and the current exploration status of specific areas it is more efficient to send all robots their currently best missions, than just sending the one robot to its best mission.

At a more detailed level, the map updates are separated from heartbeat messages so that they can be less frequently sent. The reason for this separation lies in higher preparation cost of the map updates.

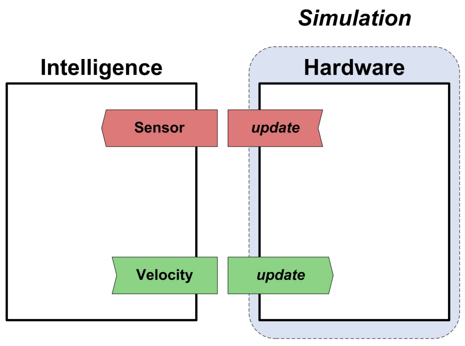


Figure 3.6: Communication of the mother robot control with the (simulated) hardware

In order to communicate with the hardware, a mother robot requires only two different messages types (see Figure 3.6). The mother robot (intelligence service) reasons with the sensor updates received from the hardware and generates a velocity command for the mother robot drive (see Figure 3.7).

Not only communication between robots builds on messages, but also the robot internal communication. Using messages for internal calls instead of direct methods or function calls is mainly done to avoid issues with multiple synchronisation technologies (see also Chapter 4).

An understanding of this message based approach is also essential to follow causalities and timing. The communication between hardware (simulation) and mother robot intelligence serves as an example:

```
typedef struct _AT91S_UPD_DataIn {
        unsigned short RT_ID; /* heartbeat */
        unsigned char master; /* Master status */
        unsigned char supply; /* PSU control */
        unsigned char R_motors; /* Motor A value */
        unsigned char L_motors; /* Motor B value */
        unsigned char camAng; /* Camera servo angle */
} AT91S_UDP_DataIn, *AT91PS_UDP_DataIn;
```

Figure 3.7: Real incoming UDP data package from [Wil07] for high level hardware control

The hardware sends state updates to the intelligence service at a frequency of 5 Hz. A message handler within the intelligence service will trigger subsequent actions once the update is received and effectively all following actions are bound to the sensor update frequency. In order to achieve better computational efficiency[3] specific actions are only called after a number of sensor updates. Thus, the sensor update frequency is the fundamental timing parameter the control is built upon.

3.5 A time-based analysis of ExploreTM

To allow a better understanding of the internal control structure Figure 3.8 illustrates a high level sequence diagram. For reasons of clarity the diagram abstracts some details, which will be discussed in this section.

Every time the mother robot's intelligence service receives a sensor update, it generates an internal map and control request. This request triggers the following actions:

1. A point cloud (or set of vertices) of the location environment is computed from current sensor data
2. The point cloud is integrated into the map of the mother robot
3. The graph structure of the map, which is used for path planning, is updated

[3]Efficiency is essential for the simulation in this project to allow multi robot scenarios.

4. A velocity command is computed taking into account current and previous sensor data

This process guarantees that the mother robot operates with the most recent sensor input for navigation.

In order to keep the grandmother robot informed about the position of the exploring unit, each mother robot sends heartbeat messages to the grandmother robot — with a basic sensor update frequency of 5 Hz the current setting generates updates at every two seconds.

Because map updates are computationally costly and in a real scenario affect communication bandwidth, mother robots generate map updates for the grandmother robot at a lower frequency (currently every 15 s). In addition map updates are actually partial map updates, because a mother robot considers only the area that might have changed during past the past travel. The grandmother robot will update a graph structure from its global map after receiving a certain number of map updates — the graph update is usually triggered when every mother robot has sent a map update.

When a single mother robot requests a mission, the grandmother generates an optimised mission set for all mother robots and updates the states of the mother robots — the state includes the current mission.

The operator influences exploration by sending information contained in an influence map to the grandmother robot. This additional information is merged with the grandmother robot's current influence map which is based on the global map data. The grandmother robot sends state updates of the group of robot and its influence map at regular intervals back to the operator interface to allow for continuous monitoring.

Tuning parameters of the control system

The sensor update frequency is the fundamental timing variable and other timing parameters are set relative to it:

- sensor update frequency
- heartbeat rate
- map update rate

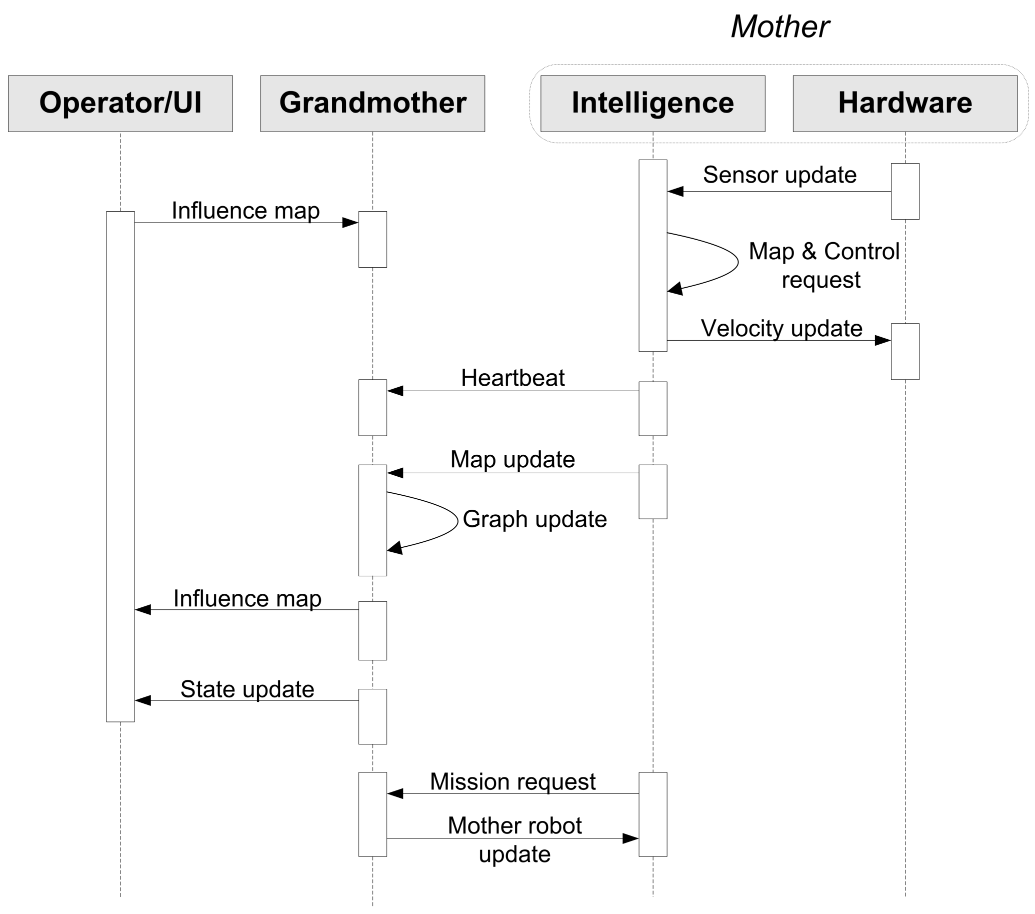


Figure 3.8: Sequence diagram of the control

The graph rebuild rate can also be defined for the grandmother robot.

In addition to the timing parameters, the message size of the map update can be influenced by setting a limiting radius around the robot's current position, so that only a map update of that specified section around the robot is forwarded. But the radius and the map update frequency are not independent, considering that the grandmother robot should receive updates which cover all explored regions. For that reason, the following constraint should be satisfied:

$$\frac{m}{f_s} v_{max} + r_s \leq R$$

where

f_s	sensor update frequency in $\frac{1}{s}$
m	map update rate (dimensionless factor)
v_{max}	maximum velocity of the robot in $\frac{m}{s}$
R	map radius in m
r_s	the robot's sensing radius in m

This constraint considers the maximum travelled distance plus the sensing distance and therefore ensures that a map update includes all recent updates of a travelled region. The current implementation does not enforce this constraint, but an alternative implementation could monitor this constraint and trigger an update when it is violated.

3.6 Summary

This chapter presented the overall system design based on a centralised control built around the grandmother robot. It introduced the individual elements of control and differentiated between the requirement for the individual mother robot and the group as a whole. The main and defining elements of the architecture are:

- participating communication partners: operator, grandmother robot and mother robots
- variety of message types: as trigger only or with payload

- timing structure: relative to the sensor update frequency

Chapter 4

Simulation

Development in the area of mobile robotics simulations is often supported by a simulation, because it is significantly cheaper and allows much faster and more flexible experimentation than using physical robots. For this particular project, a simulation was essential because there is only one physical mother robot at present and it is only partially functional, so it could not be used for real experiments. Thus, one goal of building the simulation was to have a simulation available for future projects.

MATLAB [Mat08] has been the simulator of choice for recent projects. In these cases simulation was 2D and robots ran only in planar environments such as office spaces. This is not adequate for search and rescue environments which are cluttered, non planar, and contain not only walls and obstacles, but also holes and drops — a 3D simulation is required. Such a simulation can also be used for other two dimensional simulations, although only with a higher computational overhead.

A useful simulation must satisfy a range of constraints, concerning accuracy and representing 3D, due to the nature of the application. The robots will negotiate a field of rubble and interact closely with the environment. That requires an accurate simulation to imitate physical behaviour of objects. In general, the requirements to implement a realistic environment can be large considering the detailed simulation of forces. Consequently, setting up a realistic simulation environment is a demanding task.

MRT was chosen for this project from a selection of toolkits for robotic control (see Section 2.10).

4.1 A description of the simulation framework

MRT has a mature 3D simulation, which was the main reason to choose it over all other frameworks. For simulation purposes, MRT interfaces with a third party physics engine [AGE07], which is one of the most sophisticated in the industry [CMBG07]. The physics engine allows collision detection and computes realistic collision behaviour. Furthermore, customised robots can be modelled with a high degree of accuracy, e.g. parameters such as the wheel friction, suspension and mass can be set. Such parameters can be experimental values derived from the real world, thus creating a simulated world close to the real one. These aspects are important prerequisites for simulating robot movement in cluttered environments.

In general, the framework is highly adaptable and enables the creation of a realistic approximation of a real world robot. But because MRT is just a framework, some effort is required to design services, message types and the individual robot models (also called entities) including their specific sensor devices.

4.1.1 A special technology addresses concurrency issues

Since an exploration team consists of multiple robots, concurrency issues for the control model and the simulation arise. MRT assists using a technology called Concurrency and Coordination Runtime (CCR) [Mic07] and allows the abstraction of concurrency issues. This technology eases concurrency handling, but does not free a developer from a sensible design to avoid performance bottlenecks. As one of its features, CCR allows the exclusive execution of a program section. The consequence of an uninformed use of this feature can be fatal, wasting a huge amount of computing capacity. In addition, a mixture of CCR and classic synchronization techniques such as monitors or locks should be avoided. However, this is not always possible, so that special care has to be taken when mixing the models.

4.1.2 Communication infrastructure

The communication infrastructure of MRT is built around a service based model. The main idea behind this model is the separation of functionality and encapsulation into single units called services. Each such service can handle inbound and outbound communication through so-called ports and associated message handlers.

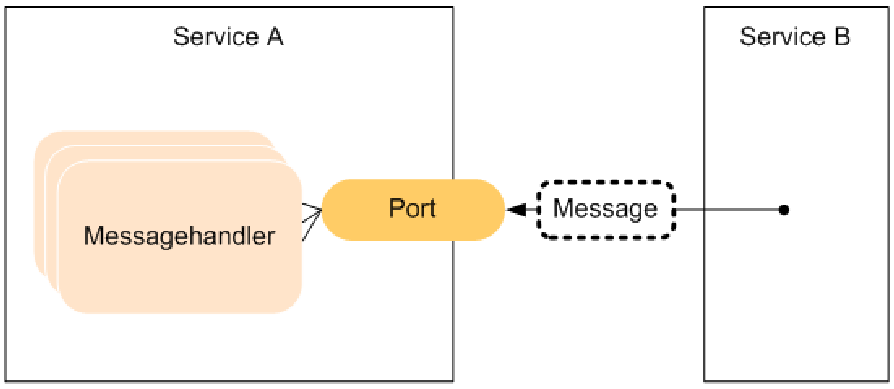


Figure 4.1: Communication with port, messages and message handlers

A service supports specific message types which can be received on a single port. A message can be forwarded to a message handler, which has been attached to the port. Once the port receives a message it forwards it to the message handler, which specifically deals this type of message. Messages are the essential elements of this communication between services — they can contain payload data or just trigger some action, so that a variety of message types can be defined.

A loosely coupled system is the main result of communicating only via messages. Hence, components (services) can be substituted as long as they satisfy the interface requirements of the service in terms of sent and received message types.

For the robot simulation, a number of custom messages define the communication model between grandmother and mother robot, as shown in Chapter 3. For example, in order to send a partial map update from the mother robot to the grandmother robot, a special message type carries map data. While the abstract design of the messages is mainly generic, the implementation has to be specific to MRT. This approach is structured and allows transparent handling of low level communication issues, which would arise with alternatives such as a purely socket based approach.

4.1.3 Modelling devices and entities

MRT provides a small standard set of modelled robots, which can be used in simulation. In addition, it provides a general implementation of a differential drive plus several simulated sensors such as a webcam and a laser range finder. Due to the unique design of the mother robot and specific sensor units, this project could only take advantage of few of them and had to build customised models for the others.

In order to simulate a device, two main actions have to be taken:

1. creating a three dimensional representation of the device/entity
2. reproducing inherent functionality of a device

Creating an initial representation of elements like wheels, joints and rigid body elements is straight forward and can be defined based on the specifications of already existing devices. Nevertheless, further customisation is essential, e.g. for a wheeled robot, the detailed representation of the wheel have a significant impact on the quality of the simulation.

In MRT (as a wrapper around the physics engine), wheels can be parameterized with a so-called tire force function. The tire function as shown in Figure 4.2 visualises four out of five overall parameters:

e	Extremum slip
e^*	Extremum value
a	Asymptote slip
a^*	Asymptote value

The function defines how much force a tire can transfer, until the tire starts to slip. A further associated parameter is 'stiffness' and it describes how much weight a tire can carry. The tire force function can be applied independently for forward and sideways forces acting on a tire.

Representing the functionality of sensors of differential drives is complex and needs close interaction with the physics engine. To generate realistic output from the range finder, for example, the physics engine provides the infrastructure to calculate the distances between devices and objects.

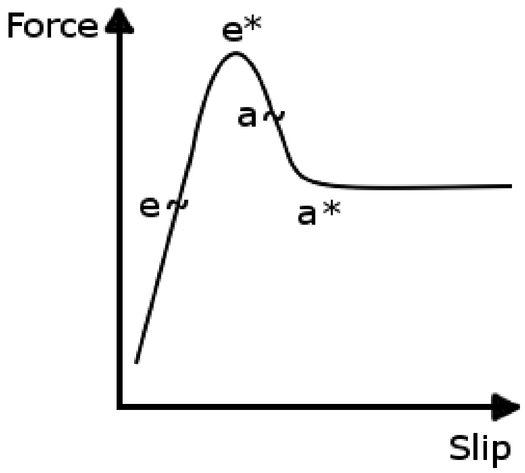


Figure 4.2: Tire force function

Although the real robot require specific elements to handle the communication between sensors and the robot, in the simulation, the details of device specific communication can be ignored and the main focus put on the actual functionality. A service can represent a sensing device by encapsulating the details of (serial) communication along with the sensing device. As already mentioned, this simplifies the transition from real world to simulation and vice versa, because the service simulating the sensor can be used interchangeably with the service for the real device.

4.1.4 Multi robot simulation

In general, simulating a group of robots is not much different from simulating a single robot — once the model for a single robot is established, the model can be reused. However, each robot needs to be uniquely addressable in order to be controlled from the grandmother robot.

Furthermore, the simulation and control of each individual mother robot is based on a group of interdependent services. In order to automate the activation process of services in this simulation, a numeric identifier allows for a consistent naming scheme: 'motherrobot-¡id¿-¡modulename¿'.

4.1.5 Special considerations for using a 3D simulation

After building a simulation for a 3D environment, it is up to the control elements to account for the third dimension in mapping, planning and navigation. The additional dimension compared to 2D, however, has an impact on the computational requirements. The simulation has to compute physics for all dynamic devices and objects in the environment. This challenge is even higher in a cluttered environment, because of the large number of individual objects.

In this project, multiple robots operate simultaneously. But clearly, the simulation of a group of robots requires significantly more resources than the simulation of a single robot, considering that each robot performs path planning, mapping and navigation individually. A dedicated physics card can ease this challenge, but was not available in this project.

The control algorithm requires additional resources, so that a scenario with two concurrently operating robots can require up to 80 per cent of the processing capacity of a 3 GHz Dual Core PC with 2GB of memory, depending on the tuning parameters of the control. For that reason, some effort of this project aims at reducing the performance requirements of simulation and control algorithms.

4.2 The modelled world and robot devices

This section describes the three major elements developed for the simulation:

(i) **world model**: the operation environment of the robot

(ii) **robot model**: the representation of the robot itself, including sensors, communication and part of the software stack

(iii) **configuration interface**: a interface to facilitate configuration of the simulation and allow quick setup of tests

The control software for the robots is not considered part of the simulation environment: the control software is the central element which is tested in the simulation and as such not part of the simulation itself.

Therefore, the term 'simulation environment' only refers to the modelled environment and the hardware parts including their low level communication.

4.2.1 The world as a field of rubble and debris

The application of USAR robots target mainly sites with structural collapses which result in a field of rubble and debris. Based on an analysis of a variety of disaster site images, this project identified the following set of unique challenges within a site:

- overcoming a field of rubble and specifically dealing with major drops
- moving on an inclined surface (on different grounds)
- detecting enclosures

The challenge of detecting enclosures arises from the idea, that an autonomously operating robot should be able to signal its own emergency status to an operator. This would allow the operator to take adequate action such as temporarily controlling the robot manually. These challenges also provide a basis for measuring performance according to the published performance guide for Urban Search and Rescue robots by the US Department of Homeland Security [Dep08]. In order to validate the functionality of the robot and its algorithms, the robots must be confronted with a number of unique challenges, i.e. dealing with one of the above challenges at a time, as well as combined challenges dealing with a overall mixture of these challenges.

Creating challenges

To create an environment full of rubble the simulation places a number of variable sized blocks over a predefined area. A random process with an underlying Gaussian distribution controls height, width and length parameters of the blocks, so that a variety of blocks is produced. The same random process determines positions for the blocks in a predefined area. By placing the blocks initially well above ground, they fall onto each other creating a random environment, which includes unstable parts and holes (which will be important for subsequent projects introducing smaller mobile robot units).

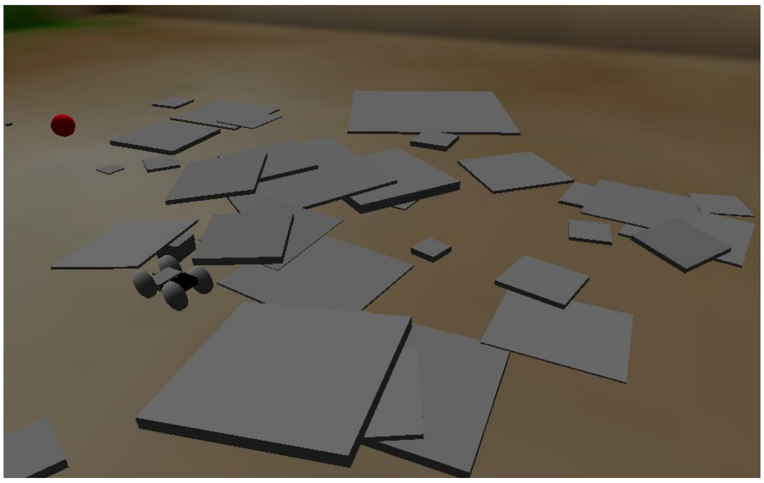
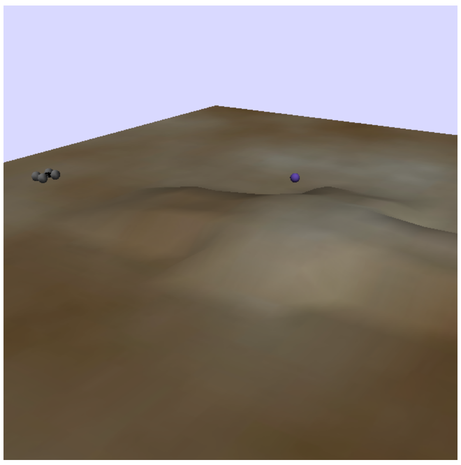


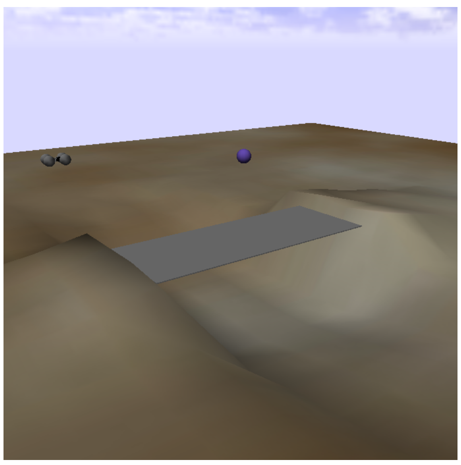
Figure 4.3: Field of rubble

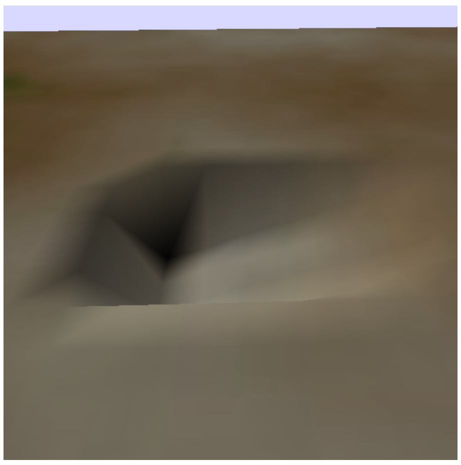
In order to generate other challenges like enclosures or drops, MRT also allows the creation of surfaces using an elevation map instead of just dropping blocks. This allows fast generation of varying surfaces with particular desired properties.

Elevation maps cannot be used to build structures such as bridges (part of the drop challenge) which need further description of free space sections. Generation of such challenges requires the manual, i.e. programmatic insertion of additional objects or structures defined in an external 3D designer tool. Figure 4.4 presents four different challenges, each addressing one of the mentioned challenges.

While this thesis was being written, Microsoft published a preview of its new version of the robotics framework. This new version also integrates a feature which allows the fast creation of building structures. Combined with the presented approach of creating a field of rubble, this should be able to create an even more realistic search environment and more complex challenges.


(a) Ramp


(b) Bridge


(c) Drop

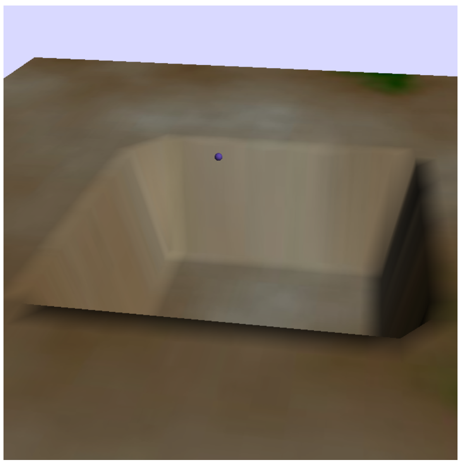

(d) Enclosure

Figure 4.4: Four different challenges

4.2.2 Modelling the mother robot

The mother robot consists of different elements which need simulation:

- a rigid body
- a single joint
- wheels
- sensors
- low level/control software

This project models the mother robot as closely as possible. The real mother robot is described in [Wil07]. Table 4.1 lists the key features of the specification of the real mother robot:

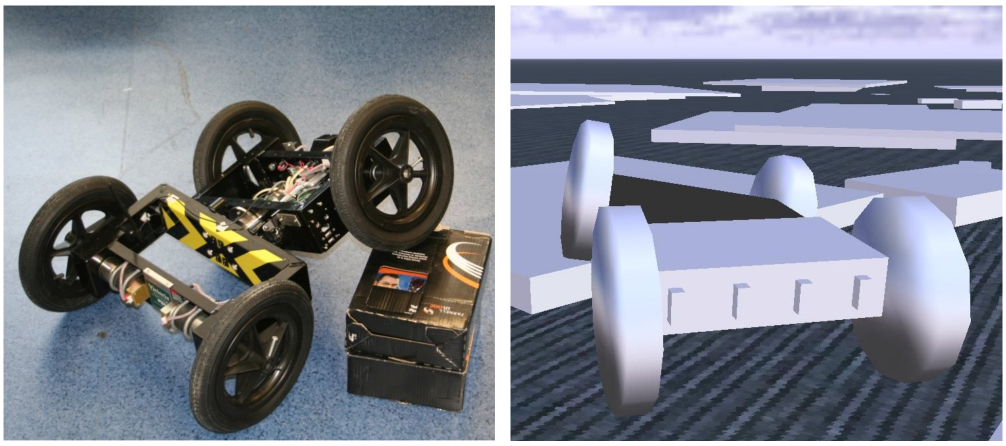
Weight (front/rear)	7.4/6.4 kg
Bodysize (front = rear)	$25 \times 40 \times 10$ cm
Wheel radius	15 cm
Max velocity	0.35-0.5 $\frac{m}{s}$ depending on battery load
Max acceleration	0.4 $\frac{m}{s^2}$

Table 4.1: Specification of real device

Implications of the mother robot design

A major feature of the mother robot is the split chassis design. The front and the rear body are only connected by a cylindrical joint with one degree of freedom (see Figure 4.5). This design allows the front and rear body to twist, with the intention to facilitate the negotiation of obstacles. The degree of freedom for the joint is limited by four (physical) stopper elements.

Compared to a robot with only a single rigid body, the specific design of the mother robot increases the complexity for the simulation and control in two ways. Firstly, the project had to create a customised model which could take little advantage of the design of other already modelled robots.



(a) real device (b) simulated device

Figure 4.5: Mother robot

Furthermore, the placement of sensors has to be more precisely specific, because sensors can be either attached to the front or the rear body. Secondly, because the control had to take into account the different orientations of front and rear body.

In addition to these challenges, the robot's symmetrical design allows operation after it has been flipped over. Because of the way that the physics engine models wheels[1], this feature cannot be directly supported. Therefore a specially designed monitor function detects a flip and triggers appropriate changes to allow for continuous operation.

The ability to flip is a feature which addresses endurance and tumble recovery as described in Section 2.1.3. To fully guarantee continuous operation after a flip (upside down), navigation and mapping algorithms require knowledge about the current orientation.

Range finder simulation

The majority of sensing devices of the mother robots are range finders [Wil07], which use single ray sensing. These are infrared range finding devices which come in two different forms: (a) single and (b) array. The

[1]In MRT's physics engine Ageia, wheels are modelled as single vectors, sliding on the ground. To cope with complex terrain, the model has to use eight superimposed 'wheels' with different orientations

infrared sensor arrays of the mother robot provide five readings, where all these readings lie in the same plane and have a spanning angle of 25°. The single sensor provides one reading. Both device types have a maximum sensing range and a minimum range — the first 20 cm in the case of the 1.2 m maximum range cannot be measured correctly, or 40 cm with the 3 m maximum range. False sensor readings can be caused by obstacles in the minimum range and can have a negative influence on the performance of the robot. However, for this project mainly a blind section with no measurements is assumed (refer Chapter 6).

Table 4.2 and Figure 4.6 illustrates the sensor configuration of the current hardware (the rear sensors are missing in this illustration, but were part of the initial design).

Front	2×single 1.2 m 2×array 3 m
Side	2×single 1.2 m
Rear	2×single 3 m

Table 4.2: Sensor configuration in original mother robot design

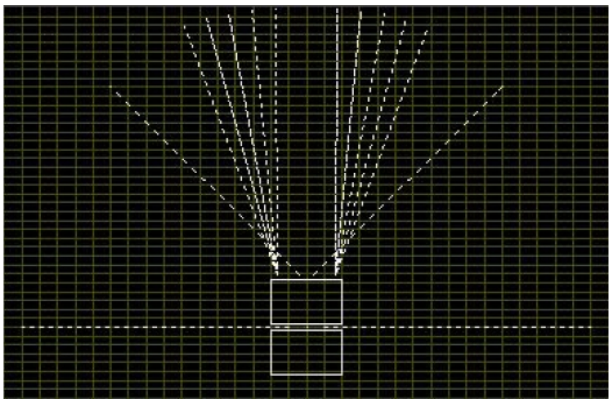


Figure 4.6: Sensor configuration from [Wil07]

Infrared range finders are similar to laser range finders, because both measure distance with a single sensing ray. However, the infrared sensors provide fewer readings and also less accurate ones than a laser range finder.

This project modified an existing module which simulates a laser range finder to model the infrared sensors. The modifications to the laser range finder module include resolution, ranging distance and opening angle, which were derived from the data sheet of the real infrared sensor devices. To simulate the inaccuracy of the infrared sensing devices, it is possible to activate an optional noisy sensor model, which is based on the mixed distribution model discussed in Chapter 2.

The simulation simplifies localisation

The simulation can provide information about the global position and attitude of entities at every time step. Therefore an explicit model of localisation sensors, i.e. a Global Positioning System (GPS) sensor or similar, is unnecessary. The knowledge about position and orientation after every control interval allows the modelling of an Inertial Measurement System (IMS), which includes accelerometers and gyroscopes. However, the localisation techniques are in place to provide exactly this: information about position and orientation. So it is not a matter of modelling these units, but to estimate the error they produce and Chapter 5 offers a broader presentation of this issue.

Noise will always affect localisation and as an example of how to model noise, this project integrates odometry error into the simulation. The odometry error will not be relevant for the design of ExploreTM, but only serves as example on how to achieve a more realistic sensor simulation.

The main sensors for odometry data are wheel encoders, which provide the distance of travel for a wheel. They operate with a certain resolution and thus integrate a small error over time. Based on the knowledge of travelled distance, the current position can be inferred from a previously known positions. The model as described in 5.1.1 initially provides exact localisation, but it also can incorporate an error, controlled by a single parameter per wheel based on experimental values of the real device. This parameter specifies the error as a fraction of distance travelled. When noise is activated in the simulation, this parameter is a sample from a Gaussian distribution with a mean of 0.6% and a standard deviation of 0.2%.

4.2.3 Central configuration for simulation and control

Using MRT has a steep learning curve due to the involvement of various technologies. To make it easier for other users of the system, I introduced a central configuration file to allow users to choose an environment, place the robots in it and configure the specific control parameter of ExploreTM without having to learn all the details of MRT.

The central configuration file can be modified to create different configurations and scenarios for multiple robots. Hence, it allows an easier setup method for testing and enables novices also to use the simulation. Changeable parameters of the configuration correspond to one of the following categories:

(a) robot

(b) terrain

(c) control

(d) physics

(a) Within the robot category, initial positions and missions are set, so that the number of robots and their starting positions can be controlled. In addition, the definition of initial missions allows the test of reaching specific areas.

(b) Terrain settings allow the selection of a specific challenge the robot has to face. The field of rubble is generated randomly, while all other challenges are predefined.

(c) The control section offers the most complex setup. It contains all the tuning parameters outlined in Section 3.5 and the relationship between map update, sensor frequency and subsequent tasks is defined here.

(d) The physics section addresses limitations of the simulation engine, which require a specific setup. This section allows also the activation of noise for range finders based on the references given in Chapter 2 (see

Figure 4.7) and, as mentioned, for odometry. Note, that in this project no algorithms are in place to filter noise.

These settings are all combined in a single XML formatted file and Figure 4.8 shows an example.

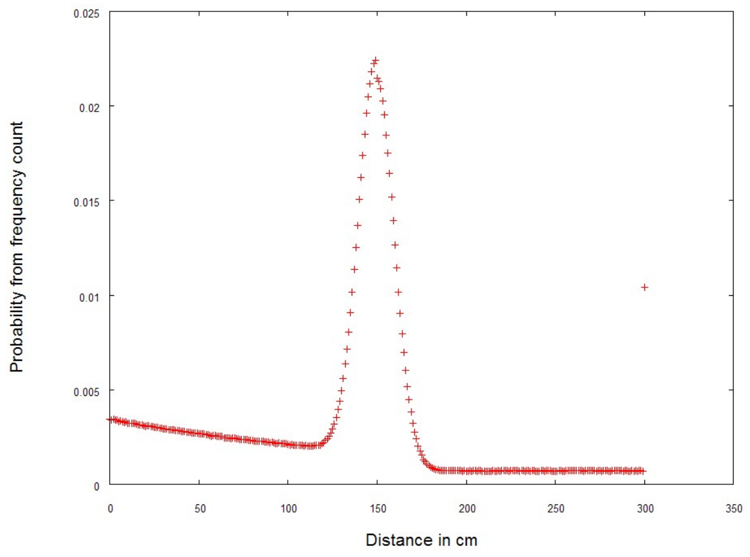
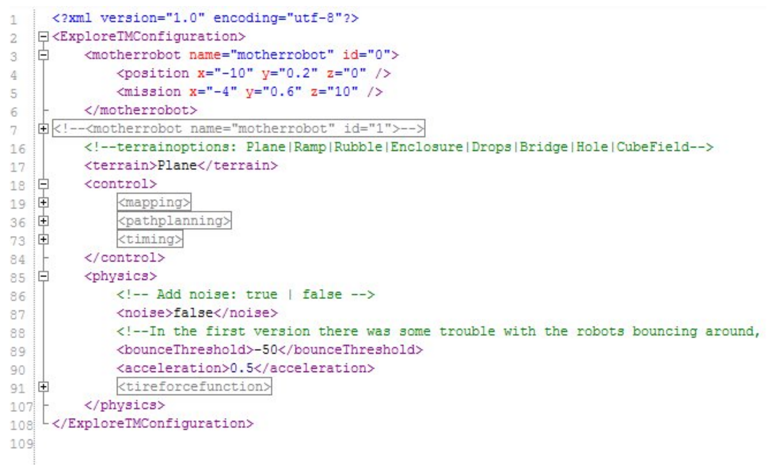


Figure 4.7: Probability distribution for a long range finder with activated noise and a current distance reading of 1.5 m



```
<?xml version="1.0" encoding="utf-8"?>
<ExploreTMConfiguration>
    <motherrobot name="motherrobot" id="0">
        <position x="-10" y="0.2" z="0" />
        <mission x="-4" y="0.6" z="10" />
    </motherrobot>
<!--<motherrobot name="motherrobot" id="1">-->
    <!--terrainoptions: Plane|Ramp|Rubble|Enclosure|Drops|Bridge|Hole|CubeField-->
    <terrain>Plane</terrain>
    <control>
        <mapping>
        <pathplanning>
        <timing>
    </control>
    <physics>
        <!-- Add noise: true | false -->
        <noise>false</noise>
        <!--In the first version there was some trouble with the robots bouncing around,
        <bounceThreshold>-50</bounceThreshold>
        <acceleration>0.5</acceleration>
        <tireforcefunction>
    </physics>
</ExploreTMConfiguration>
```

Figure 4.8: Extract of the configuration file

4.2.4 How realistic is the simulation

Research for robots which has been done solely with simulation does not easily transfer to real robots and the final solution can perform good in simulation, but badly in a real world scenario. The reason is usually a lack of environment noise. To avoid failing in the real world, the simulation should contain a realistic noise model. Due to its simplicity, the most common approach for modelling noise is Gaussian. The Gaussian model is not the ideal model for real noise and alternative approaches have used mixed distribution models with success (refer Chapter 2).

For the simulation of sensors this project includes a noise model with a mixed distribution. For some parts, such as localisation, assumptions are made to limit expected variance in the sensor data. But to meet these assumptions in a real environment, additional filter mechanisms are required (refer Chapter 5).

For an application to a real world robot, a robot needs configuration and calibration for the specific sensors. That includes a deep experimental analysis to find the characteristics of attached sensors. Learning an inverse sensor model from sensor data and a known environment with a neural net [TBF05] represent one possible approach.

Every hardware platform has its own characteristics and this adaption process is essential for an application in the real world. Thus, configuration and calibration is a necessity for a real world robot, but can be ignored in a simulation.

4.3 Analysis of the robot design

Part of the current project was an analysis of dynamic, static and simulated situations. The results of the analysis led to a change of the current robot design and sensor configuration[2]. The changes to several parameters are outlined in Table 4.3 and in addition two vertically mounted long range (3 m) infrared sensor array were added to the front of the robot.

[2]While a real implementation of a robot is not powered from an everlasting energy source, this project ignores the issue of battery discharge and uses the specifications for a fully charge battery for the simulated robot operation.

In a previous project, the mother robot was tested in a real environment [Wil07], and video capture were obtained. Due to the fact that the real robot's gearboxes are currently broken, the real life experience is based on these short video captures which show following situations:

1. moving over a field of rubble
2. moving over an obstacle to show the body joint functionality
3. moving straight 'into' a wall and flipping the robot

The video captures were too short and too constrained to provide a useful analysis of the robot's behaviours (though they were still useful to calibrate the simulation).

	original	**changed**
Wheel radius	15 cm	20 cm
Max velocity	0.5 $\frac{m}{s}$	1 $\frac{m}{s}$
Max acceleration	0.4 $\frac{m}{s^2}$	0.5 $\frac{m}{s^2}$

Table 4.3: Changes to the modelled device

Analysis showed that the original configuration did not perform as expected in simulation. Some of the problems identified in the analysis were:

- physically constrained to traverse steps higher than 17 cm
- unable to autonomously overcome obstacles higher than 10 cm safely
- heavily limited to detect steps, inclines and potential drops with range finders mounted all in one horizontal plane

The presented changes, especially the enlarged wheels, allow a more consistent behaviour for overcoming higher obstacles. The addition of two vertical sensors arrays allows the robot to detect steps, inclines and small drops to the front of the robot, while the symmetrical design is still maintained to support inverse operation. Figure 4.9 shows the new sensor configuration.

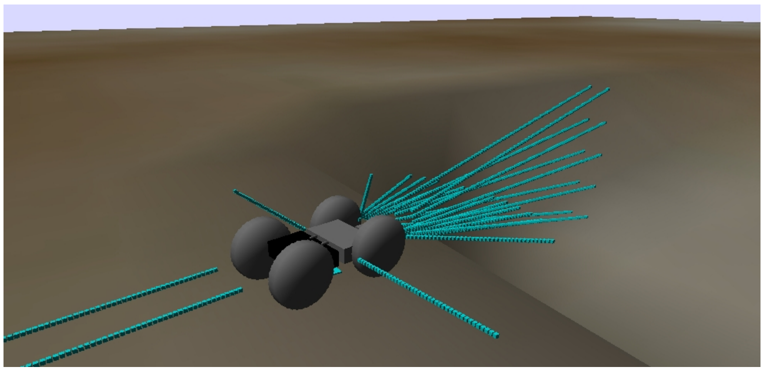


Figure 4.9: Modified front sensor configuration

With the new configuration the robot can negotiate obstacles up to a height of 15 cm, but has some persisting problems with large obstacles and steep angles of attack. The closer the obstacles come to a height of 20 cm, the more unpredictable behaviour the robot motion gets. The reasons for that can be divided into three categories:

1. limitations of the simulation
2. limitations of the control algorithm
3. limitations of the actual robot model (hardware design)

The limitations of the hardware platform are either related to perception and sensors, or to motion. Some of the identified limitations of the hardware platform led to the changes listed in Table 4.3 and adding two vertical infrared sensors array offers better robot perception. But the robot's perception is far from a desirable state for a robot moving in a cluttered field. Even with the enhancements the robot is unable to differentiate between a drop or a long stretched declining surface. Perception of the robot is therefore still limited and has a significant impact on the performance and selection of strategy for mapping, path planning and navigation.

With the modified sensor configuration the detection of surface inclination is still limited to a small angular range to the front of the mother robot, where the range finders are mounted vertically.

Thus, to distinguish between ramps and non traversable obstacles the mother robot has to approach the obstacles closely, depending on the max-

maximum traversable slope	step = 0.2 m	step = 0.15 m
30°	0.35 m	0.26 m
25°	0.43 m	0.32 m
20°	0.55 m	0.41 m
15°	0.75 m	0.55 m
10°	1.13 m	0.85 m

Table 4.4: Distances after which the robot can distinguish between incline and obstacle

imum incline the robot has to deal with. The robot platform can deal with an inclination of up to 30°, but it has to approach a not yet classified obstacle until it can distinguish between ramp or non-traversable obstacle with its planar mounted range finders. The maximum traversable incline and the maximum traversable step define how close the mother robot needs to approach a potential ramp (refer Table 4.4). Then the mother robot either reaches the inclination of a ramp and the changed attitude of the robot allows a new interpretation of the range finder measurements, or the robot faces a non-traversable obstacle/ramp.

A further issue arises from the skid steering drive in combination with the twisting body design. The simulation shows that a mother robot which tries to negotiate an obstacle with a steep angle of contact, usually fails to overcome obstacles larger than 15 cm.

The reason can be partially found in the observation of the following situation: the robot is moving on a plane and only one wheel is in contact with an obstacle. The resulting motion twists the front body and the rear body. Because the front body moves on top of an obstacle with one wheel, the joint is lifting while following a curve as a result of the twisted front body. The effect is that the rear body is twisted away from the original direction of motion (of the front body). This makes it more difficult for the rear wheels to climb up the obstacle. Figure 4.10 illustrates the effect.

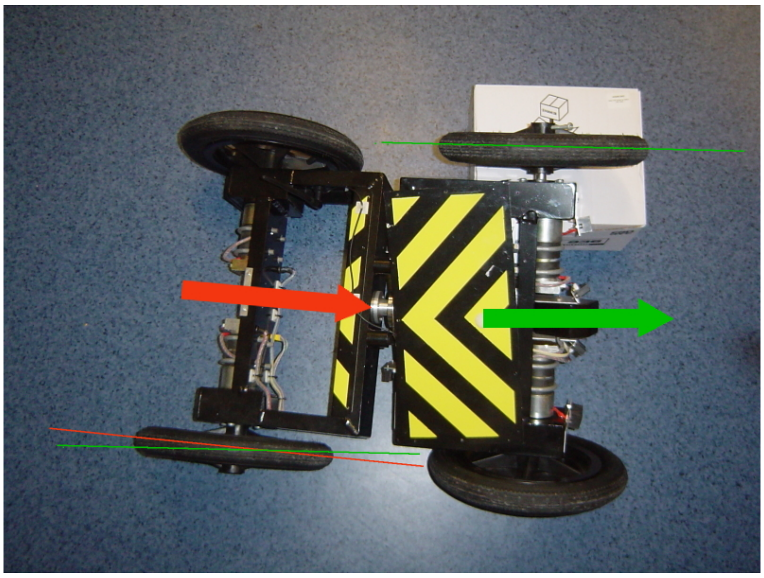


Figure 4.10: Front and rear body are moving in different directions, while overcoming obstacles

4.4 Evaluation of the simulation

Though a simulation is a great way to provide initial results, it has some limitations.

The physics configuration including the wheel parameters could only be empirically set by trying to imitate the real behaviour of the robot as seen on the videos mentioned in Section 4.3. The parameters of the tire function have a great impact on this behaviour. Because a longitudinal and lateral tire function can be defined, a good combination of both had to be found to achieve a realistic behaviour for overcoming obstacles and skid steering at the same time. The values chosen for these functions are a result of an empirical tuning process with the main criteria:

- allowing a point rotation of the robot, without twisting the body
- stable overcoming of obstacles up to a height of 15 cm
- overall impression of motion consistent with the real device

Given the limited amount of video of dynamic behaviour of the real robot, the current values can only be an approximation.

Extremum slip	2.5
Extremum value	0.7
Asymptote slip	3.5
Asymptote value	0.01
Stiffness	1500

Table 4.5: Tire force function - longitudinal movement

Extremum slip	2.5
Extremum value	0.05
Asymptote slip	3.5
Asymptote value	0.01
Stiffness	15000

Table 4.6: Tire force function - lateral movement

MRT does not use the full flexibility of the physics engine at the moment. This is mainly due to the indirect access to the physics engine through MRT. By providing only a reduced interface to the underlying physics engine, some advantages of the high fidelity engine seem to be lost [3]. During the development of this thesis, updates to MRT already addressed some of these issues and others might be resolved in future versions of MRT by having a broader API. Most of the limitations of this simulation environment are common to many others, because simulated physics are not perfect and a lack of noisy environments will remain. Though this project has shown how to implemented noisy elements for the simulation, it is far from complete.

This project embeds ExploreTM in a communication infrastructure which is specific to MRT. Thus, trying to expand the current work to other platforms (such as Linux), requires a deeper analysis of the involved message types. That means that further development is initially bound to MRT. This is a well known and accepted limitation, because MRT offers other advantages, which speed up implementation and ease overall development.

[3]The underlying API offers a more complex parameter setting.

The computational resources for this project are currently limited and require efficient algorithms with a careful setting of control parameters to allow for a multi robot simulation. The control interval and map size are the major factors to be mentioned here. The anticipated control interval by the previous project [Wil07] is $40 - 100$ ms. However, an investigation showed that the infrared range finder arrays only have a response time of 175 ms. This project adopts a common control interval of 200 ms (control frequency of 5 Hz). But even this cycle might not always be maintained by the simulation, given a large number of simulated mother robots; this project addressed this issue by using the independent simulation time instead of real time and triggers sensor updates only based on the simulation time.

4.5 Possible solutions and extensions

Simulation

Further effort should be taken to improve the simulation. The use of the physics simulation in a more direct form should be considered, although the simulation of sensor devices would then have to be addressed separately.

Clearly, the better the characteristics and specification of the real device are known, the better the modelled device will be in simulation. More information should be derived from (field) experiments to improve the matching of the kinematics of the real model with the simulated robot. Experimental values can enhance the error model of sensing devices and to improve the kinematic simulation the evaluation of damping and friction coefficients can be sensible, though considered to be of lower priority.

Robot design

Perception is one of the greatest challenges this robot design offers. With the current set of sensors, and even with the ones added by this project, perception is very limited. In order to operate in a highly cluttered environment and maintain the low cost property of the robot design at the same time, a better solution has to be found. Investing in a larger set of infrared sensors is an option, but would require additional micro-controllers to maintain a control cycle of 5 Hz.

The mother robot's current design lies within the requirement of the Department of Homeland Security, but having an four wheel drive control would be a clear advantage for the wheeled robot, although also a fundamental change of the moving platform can be considered. Currently, the current skid steering device limits motion to rather simple terrain structures and has negative implications for localisation. It could also lower the impact of steep angles of contact between wheels and obstacles. Choosing different wheel sizes with greater width offers a quick fix to create better suspension and grip, but it will worsen the properties of skid steering — wheels with steering angles would be required.

4.6 Summary

The implementation of a simulation has brought valuable results for current and future development of the robot:

- A valid and faithful simulation has been implemented including the design of a robot model and specific sensor models. It allows easy reconfiguration and testing of multi robot cooperation with this robot model, which will ease further research with this robot model.
- Major shortcomings of the robot design have been identified and this project addressed those issues with changes in the sensor configuration and the wheel design.
- This project suggested and implemented some noise models to enhance further research with this simulation environment. However, these models are not used in subsequent evaluations of the control model.
- A simple configuration interface guarantees a quick start with the simulation for beginners and simplifies testing.

It is important to deal with the limitations of the simulation, but identifying shortcomings of the control algorithm and the hardware model is the major focus and provides valuable input for further research and development. The limitations of the control algorithm and its modules will therefore be addressed separately in the following chapters.

Chapter 5

Localisation

Localisation is the task of estimating the current position of a robot and is an essential task to allow mobile robots to operate autonomously. For the simulation, no localisation is necessary, because the exact position of the robot is known. However all real localisation techniques have errors, and therefore the simulation should have a realistic error added to the actual position. This chapter describes the possible localisation techniques in order to arrive at a reasonable error estimate.

Active and passive localisation techniques exist to allow the computation of a position estimate, in order to obtain a good estimate of the current position. *Active* techniques influence the control of a robot and force an active pursuit of known landmarks in an environment. On the other hand, *passive* localisation techniques rely solely on previous and current sensor inputs. This means that the accuracy of internal and external sensor measurements are crucial for a good performance. To provide internal sensor measurements, inertial navigation systems (see Section 5.1.2) can be used, while such systems as the GPS make use of external resources by generating a position estimate from data produced by orbiting satellites, but its accuracy is highly dependant on the signal receiver quality.

For both approaches, the application of multiple robots increases the complexity, because localisation needs to be valid for the whole group of robots. The error of the relative positions between robots has to be low enough to allow a valid working of mapping and other high level control algorithms such as path-planning. A suggested solution for the multi robot scenario is to actively rendezvous two different robots [BMF$^+$00]. In this case, each robot will have its own prior hypothesis about the current position. By meeting other robots, their different hypotheses can be compared and be utilised for a correction of the position estimate.

[TBF05] represent localisation challenges with three different scenarios:

Tracking: a robot has to track its movement path

Global localisation: The general global position has to be found

Kidnapping problem: After losing localisation information caused by erroneous hardware or such, a robot has to recover and find the global position again

The search and rescue scenario causes the worst kind of localisation problem, i.e. the kidnapping problem. Drops might exist in the environment as well as unstable elements which the robot needs to traverse. A robot moving in such area can experience unexpected and sudden position changes. Odometry (see Section 5.1.1) will be useless in such situations and only an inertial measurement system (see Section 5.1.2) can help as an internal sensor. However, the reception of GPS signal or similar would ease the situation, by providing an upper bound for the localisation error.

If a map exists from the environment, a solution might be found using Markov Localisation [TBF05]. Unfortunately, the presence of a valid map cannot be expected in a search an rescue mission, which will take place in collapsed buildings or similar disrupted and irregular structures.

In combination with mapping, the problem of localisation is commonly found in literature as Simultaneous Localisation and Mapping (SLAM). Navigation is possible without localisation [SZ06], but this does not allow a valid map to be built a the same time. Because map building is essential for path planning and effective exploration techniques, localisation is necessary and the usage of probabilistic methods have created a number of solid approaches to address the issue of SLAM [DWB06][BDW06]. Most of the technologies addressing SLAM deal with it in only two dimensions. This constrains their application to planar environments such as indoors; approaches that are generalisable to three dimensions increase complexity and require more computing resources to achieve a tolerable accuracy.

The current project does not intend to solve the localisation problem with SLAM techniques, but a localisation error in the mapping and control algorithm has to be considered. The following sections evaluate existing and upcoming techniques, which can be part of an implementation to provide a position estimate. The chapter will conclude with a final error estimate.

5.1 Relative localisation techniques

Relative localisation techniques provide localisation information relative to a known starting point. These techniques also fall under the well known category of dead reckoning. One of the best examples for dead reckoning is the navigation with a compass. The target coordinates relative to the starting position are known, but the path has to be found by keeping track of the past distance and the heading information.

5.1.1 Using odometry for localisation

Odometry data for localisation is generated from wheel encoders. A model for a differential drive is given by Siegwart [SN04], where the current position can be inferred from an initial position and collected odometry data:

$$position = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \theta/2) \\ \Delta s \sin(\theta + \theta/2) \\ \Delta\theta \end{bmatrix}$$

where

Δs	$= \frac{\Delta s_r + \Delta s_l}{2}$
$\Delta\theta$	$= \frac{\Delta s_r - \Delta s_l}{b}$
b	= distance between wheels
Δs_r,Δs_l	travelled distance of the right/left wheel

The accuracy of these data depends on the resolution of odometry encoders. In general, odometry data has a questionable usefulness. Errors accumulate with time and the final error can be $\gg 100\%$ [TBF00]. The reason for this high error lies in a multitude of influences on the accuracy of odometry, i.e.:

- resolution of encoders
- alignment of wheels, i.e. calibration
- variance in wheel radius, i.e. variances of the hardware properties
- contact point of the wheel
- ground characteristic, i.e. friction coefficients

These errors can be grouped into systematic errors, which can be reduced or even eliminated by calibration, and nondeterministic errors — unpredictable influences of the environment. In addition, the robot movement has a direct influence on the integration error. Because the mother robot operates with a skid steering system, turns especially worsen results. Beside turns, the ranging of a robot is affected by drift, i.e. different errors of the left and right wheel imply a change of orientation, when there may none. Overall, skid steering causes odometry data to be highly inaccurate.

The current implementation includes a model of odometry error, which focusses on a general ranging error. The error is based on the assumption that the errors of the left and right wheels are independent and the variance is proportional to the absolute value of the travelled distances. The covariance matrix for the model of odometry is as follows:

$$\Sigma_\Delta = \begin{bmatrix} k_r & 0 \\ 0 & k_l \end{bmatrix}$$

$$\Sigma_\Delta \begin{bmatrix} \Delta s_r^p \\ \Delta s_l^p \end{bmatrix} = \begin{bmatrix} \Delta s_r \\ \Delta s_l \end{bmatrix}$$

where

k_r, k_l	error constant for right and left wheel
$\Delta s_r^p, \Delta s_l^p$	noise free distance measurement for right and left wheel

Based on the experiments of [VDLM06], errors in the range of 0.4% per metre travelled are realistic. For a noise model, a random process selects error constants k_n from a Gaussian distribution around 0.6%. This slightly higher value is justified by the skid steering. A standard deviation of 0.2% applies as well. Each wheel has a separate error model.

5.1.2 Using an Inertial Measurement System

IMS [TW97][GWA01] are widely known and used in the field of robotics to estimate position and attitude. Their application is often found in aerial vehicles. The general IMS consists of accelerometers and gyroscopes. Accelerometers belong to the category of Micro-Electro-Mechanical Systems (MEMSs) and give information about the current acceleration a of the device. The acceleration is measured for a specific axis and relative to the gravitational force. Meanwhile, gyroscopes measure the rate change ω of a given axis, which is usually measured in degrees per second.

There are two different methods to mount an inertial measurement system and each have a different reference frame for subsequent computations:

(i) platform based

(ii) strapdown

Characteristics of platform IMS

The platform based IMS uses accelerometers and gyroscopes which are gimballed. Gimbals provide (relative) independence from the movement of the vehicle which contains this system. Hence, platform based IMS provide data in the global (world) frame.

Characteristics of strapdown IMS

The far cheaper and technically easier solution is a strapdown IMS, which is commonly used in mobile robotics. A strapdown IMS is rigidly attached to the body of the moving device. Therefore all measurements of a strapdown IMS are done with respect to the body frame of the moving device[1]. But position and orientation data of the robot needs to be provided with reference to the global frame to be useful for mapping. Thus, any strapdown IMS measurements have to be converted into the global frame, requiring complex computations. Due to its fixed position on the robot, a strapdown system also needs to be able to measure higher turn rates[TW97]. Therefore IMS components have to be carefully selected, also depending on the expected motion patterns of the robot.

Accuracy of IMS

The accuracy of an IMS is affected by a number of factors:

- proper mounting and calibrating of the device
- temperature (affects gyroscopes)
- operational limits (e.g. maximum measurable turn rates), exposure to vibrations and shocks during operation
- magnetic fields in the operation environment

Other influencing factors are also the travelled distance and the accuracy of knowledge about the initial position.

Compensation for the above influences starts with a thorough testing of every device, in order to determine specific characteristics, i.e. the systematic errors. Either complex and quite specific methods compensate each of these errors or systems of the same type use a general compensation technique. Such a general compensation technique can use bias, a scale factor error, and temperature as inputs. Initial errors propagate and increase the position error with every calculation over time, especially when a transformation into the global frame has to take place. This adds to the complexity, so that ”a simple rigorous calculation of errors is not usually practical” [TW97]. For that reason an important factor for all IMS is the initial alignment and knowledge about the initial position of the device. Better knowledge about the initial position creates a better posterior position estimate. Hence, in a real scenario of an IMS application - effort has to

[1]In order to allow an easy conversion from the body frame in the world frame, the IMS has to be properly aligned to the vehicles moving axis

be spent on an initial alignment and calibration routine. A design of such initial alignment and calibration has to consider the main error sources such as:

- fixed bias
- scale factor
- correlation / sensitivity towards movements on an axis which is not the measured one
- vibro-pendulous errors

Some of these errors can have a time dependence e.g. day to day or switch-on-to-switch-on intervals. Others can have changing values based on thermal-effects. Overall all those effects have to be considered in order to reduce the initial error of an IMS. For USAR situations it is important that the setup phase for each device is as short as possible but also as accurate as possible. In a simulation the issues of an initial setup can be ignored, because the modelled device represents a perfect setup.

Murphy [Mur00] provides with ″0.1 percent of the distance travelled″ an example for the achievable accuracy of an Inertial Navigation System (INS), that operates on base of an IMS.

5.2 Absolute localisation techniques

5.2.1 Using the Global Positioning System

The GPS offers a way of absolute localization [DJ00] in contrast to the relative localisation of an INS. GPS relies on a system of 24 satellites orbiting the earth and a GPS signal receiver, which has to be (most often) in a line of sight of at least three satellites. Four ″visible″ satellites provide a better position estimate. GPS comes with different options to increase accuracy. One is Differential Global Positioning System (DGPS) which is designed to provide position and velocity data better than an accuracy of horizontal ±5 m and vertical ±10 m [YMD08]. Depending on the receiver quality (and thus increasing costs) more correction data are utilised to improve the position data and higher accuracy can be achieved [2].

5.2.2 Alternatives to the Global Positing System

European Satellite Navigation System

The European Satellite Navigation System Galileo [Eur08] will offer an alternative to the GPS in the near future and is based on a network of 30 satellites. Though not yet in a state of commercial use, the Galileo system promises equal or even higher localisation accuracy compared to the GPS, especially for non-equatorial countries. Additionally, Galileo will provide a qualifier for the integrity of the current signal, allowing a classification of the received localisation information. The Galileo project also claims the complementary use with the GPS statement as one of the main advantages. The combined use will offer better coverage due to the denser satellite infrastructure and increased precision and reliability. The stated future goal is an accuracy smaller than 5 m. A combined system covers even high latitude areas such as Antarctica, which have far worse accuracies otherwise because GPS is equatorial.

Augmentation services

Satellite-Based Augmentation Systems (SBAS) are regionally applied system to increase the accuracy of systems such as GPS and Galileo. DGPS and the European Geostationary Navigation Overlay Service (EGNOS) are examples for SBAS [ESA08]. EGNOS is designed to achieve accuracies down to 2 m.

5.3 Alternative localization methods

IMS and GPS are not the only approaches towards localisation. Current research uses the signal strength within wireless networks to find a position estimate. But determining the position with wireless sensors, requires a

[2]The 'Crescent Vector OEM Board' [Gro07] for example offers submetre accuracy for 95% of the time

good coverage of the target site and beacons might have to be manually inserted. Additionally, the relative positions of the static wireless devices has to be accurately known. Experimental studies show an achievable accuracy of about half a metre [HSS03], with an experimental setup based on an intact office space.

5.4 Combined solutions

A combined solution can help to reduce the overall uncertainty. A common technique is the combination of IMS and GPS. The error between the two different measurement systems is the input to a filter mechanism, e.g. a Kalman filter [ZM00] which helps to find the best estimate.

IMS and GPS updates are received at different frequencies — the GPS updates are usually received at a lower rate. Therefore the IMS has to work independently over a smaller time frame until a GPS update has been received and serves therefore as the main technique for position estimation. As soon as a GPS update has been received, the position estimate can be corrected. For the correction, the absolute position received from the GPS is fed into the mentioned filter mechanism. Depending on the receiver, DGPS alone can provide a bounded-error position estimate with an accuracy below 5 m [Wil07]. This error is bounded, because in contrast to IMS the GPS provides an absolute position. However, the combination of INS and DGPS has shown in theory [YS07][CGB01] and practice [KLC+03][Cra97] (operation time < 1 h) an achievable accuracy below 0.5 m, given regular DGPS updates.

5.5 Limitations

For a real application in a field of rubble, other aspects have to be taken into consideration. Robot movement is expected to have sudden changes, due to uneven surfaces including drops. This degrades the quality of the data from the relative localisation systems: acceleration and turn rates have to be within the maximum specification limits of the IMS hardware. This project refers the solutions of these problems to future projects and will assume error free IMS data and thus accurately known orientation and attitude.

5.6 Suggestions

Though the most common solution is the combination of GPS and INS, no study could be found for a GPS/INS system operating in rough terrain; an overall performance study accompanied by a terrain analysis in order to compare expected acceleration and turn rates to real ones would give more insight into the accuracies to expect.

A performance boost can be expected as soon as the Galileo project becomes operational, thus complementing the existing GPS. In any case, the expected error will be bounded. Eventually probabilistic methods should be used to improve the overall system performance.

A SLAM system such as FastSLAM [MTKW02] uses a landmarks based mapping approach and can estimate the robot position with an error below 0.4 m depending on the number of found landmarks. Combining SLAM with an absolute positioning system leads to a constrained SLAM approach. Lee et. al [KWLG07] have already shown how to constrain SLAM with a previously known roadmap, which suggests that FastSLAM in combination with GPS can be a desirable solution.

Another alternative is the application of relative localisation by constructing a local wireless network [Yan07]. This network can provide information about relative positions of robots within the area. With wireless devices on the robots, position estimation might be done with triangulation, as long as the number of robots is equal to or higher than three. However, signal coverage in a cluttered environment has to be taken into account and an implementation of a wireless network will likely only serve as additional correction information to an estimate from the global positioning system.

5.7 Summary

This chapter presented different localisation techniques: relative, absolute and combined ones. It pointed towards practical solutions of the localisation problem and presented their individual error estimates. While the device characteristics and the filter algorithm are important for achieving good results in a real scenario, this project will only include a realistic estimate of the expected localisation error.

Based on the analysed techniques, this project estimates an achievable upper bound for the localisation error of 30 cm with a medium cost receiver. Advancing technology will improve the tradeoff between receiver cost and quality and will make this localisation error also realistic for a low cost device. This project assumes the distribution of this error to be Gaussian rather than bounded. Due to the properties of the Gaussian distribution and based on the three standard deviation rule, 99.7% of these errors lie within the boundary of three standard deviations of the mean. The error is modelled therefore by a Gaussian with a standard deviation of $\sigma = 10$ cm.

Chapter 6

Mapping

One of the core tasks to allow for robot planning is a robot's ability to map an environment. This chapter evaluates some existing techniques and describes how this project solves the mapping task for the search and rescue scenario.

Initially a search and rescue robot brought to a new disaster site will not have any information about its environment. It may be possible for a human operator to provide blueprints of the building, but this will not be useful. An existing map needs translation into the robot's internal representation of the world and results in an increased setup time for the robots. This measure is without any predictable benefit for the search operation, because a given map (building blueprint or similar) cannot fully represent the current environment and is often not up to date or just inaccurate. More importantly, with the current application being Urban Search and Rescue the operation area is expected to have changed in a major sense.

Therefore, the robot must be able to construct its own map of the site and gather as much information about the environment as possible. To do so, the robot can rely on camera or range finder input or solely use the position estimate to memorise any region travelled so far as being accessible. So by just interacting with the environment, the robot gains some - even though limited - knowledge. Clearly, the application of cameras or range finders will significantly improve the quality and quantity of information and allow safer navigation.

For map building it is important for the robot to determine its (three dimensional) position, be it absolute or relative to a constructed map as in SLAM. This project assumes the localisation problem has been solved by one of the techniques discussed in Chapter 5.

The mother robot's main perceptual input comes from range finders and it will use the gathered spatial information to build a 3D map of its environment. The type of range finders has a significant impact on how accurate this map will be:

- Laser Range Finder (LRF) are often used when an exact three dimensional knowledge about the environment is required. However, LRF are costly and quite large.
- Infrared rangefinders are cheap and are similar to single ray laser rangefinders in the sense that they use a single sensor ray. But their range is lower compared to the LRF and they suffer more from noise. The infrared rangefinders of the mother robot also have a minimum range, so that they cannot sense obstacles which are too close to the robot and instead might produce false sensor readings.
- Ultrasonic devices operate with a wide beam. The received information is often more difficult to process, because these devices suffer significantly from noise and lower resolution.

The requirement for a 3D map and the use of statically mounted infrared finders has implications on the design of the mapping strategy, which will be part of the discussion in the following sections.

6.1 Mapping with limited perception

6.1.1 The perception of the mother robot

The perception of a robot depends on the number and configuration of its sensors. The mother robot uses a set of infrared sensors to provide spatial information [Wil07]. The configuration consists of high sensor density at the front and a low density at the rear of the robot. While the original device uses sensors which are only horizontally mounted, this project added a pair of vertically mounted array sensors to the front of the robot model, in order to enhance the perception. As a reminder, Figure 6.1 shows the sensor rays of the robot on a plane.

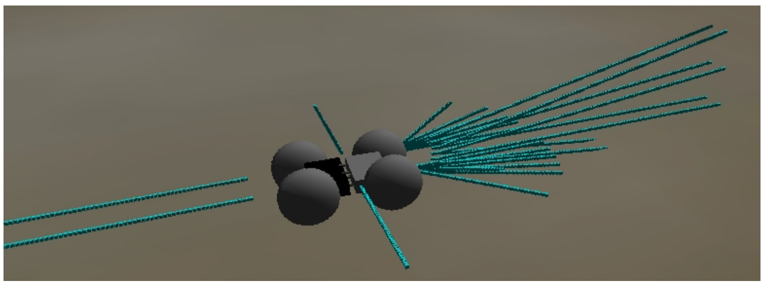


Figure 6.1: Sensor configuration blue squares are part of the sensor ray

The spatial information gained from the range finders at every update is augmented with inferred data, based on the fact that the body of the robot fills free space at the current position.

Sensors and inference produce individual position updates, which classify occupied or free space. But under realistic conditions this data will be erroneous, because:

1. limited achievable precision, current mounting and calibration of infrared sensor affect measurement quality
2. specular reflections in the environment cause false range readings
3. the localisation error causes measurements and inferences to be associated with wrong positions

Though this project does not use techniques to address these issues directly, it integrates a value of belief and a value of variance with each sensor update. Belief describes the probability that the current measurement is classified correctly to be a measurement of occupied or free space. Variance describes the current localisation error of the measurement. Belief and variance are currently set to default values but allow an easier implementation of compensation techniques in future projects. Belief is set to a standard of 0.95. Variance is set to $\sigma^2 = 0.01$ m based on the reasoning in Chapter 5.

6.1.2 Long term versus short term mapping

Mapping usually refers to the process of building a permanent map. However, the project also uses a short term local representation of the space around the robot based on the range finder measurements in the previous n control intervals. This representation is needed for reactive navigation (see Chapter 8). A set of temporary valid spherical coordinates (a so called point cloud) represents the sensor updates and thus the world from a robot egocentric viewpoint. This point cloud is the robot's temporary knowledge of the environment, which navigation (see Chapter 8) uses for reasoning. The point cloud is updated every control interval. Its points only represent the end points of the sensor rays and they have to lie within a moving circular region (or sphere in 3D) with diameter d_w (typically d_w=6 m) around the robot's centre. They are kept only for a maximum time t_m (typically t_m=1 s).

Obstacles are typically detected at long range. Nevertheless, a complex environment and the limited number of statically mounted range finders can also cause late detections and sensor data can be erroneous when obstacles are too close. This project assumes that an improved mounting of sensor devices and appropriate filter mechanisms prevent the integration of falsely generated measurements into the long term map. This results in a blind zone for the long term mapping process. However, because a general knowledge of close obstacles should be available for reactive navigation, the project assumes the distance to close obstacles (below minimum range) to be the minimum sensor range. This last assumption applies to the short term mapping only.

6.2 Evaluation of different map representations

Mapping a cluttered environment cannot work properly with a two dimensional representation such as a simple occupancy grid. The reason lies in the higher complexity of the space. Accessibility of specific positions can depend on the direction of travel, for example a higher platform might be inaccessible from one side, but accessible through a steady slope from another side. For that reason an evaluation of different available mapping solutions follows. The evaluation takes the following criteria into account:

1. complexity
2. memory footprint
3. extensibility
4. usability (features)

6.2.1 Occupancy grids as common choice

The occupancy grid is one of the most commonly used techniques for mapping in two dimensions and occupancy grids can be expanded to three dimensions (see Chapter 2). Occupancy grids are extensible to a great degree, because they provide a good representation of space down to a certain resolution. Being a rather basic data structure, occupancy grids allow flexible postprocessing. Furthermore the initial integration of rangefinder data is a straight forward task, because every cell is directly addressable. It also offers the inherent integration of localisation errors, considering that probability values describe cell occupancies. But being such a basic data structure requires further (complex) processing techniques in order to allow for path planning or other evaluations.

Though occupancy grids are initially easy to implement, the memory footprint may be large. A naive implementation has to deal with a three dimensional block of cells, which represent either free or occupied space. High grid resolution set a large memory requirement for a limited 3D space. The choice of resolution applies to all three dimensions and involves a trade off between accuracy and performance (memory constraints); a resolution of about $15 \times 15 \times 15$ cm^3 is a common choice, so that a small, fully explored map of $50 \times 50 \times 3$ m^3 would create about 2 million individual cells. There are other ways to reduce this memory requirement using dynamic initialisation techniques, octrees [Wat93] or custom strategies. But an implementation will clearly suffer from increasing complexity to reduce this memory footprint.

6.2.2 Height field maps are widely applied in games

Height field maps or elevation maps are common to games. The strength of elevation maps lies in an effective representation of a two and a half dimensional space with a very small memory footprint. Furthermore, elevation maps allow a bounded representation of space based on measurements of occupancy and free space. Such a representation can be a good solution for a robot architecture like ExploreTM, considering that robots send map data frequently to a central unit. In order to create a map to do path planning, elevation maps offer a proven solution, being part of most three dimensional games.

The disadvantage of using a height-field is obvious: the representation is limited to two and a half dimensions, so that a robot cannot deal with any environments which have any kind of multiple layers, i.e. alternating free or occupied space within the vertical direction.

6.2.3 A custom multi-level surface map

MLS maps are a recent development and offer a representation for three dimensional environments with much lower memory constraints than alternative approaches. MLS maps can represent a three dimensional environment and also integrate a factor of uncertainty. They include a slightly higher complexity to make an intelligent use of robot size constraints and representing space efficiently. The application of this representation in combination with a laser range finder has proven to be successful [TPB06] with exploration tasks.

6.2.4 Discussion of different map representations

The specific exploration purpose makes a high resolution map desirable but not essential. All three representations have an underlying 2D Grid in common and resolutions of 100 cm^2 and 625 cm^2 is assumed for the comparison.

All measurements include a localisation error, which will be even harder to eliminate for a representation in three dimensions. Incremental map building is an essential element of design of ExploreTM and means that multiple robots are involved in building the overall global map on the grandmother robot. The mapping strategy has also to allow for future projects and techniques which reduce the incorporated localisation error from multiple robots.

For the sake of a responsive simulation, and eventually a system architecture which operates without significant delay, only a few reasonable options exist:

(A) **Three dimensional occupancy grids** appear to be the most resource consuming choice, while dealing with a probability value of a huge number of cells. Though resulting memory constraints can be addressed in general by octrees, the association of probabilities with individual cells interferes with this countermeasure.

(B) The lowest resources are consumed by **height-field/elevation maps** and they fulfil the requirement of resolution and memory constraints easily. Nevertheless, the two and a half dimensional representation creates a major limitation for upcoming projects.

(C) As described above, **MLS maps** offer the most promising features, which the current project requires. They even relax the resolution constraint and offer an unlimited degree of resolution for a third dimension (as do height field maps). This map representation has also been successfully applied to incremental mapping [TPB06] and overall offers the most practical solution to the mapping requirements.

For the given reasons, this project implements a mapping algorithm on the idea of multi-level surface maps. MLS maps are built upon a (horizontal) two dimensional grid filled with specific elements to describe the third (height) dimension of each grid cell. In the following, the term 'grid' refers to this underlying two dimensional grid (of MLS maps).

6.3 Generating data points as local space description

The mapping systems contain two core modules. The first module processes sensor readings into a description of local space around the robot; the second updates a global map with the latest local data. The global map is a customised version of the MLS map.

The 'perceptual field' translates the individual range finder readings, which the hardware provides into a set of three dimensional data points describing free and occupied space around the robot.

The module bases its translation on instances of a single ray sensor model. Each single ray sensor model needs the following individual settings:

- mounting position of the robot, i.e. relative to the robot centre
- orientation, i.e. direction of the sensing ray
- minimum/maximum range

The sensor model allows the translation of a range finder reading either into world coordinates, which are needed to build a long term map, or into a set of local coordinates relative to the current robot centre. The latter allows an easier computation of reactive behaviours.

6.3.1 Data points from ray casting

A method similar to ray casting, which is popular in computer graphics [Wat93], computes the sensor data points. The following procedure produces data points with absolute positions:

1. transform the sensor model (orientation and position) into the coordinate frame of the robot
2. follow the ray direction from the sensor position with step size *k* and create a datum point at each step

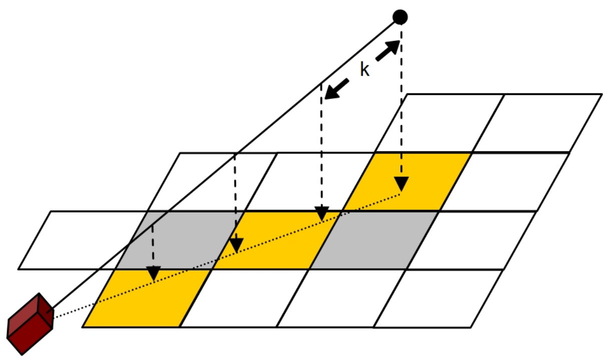


Figure 6.2: Model of single ray sensor

Figure 6.2 visualises the data point generation process, where data points lie on the solid black line between sensor and endpoint.

The advantage of this procedure lies in its independence from any underlying map representation and it therefore has a higher computational efficiency. Other existing strategies, e.g. the so called beam sensor model [TBF05], are intertwined with an underlying grid structure. Such an approach needs to iterate over a large number of cells and verify any intersection with the sensor ray. With the procedure presented, however, the properties of the sensing ray can directly output data points. A parameter k defines the sensor ray resolution so that the performance of this procedure can be controlled.

However, it is arguable that a loss of information occurs while using a fix step size. Some grid squares along the path of the ground projected sensor ray may be left out - e.g. the gray shaded squares in Figure 6.2. The reasoning to stick with this approach even for an underlying grid map is based on the following observations:

- grid cells which are left out usually have only a short intersecting line with the projection of the sensing ray
- grid cells that lie centered on the ground projected sensing ray have a longer intersection and might also contain multiple data point measurements from a single device
- the step size can be arbitrarily reduced to meet a desired specification

Therefore the length of each cell intersection with the ground projected sensing ray can be interpreted as a degree of confidence that this cell has actually been measured. Hence, this method only leaves out cells that fall below a confidence threshold — directly depending on the parameter k.

6.3.2 Data points from shadow projection

A robot fills free space with its physical body. Therefore, the robot's position allows inference of a section of actual free space. In order to derive data points from this information, so called shadow projection is applied [Hei93]. A number of grid cells around the robot are tested whether they fall under the robot's body and the height of the robot body at that position is computed. The front body might be articulated relative to the rear body. Hence, the following algorithm applies to each of the body parts individually — represented by the term 'robot body' used in the following.

The shape of the robot is well known. When the robot's body is parallel to the ground this knowledge allows to test whether a grid cell is under a robot's body — testing the grid coordinates x and z values eliminates data points which are outside of the robot's body shape. To take advantage of this simple test, the algorithm operates in the coordinate frame of the robot. In this coordinate frame the robot's body lies in the horizontal plane and can be represented by:

$$y = 0$$

where y represents the height axis [1].

Each grid cell is represented by a vector $\vec{c}_g$ in the global coordinate frame. In order to compute the vertical position of the robot's body above ground, the algorithm uses a grid cell at ground level $y = 0$. The algorithm transforms (translation to the robot centre, followed by a rotation around the centre) this global vector into the coordinate frame of the robot:

$$\vec{c}_r = T_r^g \vec{c}_g$$

where

[1]The same definition of coordinate system as the simulation applies, i.e. y-axis is the vertical axis.

$\vec{c}_g$ global coordinate

$\vec{c}_r$ coordinate in robot frame

T_r^g transformation matrix from global to robot frame

A normalised gravity vector

$$\vec{g} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$$

is also transformed into the robot frame, i.e. $\vec{g}_r$. As a result the following equation describes the shadow projection of the robot in the coordinate frame of the robot:

$$\vec{c}_r + \beta \vec{g}_r = \begin{pmatrix} x \\ 0 \\ z \end{pmatrix}$$

Parameter β defines the distance of the robot's body, which lies at $y = 0$ in the coordinate frame of the robot, to the grid cell representing the actual ground plane. Thus solving the equation for β leads to the actual height level of the robot's body.

Data points derived with this method only provide information about free space. The wheels touch the ground during most of the operation, but the robot is not safe from falling and jumping. Thus, the assumption that the robot always has a stable ground does not hold.

6.4 Customising the multi-level surface map

The limited perception of the robot demands a customised version of the MLS map. When using a laser range finder for perception, a large number of data points describe wide areas of the environment. The use of statically mounted infrared sensors allows a frequent but much smaller stream of data points. Therefore they only describe a limited area of the environment. In addition, sensors are rigidly attached to the mother robot, so that the field of perception is strongly limited by the current position and orientation of the robot. Therefore the map building process for the mother robot has to be incremental. To cope with an incremental process, this project adapts the multi surface map.

For a maximum gain of knowledge, data points provide information about free and occupied space. This reflects the idea of bounded height field maps as described in Chapter 2. Therefore, instead of using horizontal and vertical surface patches like the original MLS map approach does, each grid cell contains a set of **free space** blocks and **occupied space** blocks. Blocks have three main properties: centre height, expansion and a surface level. While centre height and the symmetric height expansion of the block around the centre height are the same for free and occupied space blocks, the surface level represents a bound on where the robot could be positioned (see Figure 6.3). For a free space blocks it is the 'bottom' of the region of free space and the surface level of an occupied space block is the 'top' of the region. The surface level is defined to be:

$$s_b = h_b + e_b$$

and for a free space blocks

$$s_b = h_b - e_b$$

where

s_b surface level of block

h_b centre height of block

e_b block expansion

The following sections describe the process of creating this customised MLS map from the local space representation (data points).

6.4.1 Three major steps to build the map from data points

The implementation aims at an online mapping algorithm, which provides the most accurate map at the current time. Processing the simulation on a single computing device sets memory constraints, ruling out a mapping algorithm which accumulates a large number of data points before processing them. Regularly reducing the overall number of data points is

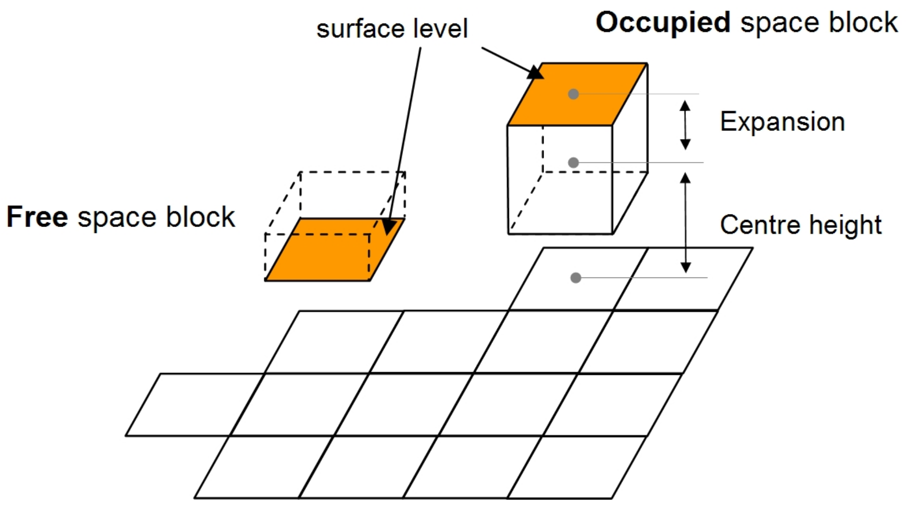


Figure 6.3: Occupied and free space blocks with properties

essential to deal with the memory constraint and avoid a loss of information. This requirement can be relaxed in a real application, where multiple computing units, i.e. one for each robot, are available.

The reduction process uses a model similar to the database scheme Insert-Update-Commit to compact data points frequently to generate 'true' measurements on the grid cell level. An update of the MLS map follows at larger intervals based on the mentioned measurement. The frequency of the update is determined by the map update frequency setting and a maximum threshold of measurements per cell.

Compacting data on the grid cell level

Due to the sensor model presented in Section 6.3.1, a grid cell may accumulate a number of data points per cell during a single control interval. The compacting of these measurement happens within one control loop and is based on a strategy which resembles the general database update strategy: insert, update and a final commit. For the insert step, data points from the local space are simply associated with a specific global grid cell. The update step compacts measurements of the same type, so that data points within a certain interval h_m are greedily merged into a single datum point — this project uses an interval of $h_m = 15$ cm based on the anticipated localisation error. The set of data points generated by the 'perceptual field' and hence also the merge data points have the following properties:

- height μ
- position variance σ^2
- belief p

In order to merge a set of data points this project uses the original MLS map approach [Tri07] and applies a Gaussian update. This update procedure allows to take the current position variance into account and computes the mean of the height plus the position variance from the set of data points:

$$\mu_{1:n} = \frac{\sigma_n^2 \mu_{1:n-1} + \sigma_{1:n-1}^2 z_n}{\sigma_{1:n-1}^2 + \sigma_n^2}$$

$$\sigma_{1:n}^2 = \frac{\sigma_{1:n-1}^2 \sigma_n^2}{\sigma_{1:n-1}^2 + \sigma_n^2}$$

where

z_n	current measurement to be integrated
σ_n^2	position variance of the measurement
μ_n	height of the measurement
n	current index of the measurement in the cluster

Position variance is also used in the way the original MLS map defines it, i.e. position uncertainty affecting all three dimensions equally.

The belief of occupancy is computed using Bayesian representation of probability as commonly done in occupancy grids (see Chapter 2), but in the current implementation data points of occupied space in an interval overrule all previous free space measurements. This is due to the nature of the previously outlined ray casting algorithm, which generates a greater number of free space measurements and might hide an obstacle otherwise.

The final outcome is a limited number of measurements per grid cell per control cycle — maximum of one per interval height h_m.

Transforming data points into blocks within grid cells

The next step transforms previously compacted data points into space blocks. This step is performed when an update of the map is requested or the number of compacted data points exceeds a threshold.

The process creates an occupied or free space block from each measurement. The resulting set of occupied and free space block is then further reduced by merging neighbouring blocks of the same type. The process finishes with a resolution step to solve issues with overlapping blocks of different types.

The conflict resolution process for overlapping blocks covers the following situations illustrated in Figure 6.4.

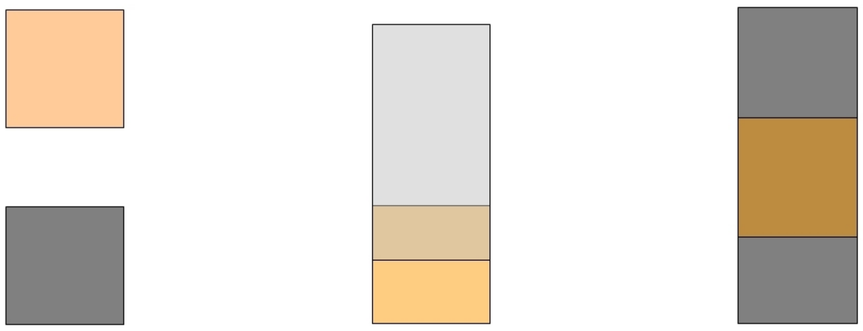

(a) No overlap (b) Partial overlap (c) Full overlap

Figure 6.4: Overlap situations

In the case of no overlap no action follows. Conflicting blocks resolve partial overlaps by adapting the centre height and expansion using the closest reasonable assumption, i.e. each block reduces the expansion in equal terms until no overlap exists. The situation of a total overlap, i.e. a free block is larger than a single occupied block and vice versa, is dominated by occupied space blocks. An occupied space block always replaces a free space block.

The transformation of data points is an important update step and reduces the amount of memory which would be needed to store data points. Because memory is constrained a general threshold m for the maximum number of measurements per cell exists.

Integration of new blocks into the existing grid cell data

The last step to create the custom MLS map integrates the newly created blocks into the grid. This is a merging process of new blocks into existing grid cell data and relies on the same strategy as described in the previous section.

In order to add a single block, the process searches for an existing block that is closer than a certain threshold. This threshold defines a lower bound for the robot which can only move through a gap of this height. This project uses a threshold of 0.7 m. If a block is close enough, the new block and the existing one are merged. Otherwise the new block is just added to the grid cell.

This process applies equally to occupied and free space blocks. Though free space blocks do not necessarily require this merging process equal treatment facilitates an implementation. The final mapping result is not affected because occupied space will dominate free space blocks.

6.5 Evaluation

The mapping process is a core element of ExploreTM. It has to deal with the large amount of incoming data efficiently to be applied as a real time mapping strategy. In addition to that, this project requires that the performance allows multi robots to run in parallel in simulation. The computing device for the simulation has a 3 GHz Dual Core Processor, 2 GB of RAM and a 256 MB graphic card. For the purposes of this project the absolute timing constraint is important to allow an operation with a control interval of 200 ms.

The evaluation is divided into three parts:

(1) validation of the perceptual field

(2) validation of map construction and evaluation of the final map

(3) performance evaluation

(1) The perceptual field (Section 6.3) has been verified with unit testing and generation of visual data as already shown in Figure 6.1. Part of this verification process was an alignment of the coordinate systems of simulation and ExploreTM, so that the sensor definitions within the perceptual field correspond to their placement in the model. The algorithm includes the orientation of the robot and the relative articulation of the rear body with respect to the front body into the computation. It thus allows normal and inverted operation of the robot.

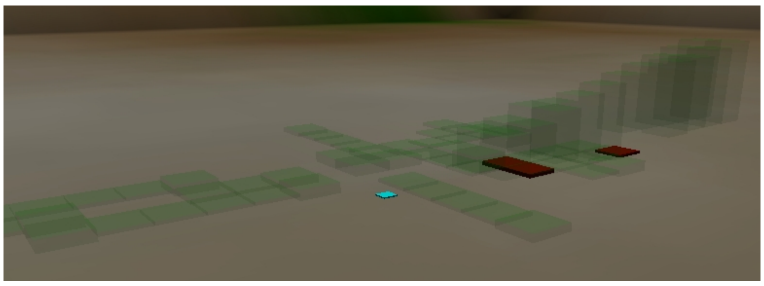


Figure 6.5: Occupied and free space block after robot standing still (at blue mark)

(2) The current implementation provides a representation of the world within the limits of resolution and a default localisation error which affects the height measurement. Figures 6.6 and 6.7 visualise the created **occupied** space blocks after mapping, where Figure 6.6 represents a mapped multi layered terrain. Figure 6.5 shows a map description resulting with **free** space blocks and illustrates the mapping results of a robot which has not moved.

The map overlays the obstacles so that the accuracy can be estimated. Figure 6.7 contains several spots where the map shows a larger obstacle than actually exists. This is the result of the localisation error combined with a missing upper bound given by free space blocks.

(3) The mapping algorithm uses various three dimensional operations to infer the global sensing point of each sensor. The current design reduces those to a minimum to achieve good performance. The final performance of the perceptual field is measured for a set of 26 sensors (14 have a maximum range of 1.2 m and 12 a maximum range of 3 m) and a resolution k of the sensor ray in between 0.005 m and 0.001 m (Section 6.3). Figure 6.8

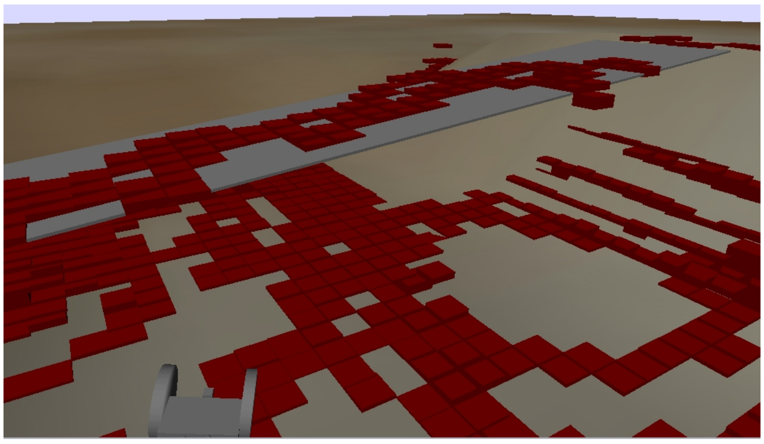


Figure 6.6: Visualisation of occupied blocks after mapping a bridge

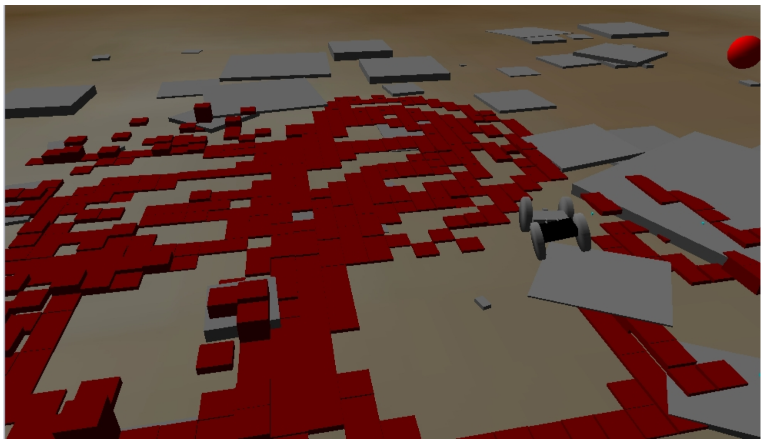


Figure 6.7: Visualisation of occupied space blocks after mapping a field of rubble

shows the relationship between the sensor resolution and the maximum number of data points produced for this configuration. The perceptual field produces the maximum number of data points for each selected sensor resolution after receiving only maximum sensor readings as input.

Figure 6.9 shows the time **performance** of the perceptual field. This performance is essential for all map representations and affects not only the MLS map. The performance of the perceptual unit is consistently below 7 ms and below 1 ms for $k > 0.025$. The performance improves for areas with higher obstacle densities, where sensors provide shorter readings.

Each individual mapping solution will add further costs to the 'perceptual field' performance. The remaining performance evaluations of

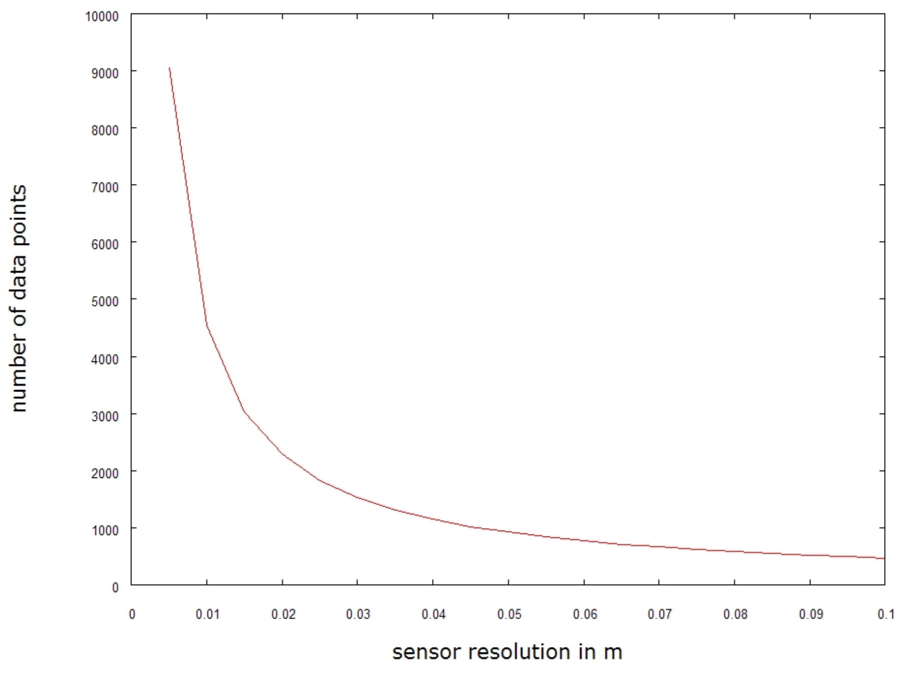


Figure 6.8: Number of data points depending on sensor resolution

mapping consider only the current MLS map implementation. To allow an integration of multiple grid based mapping algorithms into ExploreTM, the implementation uses a generic description of a grid and its cells. This generalisation increases the overhead for serialisation and decreases the overall performance, so that a less generic implementation could provide better performance results if needed. Therefore, the performance tests can be interpreted as an upper bound.

Two performance values are generated which represent the situations of travelling planar surfaces and a field of rubble. The first value is computed from the robot moving on a planar surface for 8000 simulated control cycles and sets an upper bound. The second value represents a lower bound and is produced from random sensor readings, so that fewer data points are generated (random sensor readings means that the rays are shorter on average, than in the planar case).

The resolution of a map in the current implementation can be chosen from a discrete set in the range of 1 mm^2 up to 1 m^2. Evaluations concentrate on 100 cm^2 and 625 cm^2 grid resolutions and show the tradeoff between accuracy and computing resources. This tradeoff has to consider

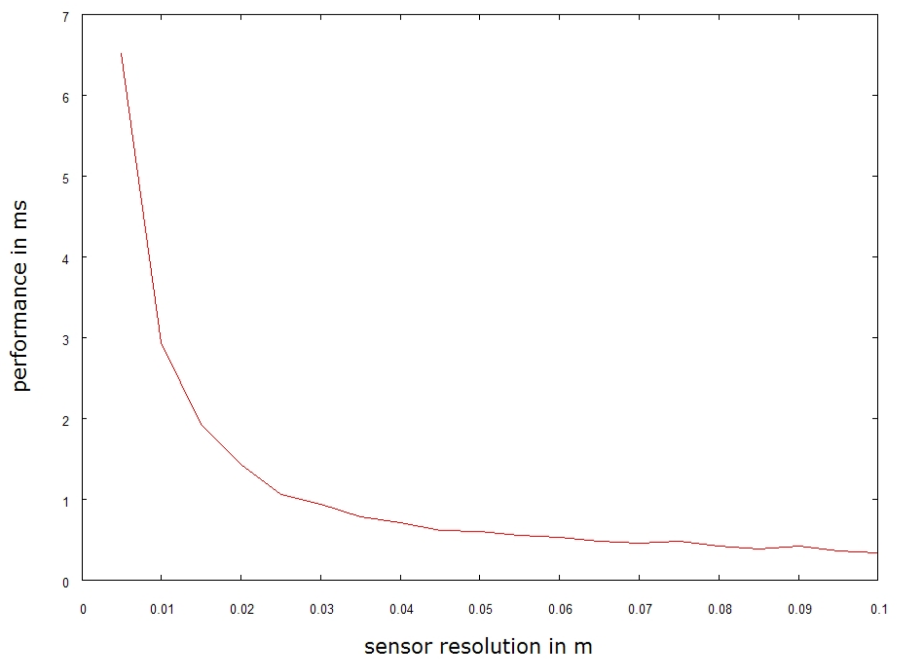
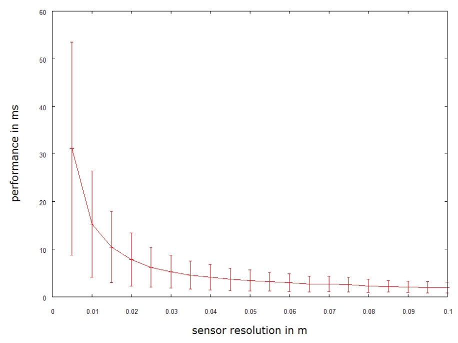


Figure 6.9: Performance of the 'perceptual field'

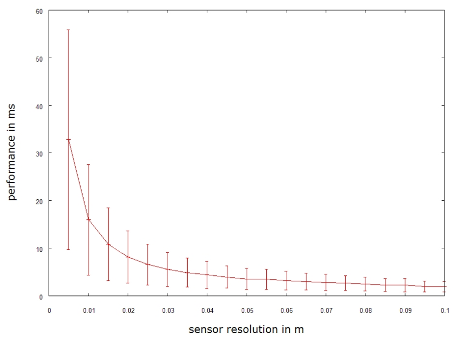
the performance of the map building process and the limitations arising for the transfer of partial maps, which will be sent from the mother robot to the grandmother robot.

Figures 6.11 and 6.10 show the performance of the mapping with 100 cm^2 and 625 cm^2 resolution. The compacted measurements stored in a cell are automatically transformed into space blocks when a threshold m of 30 or 100 is reached. The threshold m will only be reached in an application of ExploreTM when the map is not used for continuous planning operations. Thus, it provides a security measure in situations were the map is not updated in regular intervals. The evaluation shows that a lower threshold only shows a slightly better performance and hence, the threshold hardly has any influence on the performance of the current implementation.

The comparison between the sensor resolutions show that the higher resolution of 100 cm^2 has a small performance advantage for sensor ray resolutions below 0.02 m, but the 625 cm^2 map resolution shows a small advantage for sensor ray resolutions above this value. However, the differences are not significant and this project gives therefore a general recom-

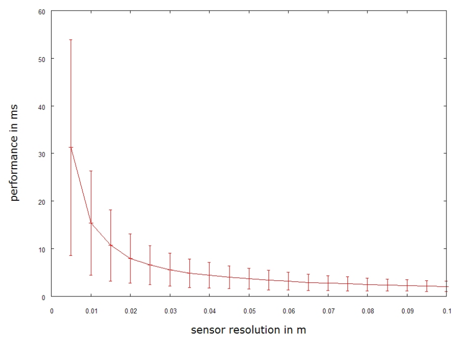


(a) Threshold= 30

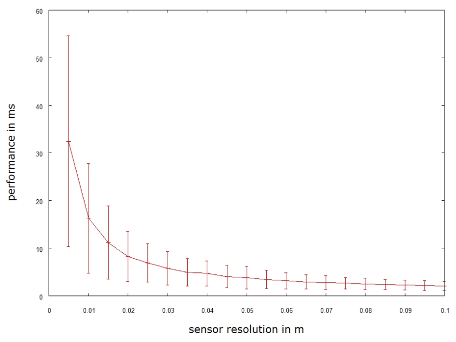


(b) Threshold= 100

Figure 6.10: Performance with variable sensor ray resolution and map resolution of 625 cm^2



(a) Threshold= 30



(b) Threshold= 100

Figure 6.11: Performance with variable sensor ray resolution and map resolution of 100 cm^2

mendation to use a sensor ray resolution of $k = 0.025$ m and a compacting threshold of $m = 30$, so that a general map building performance below 10 ms can be expected.

The generation of a partial map during the movement of the robot can be a frequent task - controlled by the tuning parameters of ExploreTM (Chapter 3). The parameters for the partial map are resolution (defined by the original map) and size (radius). This influences the performance of the simulation and can limit the number of robots which operate at the same time. To generate a partial map, the relevant grid cells are updated before the partial map is built, so that only the most recent data are integrated into this data structure. Figure 6.12 shows the results of a test, which runs outside of MRT to avoid any side effects. It simulates 32,000 control cycles (about 100 min of operation given a control frequency of 5 *Hz*), where 50% of the time the robot receives readings of a planar environment and otherwise random readings, and extracts a partial map from the final map.

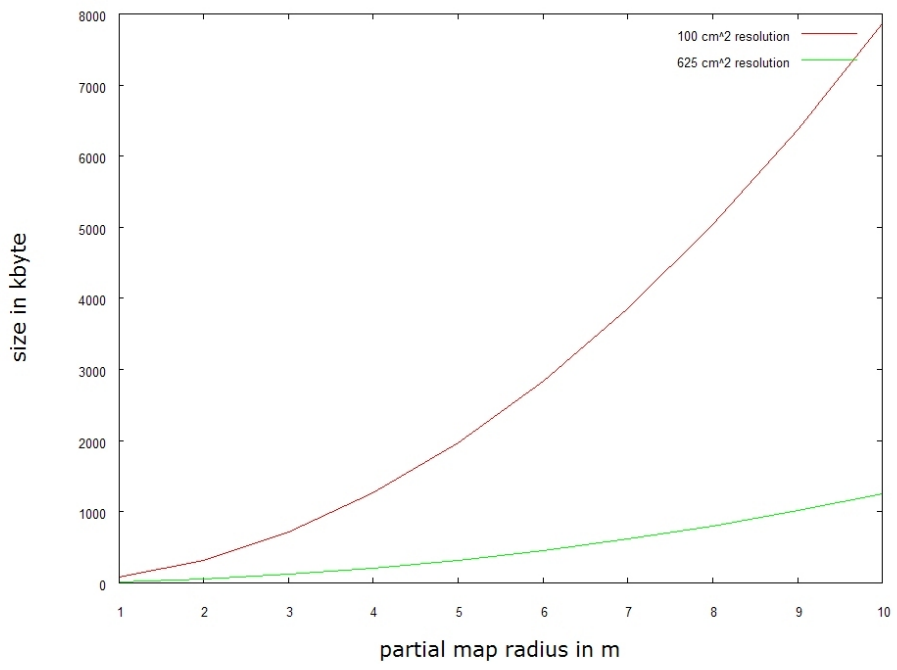


Figure 6.12: Size of the partial map in depending on the radius

Figures 6.13 and 6.14 illustrate the costs involved to compute the partial map and prepare for serialisation for two different map resolutions. It is not surprising that the increase of costs to generate maps with a higher resolution is non-linear. The analysis of the performance results suggests a selection of the low resolution in a multi robot simulation. The higher resolution can be used as well, but the configuration of the tuning parameters is far more restricted because the serialisation time is in the same range as the desired control interval. From this data the current project recommends the application of a 625 cm^2 resolution and a grid size of 250×250 cells.

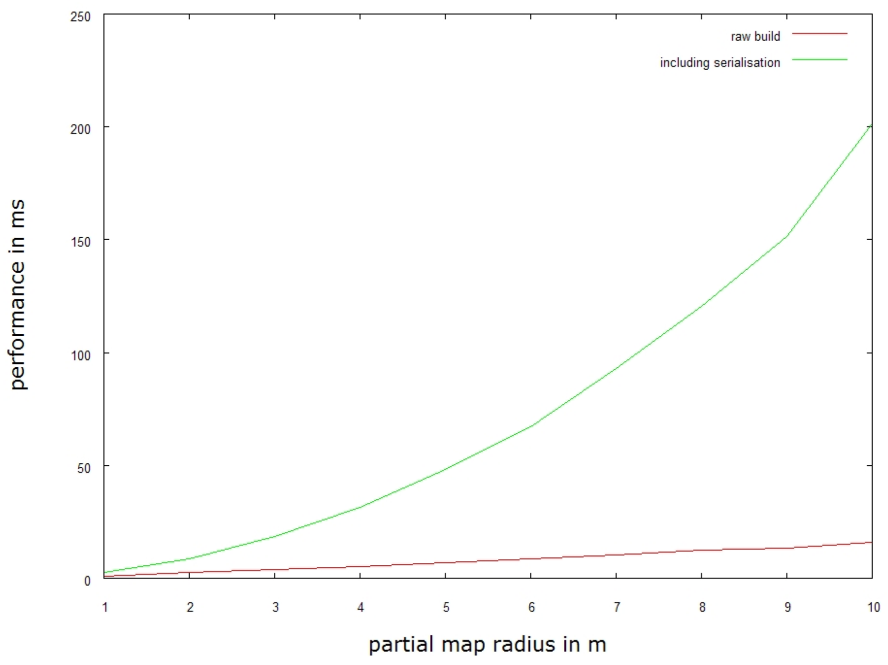


Figure 6.13: Performance of the partial map with resolution= 100 cm^2

6.6 Limitations

This project does not solve simultaneous localisation and mapping. Instead, the current approach uses an absolute position measure to build a map upon. The position of the robot is thus valid for a predefined control interval, so that over time a discrete set of positions is generated. The range finder produces measurements sequentially, i.e. the measurement takes place over a certain amount of time. Therefore the measurement

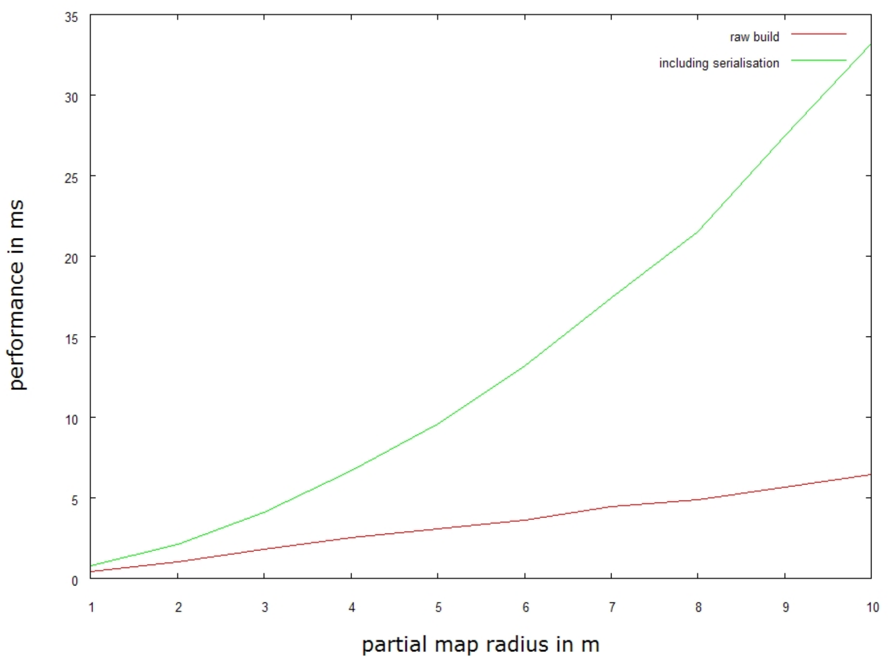


Figure 6.14: Performance of the partial map with resolution= 625 cm^2

result will be affected by any motion during the control interval. With the current assumption of a fixed position, mapping results with motions will suffer from distortions. Figure 6.15 shows the direct measurement results of a set of sensors, mounted in a horizontal plane, after performing a slow right turn to swipe the obstacle with the sensors. Figure 6.16 shows the measurement results after a subsequent quick left turn. The effect is a visible drag of measurement – a small one from the right turn, a larger from the fast left turn. This effect adds to any existing localisation error and increases - due to geometric properties - with the distance of a measurement point to the robot centre. It will be possible to account for these mapping errors in the context of controlled motion. However, in a cluttered environment containing drops, motion can consist of a number of fast, highly rotational and unpredictable motion patterns. Therefore multiple compensation techniques will be necessary to deal with this limitation.

The application of infrared sensors creates strong constraints on the mother robot's perception and is the main reason to customise the mapping algorithm. Thus, the algorithm tries to infer as much knowledge as possible from the range finder information. The simulation runs on a single

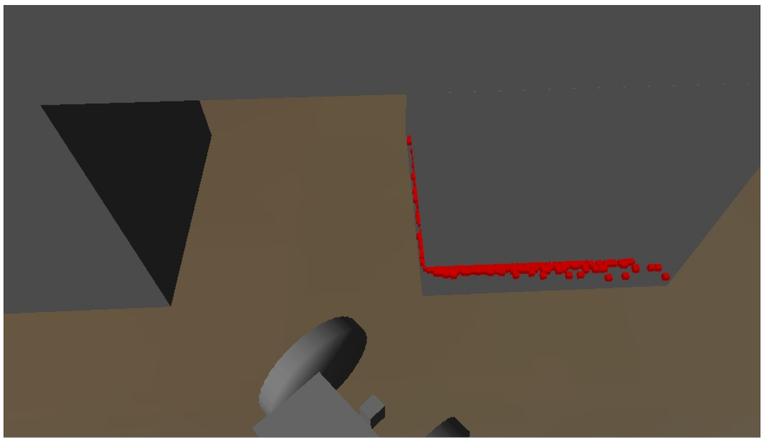


Figure 6.15: Generated measurements after a slow right turn

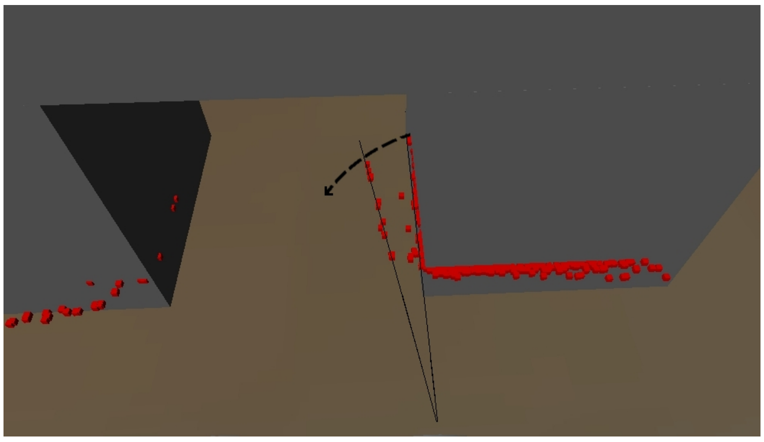


Figure 6.16: Generated measurements after mapping with a fast left turn

computer and the need for a multi robot simulation increases the memory requirement. For that reason, this mapping strategy can potentially used with less effort in a real application.

The mapping algorithm focusses on the representation of a 3D world by storing all collected information about free and occupied space. However, the handling of uncertainty of sensor measurements needs further evaluation and improvement. This also affects the merging of two different maps which is handled under the assumption that the localisation error is smaller than the resolution, so that the effects are ignored.

As with many other map representations and algorithms, the presented approach does not address the mapping of dynamic objects. This is a critical limitation in a multi robot scenario, where two robots could possibly map each other as part of the environment.

6.7 Possible solutions and extensions

Integration of uncertainty as an effect of a noisy environment is one of the main open issues. A solution to this problem starts with implementing the inverse sensor model to generate an occupancy belief of sensor measurements. This belief can then be part of the filter mechanisms. The implementation of a map merging process with uncertain localisation is a subsequent task and [TPB06] shows an approach.

In order to address the mapping of dynamic objects several options exist. Limiting the problem to the moving robots, a global knowledge of each robot's position can be used to define temporary non-mapping zones. Another solution is the extended application of occupancy belief of current and previous measurements, so that space blocks of the current map might change their type back from occupied back to unoccupied or be just invalidated.

6.8 Summary

A mapping algorithm for 3D has been implemented on the base of MLS maps and a core 'perceptual field' module. The current algorithm can deal with requirements which arise from a strongly incremental data collecting process. The implementation also allows a flexible usage of maps with different sizes and resolutions. The algorithm has been shown to be effective and fast enough to allow real time mapping. Therefore, the implementation not only provides a foundation for the operation and data integration for the mother robots, but also for the daughter robots.

Chapter 7

Path planning

Navigation in ExploreTM consists of four different elements: local reactive navigation and local path planning on the mother robots and global path planning and application of a global exploration strategy on the grandmother robot.

The grandmother robot combines the information from all exploring robots and enhances the overall search process, so that the team of robots can gather information more efficiently. Planning on the grandmother robot for the group of mother robots is intended to avoid overlaps of exploration activity due to the limited capability of the mapping process to consider dynamic objects. Therefore the grandmother robot directs the mother robots and suggests paths to reach new exploration areas. These paths should maximise the tradeoff between utility of exploration and travel risk until the robots reach their final exploration area.

Mother robots will mainly operate in previously unknown terrain, where reactive navigation has to deal with low level obstacle avoidance. The mother robots use the suggested path from the grandmother robot as guidance. But to immediately take advantage of collected information, local path planning complements the navigation capabilities of the mother robots.

Chapter 6 discussed the creation of a valid map of the environment. Path planning uses the information in this map, but has to build a graph structure from it first. This graph has to fulfil the following requirements:

1. represent uni-directionally traversable sections
2. include unexplored areas
3. support frequent (map) updates

Every new incoming measurement updates the map structure; to allow path planning with current data, the graph needs to reflect this change. Therefore this project has to cope with map updates at high frequency (5 Hz). Furthermore, the path planners frequently access the graph, so that the graph is a performance bottleneck for the planning algorithms. A tradeoff between the accuracy of the graph and performance can be made, but is not straight forward. The quality of the graph is essential to the path planning process, so that every limitation of the graph will also affect path planning.

7.1 Creating a graph from a given map

In general path planning can be done by directly interacting with a map built from an environment. However, for efficiency reasons path planning is often done on a graph. This project builds and uses a graph on all robots, grandmother and mother robot. But to support local path planning on the mother robots, frequent map updates have to considered and ExploreTM uses a tight coupling between the map and the graph data structure to achieve the required performance.

7.1.1 Basic considerations

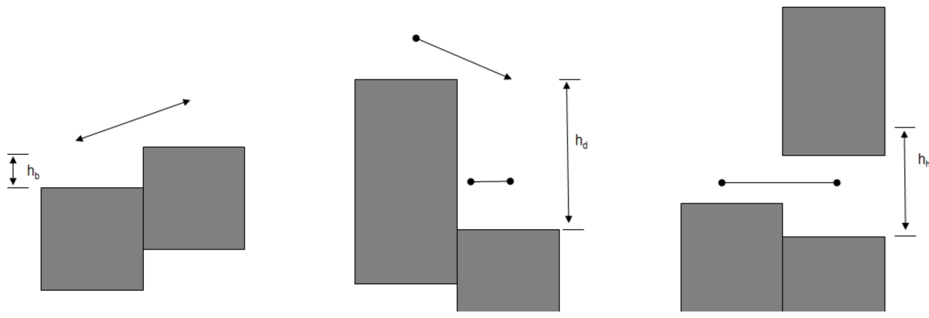
A MLS map has an underlying grid and (in this project) contains space blocks. A graph in ExploreTM, built from a MLS map, consists of 'graph nodes' and 'graph node buckets'. There is a 'graph node bucket' corresponding to each grid cell and a 'graph node' corresponding to each surface patch of an occupied space block (for cells with no occupied space block, the surface of the lowest free space block is used). Each graph node contains a link to its 'bucket', the corresponding surface patch, a traversability measure and connections to its neighbour surface patches. These connections to neighbours depend on movement constraints of the mother robot. The constraints are illustrated and classified in Figure 7.1. The graph building process is the only module in ExploreTM which evaluates the relationship between neighbouring patches. It determines whether a connection should exist and classifies the connection as:

ClimbDrop: Bidirectional traversal is possible. The robot can manage to climb up to a certain height, here $h_b = 0.2$ m

DropOnly: Unidirectional traversal only. The robot can manage drops up to a height h_d without damage ($h_d = 2$ m estimated).

Blocked: Due to the total size of the robot it can only pass through holes of a certain height h_h, here $h_h = 0.7$ m, or climb up a certain height h_b.

To take advantage of this classification in a later stage (refer Section 7.4.1), the graph building process stores the classification with each connection. Beside this information, a connection also contains a reference to its start and end node, plus an absolute cost measure (refer Section 7.4).



(a) ClimbDrop (b) DropOnly & Blocked (c) Blocked

Figure 7.1: Connection types between surface patches

The graph building process considers complex constraints such as the evaluation of the situation shown in Figure 7.1c, so that a path planner can work more efficiently. Exceeding a maximum drop height of a connection classified as 'DropOnly' could result in damage for the robot and sets a threshold to change the connection classification to 'Blocked'. Nonetheless, this project does not include this threshold into the graph building process. The traversability measure of a graph node (Section 7.4.1) will include this threshold because it is a less complex constraint. This measurement allows a more flexible (and dynamic) adaptation of the drop threshold.

The graph building process does not incorporate the full description of the mother robot's kinematic constraints. It is limited by the resolution of the map and does not take into account the robot's footprint. Hence, path planners have to include a cost measure which considers the robot's footprint size.

For the final construction of the graph, the map structure has further properties that must be represented:

1. Neighbouring *grid cells* can have multiple connections between surface patches.
2. Each *surface patch* can have only one outgoing connection for each of the possible eight neighbour directions[1].
3. Unexplored *grid cells*

The final result of the graph build process is a directed labeled graph, where the label on an edge includes a classification of the connection.

7.1.2 Updating

The implementation of ExploreTM takes advantage of the close connection between a map and its corresponding graph. Requesting an update of the graph will trigger an update of the underlying map first. The map will update each grid cell that contains new data and then the graph will update only the graph node buckets that correspond to an updated grid cell. For each of these graph node buckets, the graph nodes will be reconstructed, as well as their links to any graph nodes in the neighbourhood.

[1]Since the robot cannot travel to two different levels at the same time, it does not make sense to have more than one connection.

Optimistic planning

Chapter 6 described the creation of occupied and free space blocks. In both cases, the perception of the mother robots is too limited to provide an exact height measure. A free space block provides an upper bound for the surface level, while an occupied space block provides a lower bound. In order to deal with this limitation the planner uses an optimistic planning approach. The graph building process evaluates each grid cell of the MLS map and uses the surface level descriptions of the occupied space blocks, if at least one such block exists. If a grid cell contains only a free space block, the construction of the graph assumes the surface level of the free space block is traversable. This bears the risk that the surface level does not exist and that there is a drop instead. Thus, this approach has to be called optimistic as long as the perception of the mother robots is not enhanced significantly.

7.2 Selection of an adequate path planner

Chapter 2 showed that A* and D* Lite are both good algorithms for path planning and both path planners will find the currently best path to a goal. The performance properties of A* and the easy integration of different cost measures mean that it is a good general path planner. However, D* Lite can reuse information generated in previous searches as long as the goal remains the same. This allows D* Lite to outperform A* over subsequent searches [KL02]. Nevertheless, A* is less complex and requires less expensive computation and comparisons: both planners use keys to order their priority queue, but a key in D* Lite is based on two values, while a A* uses a single value. Therefore A* can outperform D* Lite on searches where the path planner constantly plans to new goals.

The design of ExploreTM design does not use a constant replanning process on the grandmother robot. The grandmother robot computes a new mission set and suggests paths only on request. The newly generated set of missions for the robots will also greatly differ from any previous set, so that starting locations of the robots and the assigned target locations will have changed. Thus, for the general mission planning on the grandmother robot, the path planner operates infrequently and the map can be treated as static (though it will be updated in between planning by new information from mother robots). Therefore ExploreTM uses A* to plan paths on the grandmother robot.

Mother robots frequently update their maps because they receive new data at every control interval. Planning within a control interval would be ideal but would require a fast path planner. Because each mother robot has a goal point set by the grandmother robot or its local exploration strategy (see Chapter 9) it will replan to the same goal until the goal is reached. Therefore D* Lite is used in the mother robots.

7.3 Performance optimisations

The implementation of A* and D* Lite is initially straightforward, but requires optimisation to achieve good performance. The implementation of both algorithms is based on similar properties, so that this project could optimise both path planners in a similar way. The number of search nodes can be huge and requires an efficient priority queue with an efficient modify operation. No memory management is implemented in ExploreTM, so that the size of the search nodes affects the performance (due to necessary memory allocation). The priority queue uses the standard heap implementation of a complete, partially ordered, binary tree with enqueue, dequeue and also a modify priority operation. To implement the modify operation efficiently, each node includes a reference to its current index within the heap. The modify operation can therefore address the node with constant costs.

The search operates with two data structures: the graph node and a 'search node' associated with this graph node. To reduce size each search node contains only a reference to its associated graph node and its index within the priority queue. The referenced graph node stores search specific properties.

As a result of the optimisation the implementation guarantees the following operation costs:

1. Access of a search node property in $O(1)$

2. A priority queue with enqueue in $O(\log n)$, dequeue in $O(\log n)$ and modify in maximum $O(\log n)$

Lazy initialisation

Because graph nodes store search information (for efficiency), each graph node has to be initialised with this information at the beginning of each search. For efficiency, initialisation takes place for nodes only when the algorithm needs them. By generating a unique identifier for each search, stored in the nodes used in the current search, the algorithm is able to detect if a node needs reinitialisation.

D* Lite specific update

Compared to A*, the D* Lite implementation introduces more complexity and the implementation of an update mechanism is essential for the D* Lite algorithm. Nodes which have changed need to be invalidated or better still be updated to allow a propagation of changes. ExploreTM therefore maintains a queue of graph nodes which have been updated. This queue has to be processed before each run of the D* Lite algorithm.

The path planner will run at every control cycle to enhance the navigation of the mother robot. Fortunately, this characteristic limits the possible number of grid cells which require an update. In addition, the number of nodes within the grid cells depends on the environment but a maximum average of three is expected. This expectation is founded on the kinematic constraints of the mother robot and the required passage height h_h, so that three nodes will cover a minimum height of $3 \cdot h_h = 2.1$ m. That finally means that an expected upper performance bound for the update of cells can be determined.

7.4 Cost measures

Path planning in this project requires planning through known and unknown terrain and includes partial information for planning through unexplored environments. The heuristic cost measure applied in this project will be based on Euclidean distance, whereas the final path depends on a variety of parameters described in Chapter 9. One of these parameters to mention here is traversability. The path finder mission has shown that an embedded traversability map [MBT+06] can enhance navigation. Inspired by these kind of projects, traversability in this project will be used to define the costs of traversing sections and is an important measure for computing traversable paths. Traversability is closely related to the definition of the graph structure. This measure builds the foundation for the cost measure in path planning and will be explained in the following subsection.

7.4.1 Traversability

The traversability factor, as already mentioned in Chapter 2, is a value between 0 and 1, where a lower value means less traversable terrain. To classify terrain into different degrees of traversability, the graph representation contains height information about nodes and their neighbourhood.

To integrate the idea of traversability this project implements a 'traversability analyser' [USN03]. The analyser uses a single node of the graph and computes an estimation of traversability for the given node, based on its connections to neighbour nodes. The implementation applies a traversability definition mainly derived from [JSPB07] and describes a patch p with the two parameters slope $\tau_s(p)$ and roughness $\tau_r(p)$. It also integrates a threshold for drops, so that traversability is set to zero once a drop exceeds the threshold. The current threshold is 2 m and is based on the maximum estimated drop height a robot can traverse without damage. The measure of roughness uses the squared height difference between neighbouring nodes:

$$\tau_r(p) = \max\left(0, 1 - \frac{1}{N \cdot H^2}\sum_{i=1}^{N}(\Delta h_i(p))^2\right)$$

where

$\Delta h_i(p)$	height difference of the i^{th} outgoing connection of patch p
H^2	maximum possible squared height difference, i.e. the squared drop threshold
N	maximum number of outgoing connections

The traversability analyser also computes the slope parameter (different to the original approach) by combining the number of precomputed edge classifications for ClimbDrop ($\mathrm{n}_{ClimbDrop}$), DropOnly (n_{Drop}) and unexplored connections ($\mathrm{n}_{Unexplored}$). It considers the maximum number of possible outgoing connections $n_{max} = 8$. Although not a computation of slope in the original sense the classifications represent a fuzzy description of the slope of an edge. Classification are combined into a single measure using the following equation:

$$\tau_s(p) = \frac{\mathrm{n}_{ClimbDrop} + 0.5 \cdot \mathrm{n}_{Unexplored} + 0.2 \cdot \mathrm{n}_{Drop}}{\mathrm{n}_{max}}$$

The weighting factors have been empirically set and are derived from the fact that any unidirectional terrain reduces traversability and missing knowledge contributes to a normalised value of $\tau_s p = 0.5$.

The final traversability value is $\tau(p) = \tau_s(p) \cdot \tau_r(p)$. Unexplored nodes will have a constant traversability value of 0.5. Figure 7.2 shows the traversability of surface patches around an obstacle. Figure 7.3 shows the

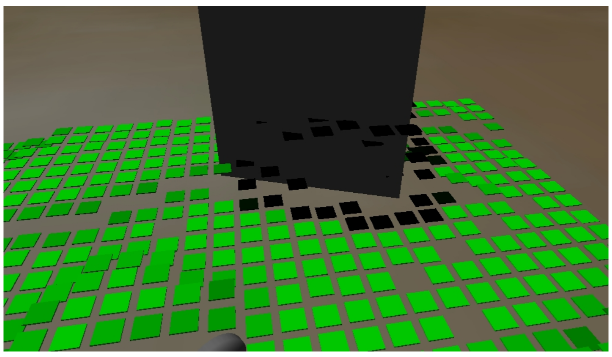


Figure 7.2: Traversability map - darker green values represent lower traversability

(mostly) explored surface patches in a planar environment, which are part of the graph. Note, that most of the patches with higher surface levels in the planar areas represent surface patches from free space blocks and represent areas which the robot has swiped with sensors, but not traversed. Figure 7.4 illustrates the traversability analysis for a path under a bridge.

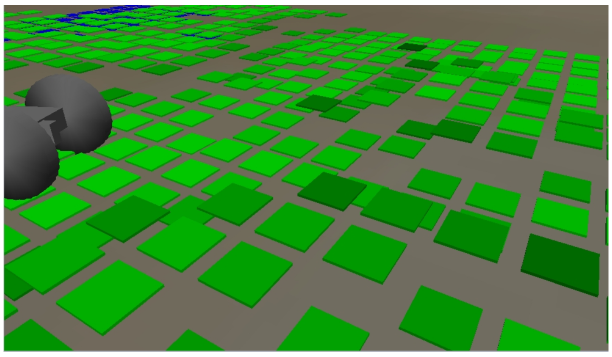


Figure 7.3: Traversability on planar surface

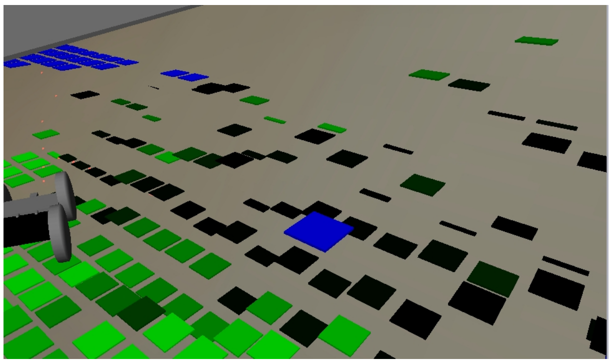


Figure 7.4: Traversability on inclined surfaces - blue patches are not explored

7.4.2 Dealing with unexplored nodes

To allow planning through unexplored terrain with the given graph structure, assumptions have to be made for connections between explored and unexplored regions.

Travel from unexplored regions into known terrain opens up different options — at least when a multitude of surface levels exists. In that case the unexplored node at the boundary has as many incoming connections as neighbouring nodes in a specific direction. Creating all of those connections will violate the constraint of one connection per direction. To keep this constraint for unexplored regions, this project assumes that coming from an unexplored region the travel is most likely to be continued on the highest neighbouring surface patch[2]. Connections from neighbour

[2]This project expects the highest neighbouring surface to be the main travel surface

nodes to the unexplored node do not violate the constraint, so that connections can be established in the standard way by assuming that no height difference exists to the exploration node.

7.4.3 Cost measure

Traversability is a major element of the costs to traverse a node. However, path planning can combine objectives, e.g. the traversal of a node can be rewarded under certain circumstances. This allows the maximisation of the information gain of every robot action. Chapter 9 will therefore introduce factors which influence the exploration task. Path planning uses the definition of traversal costs from one node to another combined with a single normalised influence value $i(n)$. This integration allows path planning to be influenced by additional factors of travelling between two nodes, although traversability remains the most significant factor. The cost measure is currently calculated as follows:

$$c(n_s, n_t) = \begin{cases} \varsigma\,(2 - i(n_s) - i(n_t))\,\frac{d(n_s,n_t)}{\tau(n_s)} & \text{if } \tau(n_s) \neq 0, \\ +\infty & \text{otherwise} \end{cases}$$

where

n_s, n_t	start and target node
$i(x)$	normalised influence value of node x
$d(x, y)$	Euclidean distance between nodes x and y (3D)
$\tau(x)$	traversability of node x
ς	sensitivity constant, currently ς=100

Path planning costs must depend on both the effort required to move from the current position to a target position, and the value of the move. The effort depends on the Euclidean distance between the positions and the traversability of the current position — clearly, a lower traversability has to increase path costs. The influence measure represents the value of traversing a specific node; the cost measure therefore includes the influence values of the current and target positions. Because of the normalisation of influence values, the cost function includes a sensitivity constant to allow tuning of the contribution of the influence values.

within a disaster site.

7.5 Evaluation

The evaluation of the graph building process and path planning is based on two steps:

(1) time performance evaluation of graph building process and path planning with A* and D* Lite

(2) validation of the cost measures

(Test 1) A unit test simulated the motion of the mother robot over 1600 control intervals (>5 min operation time) in a planar environment and let it constantly plan from the current position to a fixed target point. The average path length was 7 m. While the A* path planner took an average of 76.5 ms the D* Lite path planner required only 21.5 ms for the same task. These times include initialisation, search and path building process. The evaluation shows that the path planner implementation fulfils the anticipated performance characteristics [Ste95].

Planning on the grandmother robot is not time critical, i.e. it does not need to plan paths for the group of mother robots within a single control interval. However, the amount of time spent should be reasonable, i.e. within few seconds. This total time is influenced by the number of robots *n* the grandmother has to plan for, but will be mainly linear as a result of performing A* search *n* times (Chapter 9).

Computation time is limited on the mother robots. Applying the D* Lite planner within a single control interval required performance optimisation of the graph build process to allow a frequent graph update. Figures 7.5 and 7.6 show the performance results for updating the graph during 6400 simulated control cycles with the robot moving in a 10×10 m^2 field while receiving random sensor measurements. The map update is limited around the robots position by radius *r*, so that the algorithm touches as few cells as necessary. The performance heavily depends on the selected resolution, though both tested map building processes – with 100 cm^2 and 625 cm^2 resolution – operate within the given performance constraints. However, for a multi robot simulation this project recommends a resolution of 625 cm^2.

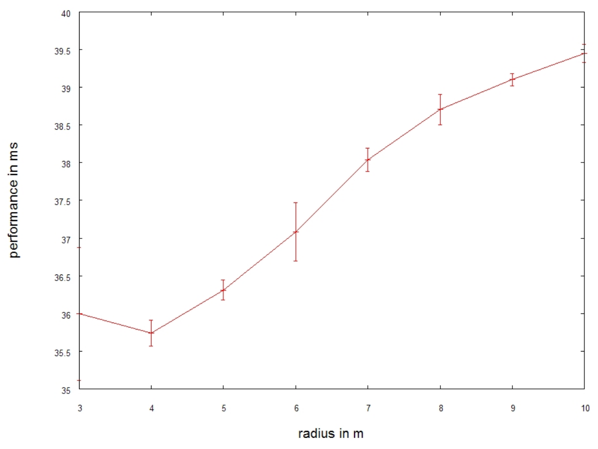


Figure 7.5: Graph update for 100 cm^2 resolution

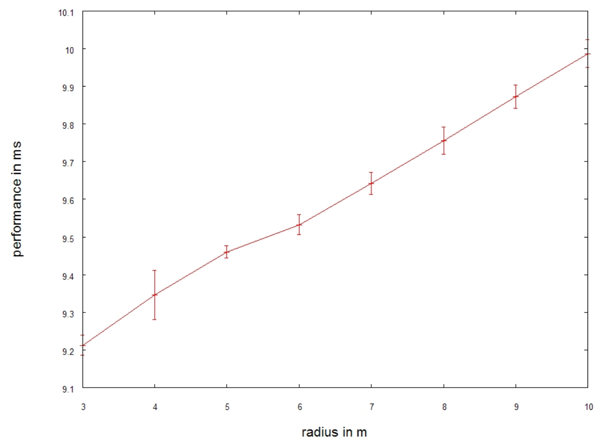
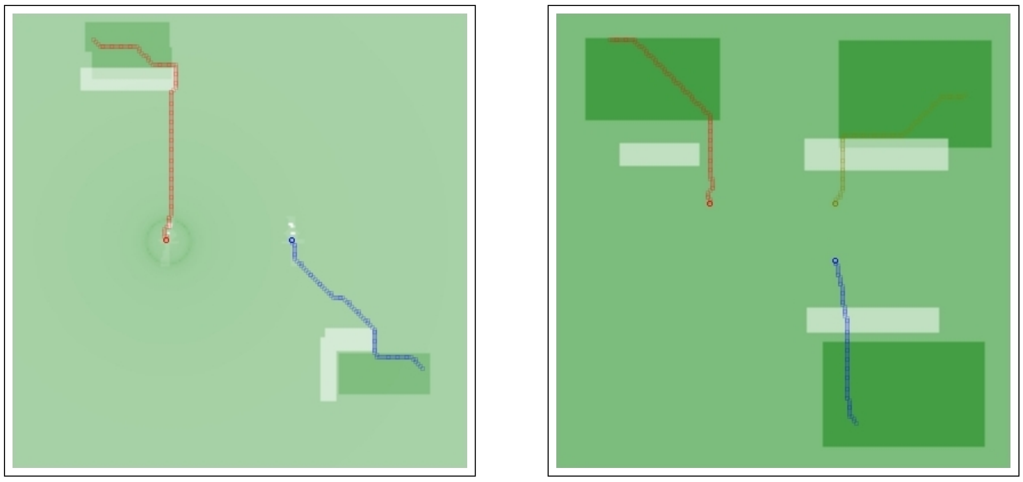


Figure 7.6: Graph update for 625 cm^2 resolution

(Test 2) The grandmother path planning can be visualised through the operator interface (see Chapter 9). Figure 7.7a illustrates the outcome of path planning for two robots to assigned regions of interest. Areas of higher interest are marked with a darker green and areas of high risk tend to white. The two planned paths avoid the risk regions, but due to the properties of the path plan the path is still close to the risk region. Lowering the sensitivity value ς causes the planner to ignore the influences, which results in planning through risk areas as illustrated in Figure 7.7b. Hence, the sensitivity factor influences the maximum region of risk areas which can be traversed.

Figure 7.8 illustrates the effects of a subsequent increase of the risk



(a) Sensitivity $\varsigma = 100$ (b) Sensitivity $\varsigma = 50$

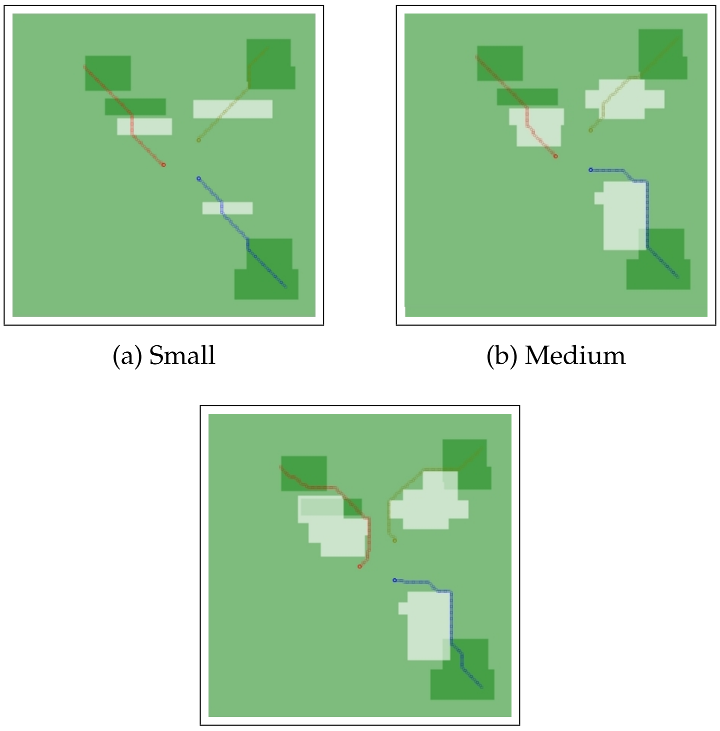
Figure 7.7: Grandmother robot planning for multiple mother robots in the field

area size, with a constant sensitivity factor $\varsigma = 50$. A modulation of this sensitivity value might be incorporated to allow a more flexible path planning, but is not part of this project.

The D* Lite path planning is illustrated in Figure 7.9, where the planner computes a path over the edge of an abyss (red dotted line in Figure 7.9a). This points again to the weaknesses of perception which also affect path planning. The robot cannot detect the drop properly, so that the computation of traversability and subsequent path planning cannot provide any advantage. But path planning still adds value to navigation through cluttered areas (Figure 7.9b).

The application of a traversability measure also represent the foundation to detect enclosures (Figure 7.10) — one of the challenges of USAR as identified in Chapter 4.

Current parameter setting This chapter introduced a traversability measure to allow path generation. This traversability measure is applied in combination with the normalised influence value to control the path costs. While the main influence of the traversability measure has been shown in Section 7.4.1, it is also included in the single influence value. However, for this evaluation a weighted sum was used to compute the single value from


(a) Small (b) Medium

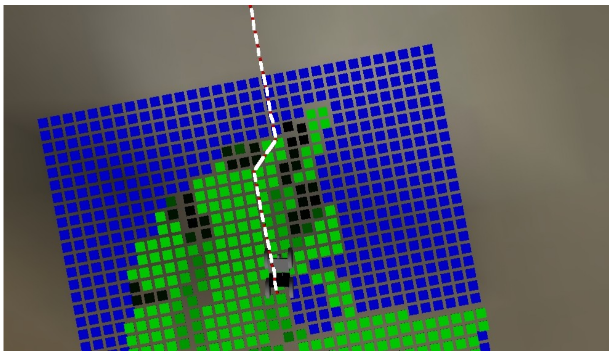
(c) Large

Figure 7.8: Sensitivity=50 with changing sizes of risk areas

the individual influences and traversability has a low associated weight as shown in Table 7.1. These weight parameters have been set empirically and need more evaluation for future research.

influence	**weight**
traversability	0.1
interest	0.5
exploration status	1
general risk	0.9

Table 7.1: Weight setting to calculate numeric influence


(a) Planning at an abyss - red dots visualise the path

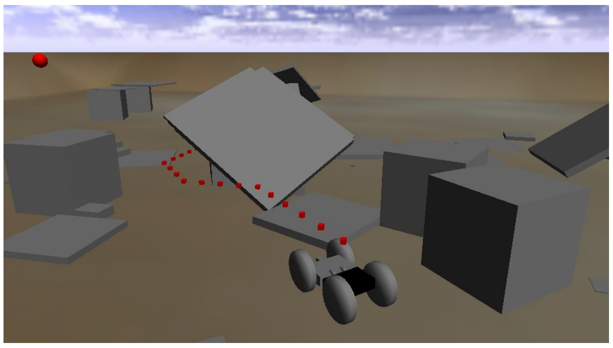
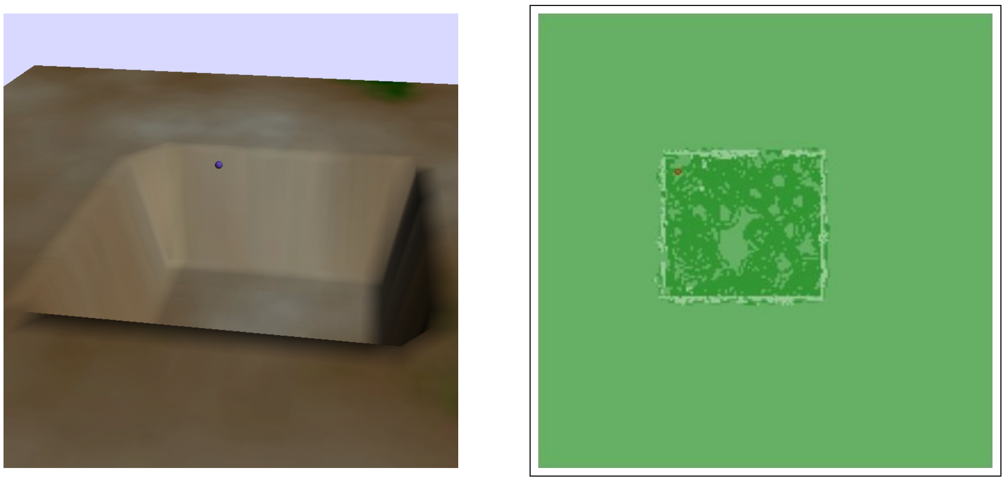

(b) Planning through densely cluttered areas

Figure 7.9: D* Lite path planning

7.6 Limitations

This chapter described a planning approach which uses an optimistic free space assumption and contains a general travel risk, which is not eliminated by path planning. Nevertheless, this does not change the properties for path planning, which could be used in the same way once the perception of the robot is enhanced and surface levels can be detected with greater reliability.

This project customises the graph to the current configuration of the mother robots by using a number of thresholds reflecting the kinematic constraints. To generalise the path planner for different types of robots, the graph building process would need to remove the robot specific thresholds. But this could only be done by applying different cost measures for each robot, which would introduce more complex computations of travel costs and decrease the performance of the path planners.



(a) Enclosure (b) Traversability map

Figure 7.10: Traversability of an enclosure

The implemented graph update is disruptive by releasing all connections of nodes within a grid cell (for a MLS map) which receives an update. This increases the number of graph nodes which need an update before path planning with D* Lite can be done. This update method currently offers a performance within the requirements, but other implementations can reevaluate this design to increase performance if necessary.

Without using any methods for obstacle enlargement or region growing the path planner will tend to take paths close to any risk regions or obstacles (see Figure 7.7a). A possible countermeasure is a filter mechanism to propagate low traversability values within the graph. However, this requires a further preprocessing step before path planning and might also affect the planning through narrow passages. Alternatively to a filter, the definition of a region can be easily done by an operator (Chapter 9) by including a 'keep out' area in the definition of risk regions.

This project makes an assumption when planning paths from unexplored terrain to explored terrain. This does not have a major impact on the planning in the current project, because the test environments are not heavily layered. Nevertheless, this assumption will need reevaluation as soon as a similar path planning strategy has to be applied to the daughter robots which are intended to operate in a more layered terrain. The presented limitation is also related to the constraint of directing the robots to a specific point in the 3D map.

7.7 Possible solutions and extensions

Tuning and optimisation of the path planners can be a goal of future projects. There are already a number of ways to optimise D* Lite listed in [KL02], which have not been applied yet in order to keep the complexity of the implementation low and because the performance improvement could not be estimated.

One improvement of the path planner would involve a more sophisticated environment analysis of the given map. Instead of relying only on a cost measure and traversability value, the setup of a hazard detection with a more detailed definition of keep out area for the robot could be considered.

In combination with the environment analysis or as a separate extension, more flexible path planning should be enabled. A dynamic change of the cost measure would allow an expansion of the risk measure. Requesting a path could incorporate a fuzzy degree of risk taking in planning. Such planning can range from complete avoidance of unexplored and unsafe areas to a full risk taking approach, which ignores any threshold for drops in the environment. The following degrees of risk in combination with exploration serve as an example classification:

(A) no loss of a device is tolerable, i.e. the robot should get as quickly and as securely as possible to the target destination

(B) the robot should get to the goal in a bounded time and switch to exploration of type (A) after a timeout. Exploration activity is rather conservative and safe.

(C) A robot explores as much as possible on the way to the target, i.e. the planned path leads through mainly unexplored areas: this path-planning approach gives very low priority to reaching the target destination safely.

7.8 Summary

This chapter presented the essential elements of path planning in ExploreTM. It discussed the necessary generation of a graph data structure and explained its properties that represent kinematic constraints of the robots. This project selected path planners for planning on the grandmother and mother robot and optimised them to fulfil the performance requirements. In addition a cost measure based on influence and traversability was introduced, which can be modulated through a sensitivity factor.

Chapter 8

Navigation

The mother robots need to be able to navigate safely in a 3D environment to perform exploration properly. This chapter describes the challenges of mother robot navigation in 3D environments and an approach to solving them. This project builds on a novel navigation approach described by Lee-Johnson [LJ08][LJC07], further referred to as 'EmotioNav', which uses emotion based parameter modulation. There are special properties of the mother robot and the 3D environment which mean that EmotioNav does not work unmodified. This chapter describes modifications to EmotioNav to overcome these problems.

8.1 EmotioNav

Local path planning on the mother robot incorporates the current research efforts of EmotioNav. Navigation in EmotioNav is based on the combination of the well known VFH+ and the dynamic window approach (see Chapter 2), which serve as a directional and a velocity controller respectively. VFH+ is applied to evaluate a number of candidate directions which the robot can follow and is thus the directional controller of EmotioNav. The output of the directional controller is a recommended direction for the robot. The dynamic window approach serves as velocity controller and if possible follows the recommendation of the directional controller. It therefore evaluates a set of curvatures — pairs of linear and angular velocities. This section first describes the main elements of EmotioNav. This description is followed by the requirements for change and a presentation of the modifications introduced in this project.

8.1.1 Calculating an objective value

Both controllers of EmotioNav rely on the maximisation of a multiplicative objective function which combines different influencing factors. This objective function is similar to the inverse cost function of the VFH+ approach and allows the evaluation of candidate directions or velocities. Each influencing factor is represented by a normalised value and also has an associated weight in $[0, 1]$. The combination of the influencing factors is based on the following equation:

$$O = \prod_{i=0}^{n} (1 - w_i(1 - o_i))$$

where

O	total objective value
o_i	individual objective value
w_i	weight associated with an individual objective

Weights for this objective function are typically chosen between 0 and 0.99, so that all objectives can influence a decision. Nevertheless, a weight w_n can be set to 1 when needed, so that an individual objective o_n can lead to a total objective value of zero. Compared to the weighted sum function used in the VFH approach the author of EmotionNav claims that such function prevents a situation where multiple but lesser weighted objectives can overrule a single, though higher weighted objective value.

Objective functions of EmotioNav use angular differences of current and possible directions. Such angular differences are adjusted to the smallest angular difference according to the unit circle, so that they are bound to the interval of $[-\pi, +\pi]$.

8.1.2 Directional controller

The directional controller of EmotioNav is based on VFH+ but instead of using an occupancy grid to compute its vector field histogram it uses sensed obstacles distances. To allow the point size treatment of the robot, EmotioNav incorporates obstacle enlargement of VFH+, though followed by an additional low pass filter. It then follows the suggestions of Ulrich *et. al.* [UB98] and includes a greater variety of objectives:

1. avoidance
2. goal seeking
3. angular inertia
4. wander
5. path following

Avoidance Obstacle avoidance is the most important aspect of navigation. EmotioNav computes an objective value for a candidate direction θ based on the following equation:

$$a(\theta) = \begin{cases} \frac{d_0(\theta)}{d_{max}} & \text{if } d_0(\theta) < d_{max} \\ 1 & \textit{otherwise} \end{cases}$$

where

$d_0(\theta)$	distance to closest obstacle in direction θ
d_{max}	maximum sensing range

Goal seeking Goal oriented navigation favours directions that are close to the target direction and penalises others.

$$g(\theta) = 1 - \frac{|\theta - \theta_g|}{\pi}$$

where θ_g is the goal direction.

Angular inertia The objective of angular inertia favours directions close to the current one. Significant changes of the travel direction without any change of other objectives are therefore prevented, so that the robot will operate on a smooth path.

$$i(\theta) = 1 - \frac{|\theta - \theta_c|}{\pi}$$

where θ_c is the current direction.

Wander The 'wander' objective allows the robot to navigate in a random direction, helps to escape local minima, and supports an explorative behaviour of a robot. At every time interval t_r=15 s a random direction is generated. To keep the focus on the goal the random direction is computed from a limited angle interval centred around the goal direction:

$$w(\theta) = 1 - \frac{|\theta - \theta_w|}{\pi}$$

where θ_w is sample from a uniform distribution in the interval $[\theta_g - \alpha\pi, \theta_g + \alpha\pi]$, with α as a parameter between 0 and 1.

Path following Global knowledge can enhance navigation by suggesting a path to follow. The incorporation of this path into navigation is addressed by the 'path following' objective. EmotioNav first searches for the closest node of the suggested path. Once this closest node has been found a target node is computed. This target node lies a distance d_{pf} from the closest node along the path towards the goal. Path following steers the robot to the computed target node and computes a separate path following direction θ_{pf} for that purpose.

$$pf(\theta) = 1 - \frac{|\theta - \theta_{pf}|}{\pi}$$

8.1.3 Velocity controller

The velocity controller of EmotioNav uses the Dynamic Window Approach and selects its final linear and angular velocity pair (v, ω) from a set of candidates. The evaluation of these candidates relies on various objectives. In general the velocity controller takes a candidate velocity pair and predicts the motion of the robot over a time frame — typically less than two control intervals. The controller analyses the predicted path by computing the minimum distance to any detected obstacle. This process eliminates candidate velocities which lead into a collision from the further process and ranks the rest based on various objectives.

Part of the various objective computations is a function to compute a general objective from a linear velocity v. The result depends on the general maximum achievable velocity v_L set from hardware limits and a velocity threshold v_{th}:

$$O_v(v, v_{th}) = \begin{cases} \frac{v_L - v}{v_L - v_{th}} & \text{if } v > v_{th} \\ 1 & \text{otherwise} \end{cases} \tag{8.1}$$

Avoidance The avoidance function considers the path of motion defined by the curvature resulting from a velocity pair (v, ω) and computes the minimum distance d_{min} to obstacles over this course of motion.

The velocity controller uses knowledge about obstacle positions to compute the minimum distance to any obstacles once the mother robot moves on a specific curvature. A number of control points $\vec{c_i}$ represent the curvature at times $t_i = m_i \cdot T$. Values of m_i are chosen to represent a specific time interval and result in a curvature segment which commonly starts after one control interval T.

The minimum obstacle distance $d_{min}(v, \omega)$ is the minimum distance from any of the control points to any obstacle.

With knowledge about the minimum obstacle distance the obstacle is enlarged by the robot radius r_o to compute the objective value.

$$a_{v,\omega} = \begin{cases} \kappa \frac{d_{min}(v,\omega)}{r_o} & \text{if } d_{min}(v, \omega) < r_o \\ \kappa + (1 - \kappa) \sqrt{\frac{d_{max}(v,\omega) - r_o}{d_{max} - r_o}} & \text{else if } d_{min}(v, \omega) < d_{max} \\ 1 & \text{otherwise} \end{cases}$$

The parameter κ is set to $\kappa = 0.05$ and represents the maximum objective value for curvatures that (theoretically) intersect with obstacles.

The objective $a_{v,\omega}$ is a risk evaluation of obstacles and for a specific curvature. To produce the final avoidance objective value, a value a_v is computed, so that the linear velocity is considered in this evaluation also:

$$a_v = O_v(a_{v,\omega}, \sqrt{a_{v,\omega}} \cdot v_L)$$

The final avoidance objective value is a combination of both previously described values and thus increases any (negative) influence of obstacles within a critical distance to the computed curvature:

$$a(v, \omega) = a_{v,\omega} a_v$$

Goal seeking The velocity controller receives a target heading θ_{dc} from the directional controller. The candidate velocity pair which most closely matches this heading after one control interval will be favoured. Linear and angular velocities are analysed separately and the predicted heading is calculated from the angular velocity ω.

$$g_\omega = 1 - \frac{|\theta_{dc} - (\theta_c + \omega T)|}{\pi}$$

Depending on the angular error a threshold for the linear velocity is set, so that high turn rates and high velocities are penalised:

$$v_{max} = \begin{cases} \left(1 - \frac{|\theta_{dc} - \theta_c|}{\beta\pi}\right) v_L & \text{if } |\theta_{dc} - \theta_c| > \beta\pi \\ v_L & \text{otherwise} \end{cases}$$

To combine the influence of the angular and linear velocity the final objective value remains a result of:

$$g(v, \omega) = g_\omega(\omega) O_v(v, v_{max})$$

Distance to goal The distance of the robot to a goal location d_g can have an influence on the velocity. Once the robot is closer than a distance $d_{g_{th}}$ to a goal location, the robot reduces its maximum velocity to avoid to overshoot.

$$v_{max} = \begin{cases} \frac{d_g}{d_{g_{th}}} v_L & \text{if } d_g < d_{g_{max}} \\ v_L & otherwise \end{cases}$$

The distance objective is then:

$$d(v, \omega) = O_v(v, v_{max})$$

Speed up The speed up function of EmotioNav favours high linear velocities up to the maximum velocity v_L:

$$s(v, \omega) = \frac{v}{v_L}$$

8.2 Challenges of 3D navigation

Because this project deals with 3D navigation of a wheeled robot, the main challenges of 3D navigation had to be identified. In the context of the limited capabilities of the mother robot, the main challenges are obstacles, ramps and drops.

8.2.1 Classifying Obstacles

The mother robot has the ability to negotiate obstacles, as long as they are not too high. But the classification of an actual obstacle as traversable or not depends on the current position of the robot and thus is not determined by an absolute height difference of neighbouring terrain locations. Factors such as current incline of terrain, attitude of the robot, and, in the case of the mother robot, a front body twisted with respect to the rear body can have an influence on the classification of obstacles. The following analysis will explain why the current attitude of the mother has to be an essential part of an analysis of the environment to allow the correct classification of obstacles as traversable or not traversable.

On an incline

The ability of the mother robot to traverse terrain changes, when its operates on inclined surfaces, due to slip, a different influence of gravity, and range finder data which is harder to interpret. Figure 8.1 illustrates the situation for the wheeled mother robot on an incline. When following a declining path, gravity and geometric configuration help the robot to overcome obstacles in the forward direction. However, when the robot has to climb a surface, the actual maximum height of traversable obstacles shrinks. This analysis assumes that this maximum height is defined by the vertical tangent to the wheels; a validation with the real mother robot is required before including this aspect into an implementation.

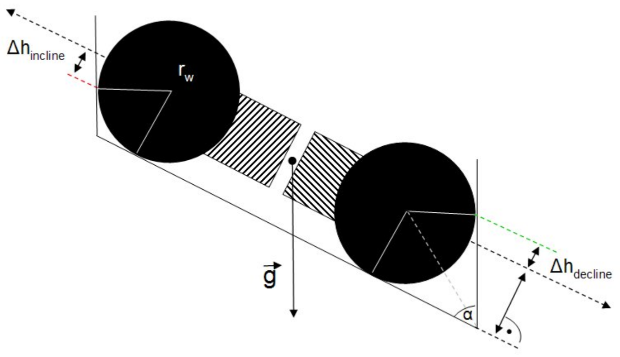


Figure 8.1: Decline increases maximum step height

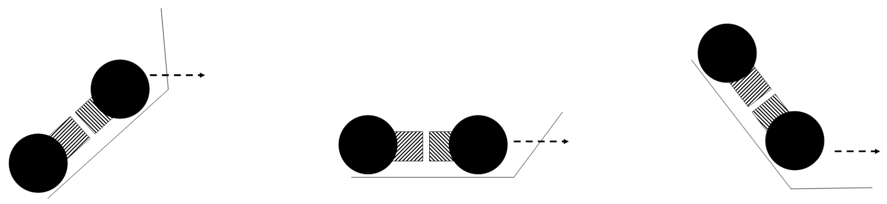
Local map ambiguity

The identification of obstacles results from an interpretation of the current range finder measurements and possibly previous ones (see Chapter 6), although with the risk of introducing errors through inaccurate localisation. The mother robot interprets its local map. This local map contains range finder measurements relative to the robot's centre ignoring the current attitude of the robot. Therefore, different situations can create the same local map. Figure 8.2 illustrates three situations with the same local maps, though the interpretation of obstacles should differ. The mother robot must therefore interpret its local map, taking into account the robot's attitude.

Twisted front to rear body

Any significant articulation of the front body with respect to the rear body increases the robot's challenge to interpret the environment and also changes its capability to overcome obstacles — previously traversable objects might be non traversable when the robot is twisted. In addition, the result of a velocity command is rather unpredictable in such situations considering the differential drive of the mother robot plus the limitations described in Chapter 4.

Angle of attack Whether an obstacle is traversable or not also depends on the angle of attack. This results to a main part from the limitation



(a) Ramp upwards (b) Planar (c) Ramp downwards

Figure 8.2: Different robot situations with same perception

pointed out in Chapter 4 and is a result of a drive with fix wheel directions and the twisted body design.

8.2.2 Detecting ramps - false obstacles

Ramps, if not too steep, are generally traversable. However, a sensor configuration in one plane — such as the original one of the mother robot — cannot detect ramps except in special cases. Range finder measurements must cover a vertical range of the environment to allow the detection of a ramp. Since the range finders are rigidly attached to the mother robot, the robot needs to change its attitude to create such a set of measurements. This is impossible in situations where the robot approaches a ramp from a planar surface and sweeps its sensors over the ramp with a rotational motion, or passes a ramp sideways. Therefore as explained in Chapter 4, either the sensor configuration has to be adapted to allow a more sophisticated interpretation of the environment or the robot has to approach possible obstacles until the robot is close enough to be able to differentiate between a ramp and an obstacle.

8.2.3 Traversing drops

Even though drops can be traversed, they pose a threat to a robot. The height of drops might be unknown and the effect of control commands is unpredictable once the robot has entered a drop. In a worst case scenario the robot ends up in a state which prevents any further operation. A general robust hardware design can lower the risk of breakage, but the loss of control remains.

8.3 Modifications to EmotioNav

EmotioNav is primarily designed for planar environments; adapting it to 3D environments involves modifying the avoidance function and its embedded detection of obstacles.

This project adds an additional stage that does a gradient analysis of range finder measurements to compute obstacle positions and to compute a traversability measure. These are used in a modified version of the directional and velocity controller. It also adapts the existing obstacle enlargement for the mother robot.

8.3.1 Gradient analysis

Short term mapping as presented in Chapter 6 generates a point cloud; plane fitting or other more sophisticated methods from object reconstruction promise accurate measures of point cloud analysis. Nonetheless, reactivity of the mother robot is essential and this project suggests a more simplistic analysis of the collected data to allow navigation in 3D.

This project builds a polar histogram which consists of sectors with a changeable resolution (currently 5°), so that a discrete number of sections exists. The gradient analysis is performed for each section, estimates the slope in the selected section and estimates the traversability in this direction. It also computes a minimum distance to non traversable obstacles in this section and takes into account the maximum step height and the maximum gradient that the mother robot can overcome.

Gradient analysis is performed on the local map of the mother robot. This local map ignores the current orientation of the robot (unless the robot is completely flipped, in which case the map will be adjusted with a back rotation to keep the correct reference to the gravity vector — required for the gradient analysis).

To facilitate processing of range finder measurements that build the local map, the measurements are augmented with their spherical coordinates — adding elevation and azimuth with respect to the robot's current position. The attitude of the robot gives further information about the ground underneath and allows a computation of slope in each direction from the robot centre. Each section is therefore also characterised with a base gradient g_b.

In order to compute a traversability measure τ the gradient analysis generates the following values, which are mainly derived from the traversability description in Chapter 2:

(1) actual mean slope $\overline{\tau}_s$ for each section

(2) roughness τ_r

(3) slope τ_s

(4) minimum distance to non traversable obstacle d_o

(1) The measurement points illustrated in Figure 8.3 allow the computation of a slope estimate τ_{s_i} for individual segments i taking into account the base gradient of a section:

$$\tau_{s_i} = \arctan(\frac{\Delta h_i}{\Delta d_i}) + g_b$$

where

Δh_i	vertical extent of segment i
Δd_i	horizontal extend of segment i

(2) After computation of the individual slope estimates, a mean slope $\overline{\tau}_s$ for a section is derived from all slope segments and the measure of roughness τ_r is computed:

$$\theta_r = 1 - \frac{1}{2N\pi} \sum_{i=1}^{N} (\overline{\tau}_s - \tau_{s_i})^2$$

(3) The gradient analysis computes the normalised slope measure τ_s for the current section and differentiates between inclines and declines:

$$\tau_s = \begin{cases} 1 - \frac{\overline{\tau}_s}{\tau_{smax}} & \text{if } \overline{\tau}_s >= 0 \\ 1 - \eta \frac{\overline{\tau}_s}{\tau_{s_{min}}} & \text{if } \overline{\tau}_s < g \\ 1 & \text{otherwise} \end{cases}$$

where

$\tau_{s_{max}}$	maximum traversable slope, here 20°
$\tau_{s_{min}}$	maximum decline, here -45°
η	drop aversion parameter in the range of $[0, 1]$

In order to compute a slope for the first segment (from 0 to 1 in Figure 8.3), this project considers that a wheel will contact an obstacle or ramp at a radius of at least 25 cm from the centre of the robot.

The drop aversion parameter η influences the sensitivity towards declining surfaces and is per default set to $\eta = 0.5$. Declining surfaces can have an maximum decline of $\tau_{s_{min}}$, which is assumed to be $-45°$. However, the mother robot only allows a drop detection up to $h_{drop} = 0.47$ m. This height derives from a sensing range $d_{max} = 3\,m$, the sensor mounting height $h_s = 0.2\,m$ measured from the travel surface of the robot, and the maximum angle of the sensing ray (12.5°):

$$h_{drop} = d_{max} \cdot \tan 12.5° - h_s$$

The drop detection height is far less than the maximum estimated drop height (2 m) and thus, does not have significance in the current project.

(4) A single segment can be steeper than the maximum traversable slope and therefore represents a step. If this step exceeds the maximum step height that the mother robot can traverse, the gradient analysis classifies it as a non traversable obstacle. When such a non traversable obstacle has been detected, the minimum obstacle distance d_o is updated to the distance between the segment start and the robot centre.

This process ignores negative steps or rather drops, otherwise the mother robot would not be able to drive on a declining surface.

Final value The final traversability τ for a section is the product of the individual traversability value for roughness and slope (see Chapter 2), so that:

$$\tau = \tau_r \cdot \tau_s$$

Remarks In some cases free space measurements (from a range finder showing maximum sensing distance) can be dropped from the analysis. In this project the current inclination of the mother robot sets an upper threshold for any free space measurements. Hence, free space can only classify drops but not (false) obstacles. In addition, subsequent free space measurements are reduced to the measurement which produces the minimum or better maximum negative slope — computed with respect to the last occupied space measurement in between the current free space measurement and the robot.

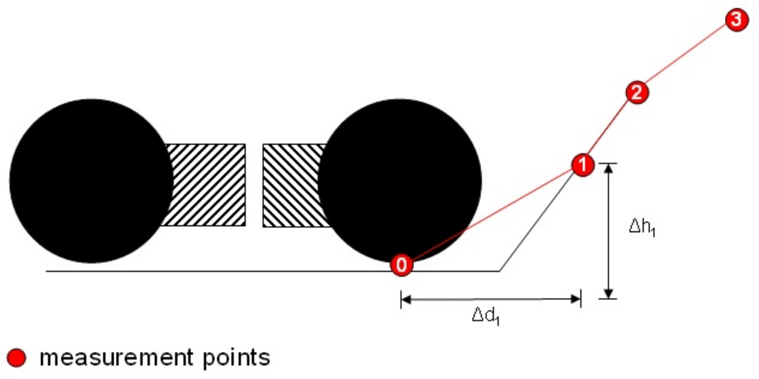


Figure 8.3: Analysis of measurement points

8.3.2 Modified directional controller

VFH+ is the foundation of the directional controller of EmotioNav and includes obstacle enlargement to compute its polar histogram. While EmotioNav performs circular obstacle enlargement, this project uses a more conservative approach for obstacle enlargement for the mother robot. This section will explain how obstacle enlargement is performed in this project and how the existing objective avoidance is adapted to include the newly developed gradient analysis. It will also note minor changes to the path following objective.

Obstacle enlargement

The directional controller of EmotioNav uses radius r_{oe} for obstacle enlargement. This radius is based on the circular approximation of the robot's actual extent plus a security distance to obstacles. After obstacle enlargement and the incorporation of a security distance the algorithm can treat the mother robot as a point size object.

An enlargement angle γ is calculated from the enlargement radius and the minimum distance of an detected obstacle to the robot centre:

$$\gamma = \arcsin \frac{r_{oe}}{d_o};$$

where

r_{oe}	obstacle enlargement radius (includes security distance)
d_o	distance of obstacle from to robot centre

This enlargement angle will cover a number of sections in the polar histogram — the neighbourhood of the section where the gradient analysis detected an obstacle. Though the robot is assumed to be roughly circular the minimum obstacle distance d_{o_i} of a specific section i in the neighbourhood is updated according to an obstacle line fitted to the enlarged section (see Figure 8.4). This results in an easier to compute and more conservative distance measure:

$$d_{o_i} = \frac{d_o}{\cos \alpha}$$

where

d_{o_i}	minimum obstacle distance limited to $[0, d_{max}]$

This approach can also be somewhat justified with the actual rectangular footprint of the robot and the requirement to face its travel direction with either its rear or back.

Avoidance objective

Gradient analysis allows the estimation of traversability in a specific direction, but it does not consider a neighbourhood of sections like obstacle enlargement. The avoidance function of EmotioNav takes the minimum

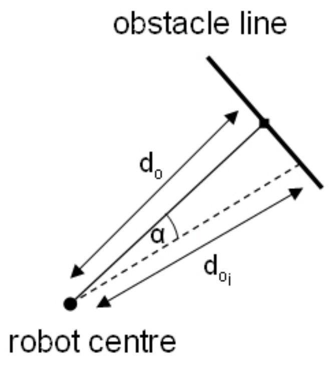


Figure 8.4: Obstacle line

obstacle distance after obstacle enlargement into account. Thus, the avoidance function of this project uses a conservative approach to combine the results of gradient analysis and obstacle enlargement to guarantee safe travel:

$$a(\theta) = \min\left(\tau, \frac{d_0(\theta)}{d_{max}}\right)$$

where

$d_0(\theta)$	closest obstacle in direction θ
d_{max}	maximum sensing range

Path following objective

Path following is applied to either follow the suggested path from the grandmother robot or follow a current path computed to the local environment. The mother robot uses a following distance of d_{pf_0}=3 m for the suggested path. Once the mother robot reaches the target area and starts its exploration mission (refer Chapter 9) a shorter following distance of d_{pf_1}=1 m applies to path planning — it then follows the continuously planned path through the local environment. The difference in the path following distances reflects the fact that path planning through the local environment is based on more current data and therefore more reliable.

Low pass filtering

Low pass filtering is part of navigation with VFH and also EmotioNav and intends to smooth the final objective function. Combined with a minimum function it reflects the importance of negative objective values. The low pass filter applied in this project is a weighted average filter to update the final objective values O'_n for each section n:

$$O'_n = \min(O_n, \frac{\sum_{i=-m}^{m}(m+1-|i|)\cdot O_{n+i}}{m(m+2)+1})$$

where m determines the size of the low pass filter in number of sections $(2m+1)$.

8.3.3 Modified velocity controller

Similar to the directional controller, the modified controller needs modification because the robot in this project is different to the one considered for EmotioNav. Hence, this project adapted the computation of the curvature pairs and some of the objective functions that are part of EmotioNav. These changes will be described in the following subsections.

The velocity controller of EmotioNav relies on knowledge about obstacle positions. The directional controller is executed before the velocity controller and in this project the identified obstacles are stored with their polar coordinates — with respect to the local map of the mother robot. The velocity controller can then access this information to compute a velocity pair (v, ω).

With the dynamic window approach as the foundation of the velocity controller only velocity pairs are evaluated, which can be reached from the current velocity setting. The velocity pair $(v = 0, \omega = 0)$ is part of every evaluation and represents the possibility of an emergency stop. Due to inertia this command might not result in an actual immediate stop, but it allows a reevaluation of the situation in the next control cycle to find the best solution. EmotioNav also incorporates the security distance to avoid a collision with obstacles in such cases.

Curvature segment

The computation of the control of the curvature segment is done for multiples of the control interval $t_i = m_i \cdot T$, where $m_i \epsilon M = \{1, 2, 5, 9, 14\}$ has been defined for the mother robot's control interval T = 0.2 s, representing the potential movement for the next 2.8 s. The values of m are motivated to give higher importance (and therefore higher resolution) to closer curvature segments, while also including a further look ahead. The velocity pair (v,ω) allows the computation of the radius of a curvature $r_c = \frac{v}{\omega}$. Thus, for each control interval t_i a control point exists, which lies on this curvature. Figure 8.5 illustrates the computation of control points. For $\omega \neq 0$ and $\alpha = \omega t$ follows:

$$c_{t_i} = \frac{v}{\omega}\begin{pmatrix} 1 - \cos 2\omega t_i \\ 0 \\ \sin 2\omega t_i \end{pmatrix}$$

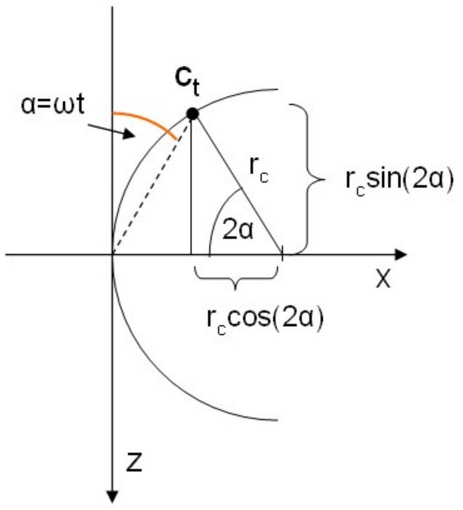


Figure 8.5: Computation of a curvature control point

Goal seeking objective

This project corrected obvious errors in the known function of EmotioNav, because its setting of the velocity threshold resulted in a negative velocity threshold:

$$v_{max} = \begin{cases} \left[1 - (1-\beta)\frac{|\theta_{dc}-\theta_c|}{\pi}\right] v_L & \text{if } |\theta_{dc} - \theta_c| > \beta\pi \\ v_L & \text{otherwise} \end{cases}$$

$$g(v, \omega) = g_\omega O_v(|v|, v_{max})$$

Speed up objective

The speedup objective of EmotioNav cannot be applied to the mother robot, because the mother robot has a field of perception with higher density to the front and only sparse information about the side and rear environment. Therefore, the mother robot should prefer forward motion. Nonetheless, backward motion cannot be excluded and even helps to avoid collisions or in situations where a point turn is unfavourable. The modified speed up function accounts for this situation and deals with positive and negative linear velocities:

$$s(v) = \begin{cases} 0 & \text{if } v = 0 \text{ and } \omega = 0 \\ \frac{1}{2}(1 + \frac{v}{v_L}) & otherwise \end{cases}$$

8.4 Evaluation

EmotioNav has been developed and tested in 2D environments. An evaluation of this project has to show a valid reimplementation and confirm the effective modification improving the robot's behaviour in 3D environment. The main focus is the ability of the mother robot to differentiate non-traversable and traversable terrain.

8.4.1 Directional controller

Gradient analysis

The gradient analysis introduces a major enhancement to EmotioNav and this section presents the output of the directional controller for a mother robot in different situations to show the working of the modifications. The output of the directional controller is evaluated using a diagram which presents the normalised objective values for the local map of the robot from direction from 0° to 360°.

Standard situations are analysed: Figure 8.7a represents the common situation where the robot travels on a flat surface. The current forward orientation can be derived from the angular inertia function (180°), which is equal to the direction of the goal. The avoidance function is not influenced by any obstacles in contrast to Figure 8.7b, where the robot, which currently sits on a flat surface, faces an edge and no range finder hits an obstacle. This situation can be also interpreted as a maximum upcoming decline of 12.5° (the drop aversion parameter is set to $\eta = 0.5$).

Travel on an inclined surface can be difficult for the mother robot, which is designed to negotiate up to a maximum incline of 30°. A decline does not change the behaviour of the directional controller as shown in Figure 8.8a compared to a flat surface. However, drops are only detected relative to the current position of the robot — otherwise the robot will already be moving in the drop section. The mother robot moves upwards on a ramp by avoiding the steepest incline. Depending on the current goal, the robot will avoid the straight path over the ramp (see Figure 8.8b) and influenced to traverse it with multiple curves.

To access a steep ramp comes at higher costs for the mother robot and alternative directions are preferred — illustrated by Figure 8.9a for a traversable ramp close the maximum traversable incline. It is also important that the mother robot can travel downwards on a ramp and that upcoming ground is not an obstacle. In Figure 8.9b the controller output shows deceased traversability at the right side, where its sensor hits an obstacle, but the main travel direction is considered as being traversable.

The sensed distance is of main importance for the gradient analysis to be able to differentiate between obstacles and ramps. Thus, the directional controller results in significant different outputs in cases of a close and a distant obstacle (see Figure 8.10).

Path following

The mother robot can apply path following for the suggested path from the grandmother robot and the continuously planned path through the local environment. Because the main properties for the path following behaviour are the same, this project validates path following by tracing the initial path of the robot to reach a target area. Figure 8.11 shows the environment which the robot has to explore and the subsequent results. The path which the grandmother robot suggested is visualised as a light red line. The mother robot tries to follow this path, but has to avoid obstacles at the same time — explaining the deviations in the exploration path of the mother robot. Path following ends when the robot starts its frontier based exploration (the solid red dot in the operation monitor of Figure 8.11b illustrates the next frontier region for the mother robot).

Continuous path planning with D* Lite intends to improve navigation of the mother robot. Evaluation shows that this approach does not hold for ramps which are steeper than the maximum traversable incline of the robot. The current graph building process does not account for subsequent steps and thus the suggested computed traversability measure allows paths with a slope of $\tau_p \approx 39°$ considering a maximum step height of 0.2 m and a resolution of 625 cm^2. However, the scenario in Figure 8.12 validates that continuous path planning still improves reactive navigation for steep sections.

Note that the evaluation in Chapter 9 relies on the directional controller setting listed in Table 8.1 and thus also validates the successful integration of continuous path planning into navigation.

objective	**weight**
avoidance	0.99
angular inertia	0.2
goal seeking	0
wander	0
path following	0.5

Table 8.1: Settings of the directional controller

objective	weight
avoidance	0.99
goal seeking	0.5
distance	0.2
speed up	0.2

Table 8.2: Parameter setting for the velocity controller

8.4.2 Velocity controller

For evaluation of the velocity controller a two dimensional curvature diagram is computed, which represents velocity pairs and obstacles. Figure 8.6 illustrates the model. Obstacles are represented as additional blue circles — the curvatures are heavily enlarged to allow a better visualisation and analysis. Therefore the obstacle size is schematic and does not represent a 1:1 relation to curvature sizes.

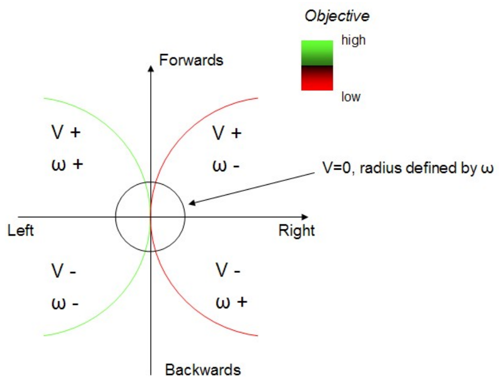


Figure 8.6: Curvature diagram including objective value

The evaluation in this section is based on a goal in the forward direction of the mother robot and EmotioNav parameters are set to values listed in Table 8.2.

Figure 8.13 presents four standard situations of the mother robot when the mother robot is currently stopped and computes the next velocity command, and further two situations when the mother robot is in motion. In the first example in 8.13a the mother robot's environment is free of obstacles. The velocity controller clearly favours forward velocities which lead into the direction of the goal. As soon as an obstacle exists (Figure 8.13b) the velocity controller favours velocity pairs which lead away from the obstacle. By analysing the current local map the robot can only detect obstacles to its front, side and rear. Being surrounded by obstacles the distance to obstacles becomes more relevant. Close side obstacles for example are then treated as a greater threat than distant (1 m) obstacles at the front of the robot (Figure 8.13c). Therefore even a straight forward path in direction of the front obstacles can be still considered.

The influence of the dynamic window becomes obvious, when the mother robot has a current forward velocity (0.5 $\frac{\mathrm{m}}{\mathrm{s}}$ in Figure 8.13e). The robot selects the best option from its reachable velocities and tries to avoid any obstacles (here at 1 m distance). Figure 8.13f shows the mother robot at the same speed but with closer front obstacles. The robot is forced to perform an emergency stop because the optional forward velocities have worse objective values.

The presented ′snapshot′ evaluation of standard situations shows the ability of the velocity controller to interpret situations correctly and compute appropriate control commands. Nevertheless this behaviour has to be validated with a constantly operating robot in simulation. The following section will therefore discuss the robot′s overall navigation behaviour in simulation.

8.4.3 General behaviour

Testing the mother robot in cube fields (see Figure 8.11a) showed that obstacle avoidance has been successfully implemented with the current modifications. The selected security distance is the main parameter to control when the robot commences its avoidance behaviour. In cases where the security distance is low (≤ 0.2 m) the mother robot can suffer from cutting corners, while considering only the current range finder measurements. Nonetheless an application of the mother robot in a field of rubble requires low security distance settings and simulation shows that currently collisions cannot be completely avoided in such environments with the mother robot — a result of the low sensor density at the sides and a local map which currently keeps none of the previous measurements. But the overall hardware design of the robot and its ability to operate inversely allow more tolerance to such behaviour.

The operation in a field of rubble sets a challenge for the mother robot, but with a modified EmotioNav the mother robot manages to explore major obstructed areas autonomously (see Figure 8.14) and becomes stuck only in rare cases. Then the manual interaction of the operator is required to free the robot. An example of a critical section is highlighted in Figure 8.15. One of the highlighted obstacles is thin and lifted. Initially, this increases the difficulty of detecting the obstacle. Furthermore, the object is light and likely to change position during contact with the mother robot, but the robot is yet not able to deal with dynamic objects properly.

8.5 Limitations

One of the major aspects of EmotioNav is parameter setting. This project embedded the infrastructure of EmotioNav into ExploreTM and allowed the activation of a parameter set at the start of the simulation. Therefore, the current navigation behaviour can be improved in situations of enclosures or other situations where the routes to a goal are heavily constrained. Though path planning can cope with some of these situations it becomes clear that path planning needs further extensions to cope with non traversable ramps and has to embed the maximum incline of the mother robot. Given such changes have been made, the influence of path planning has to rise dynamically in situations where the robot is stuck in order to take advantage of long term knowledge by increasing the confidence into the planned escape route.

The task to classify real obstacles is challenging for the mother robot. Hence, the mother robot can collide with obstacles and especially when using only a small security distance. This project introduces a parameter to influence drop avoidance, but this measure is currently not applicable due to limited field of perception. Thus, the ability to distinguish between a drop and a ramp remains a challenge — the mother robot cannot identify a high drop properly without having moved through it.

8.6 Possible solutions and extensions

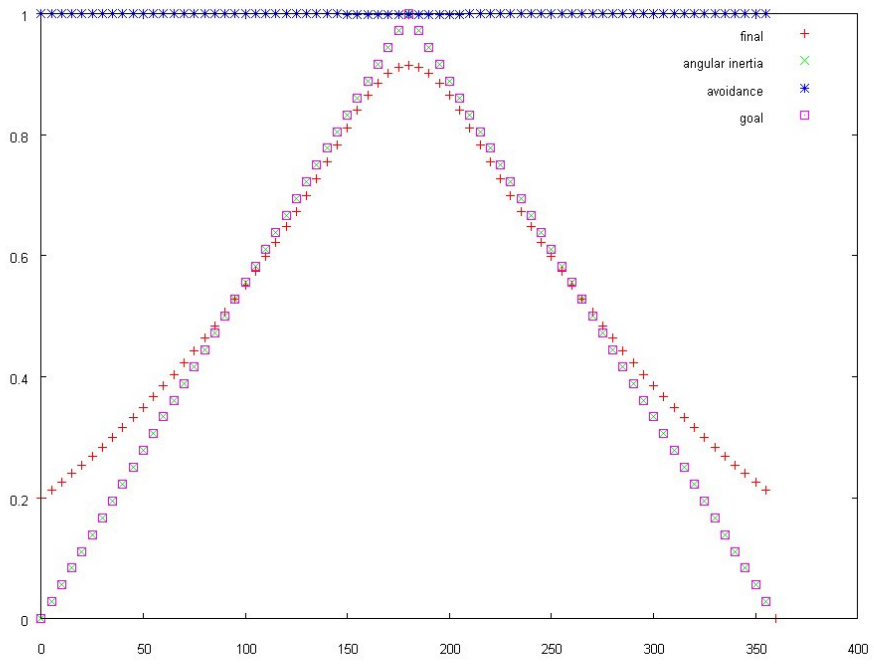
The requirements for a successful detection of inclines suffers from the planar orientation of the majority of the mounted range finders. In order to ease detection of ramps the symmetric sensor configuration has to be reconsidered. An alternative configuration is suggested in Figure 8.16, but the illustrated design might result in an earlier obstacle detection on side of the robot and thus might set a preference to a left or right hand turns to avoid obstacles.

EmotioNav comes with a large number of parameters and creates a challenge to find the right set of parameters for specific situations. To cope with specific situations the inclusion of a behavioural system with a rule base can be a possible solution. It could compensate for special weaknesses of the robotic platform and problems in specific situations, e.g. when the robot is somewhere stuck. Though generic solutions are desirable a rule base system can help to overcome initial problems until a generic solution has been found.

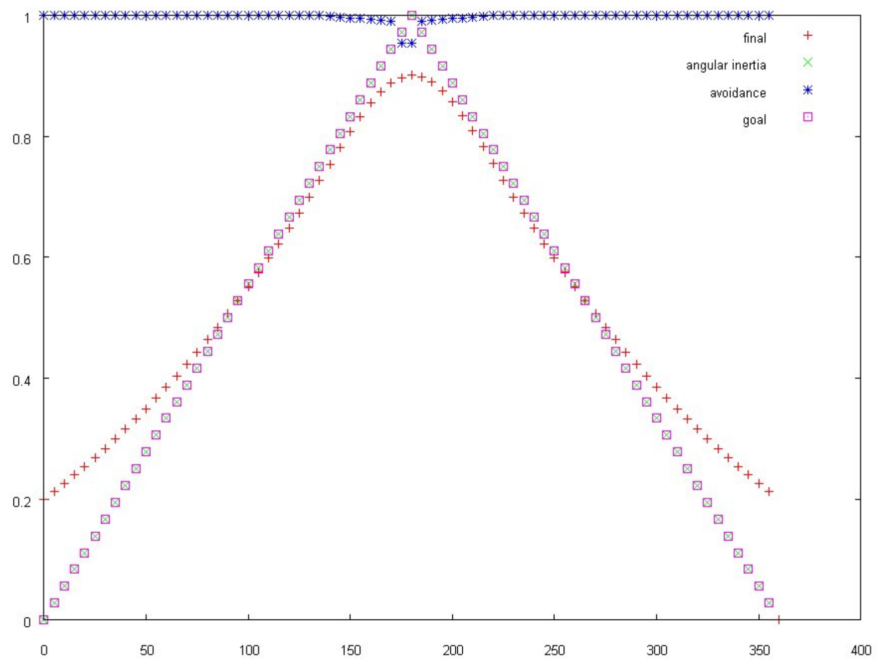
8.7 Summary

This chapter described the foundation of the navigational approach EmotioNav and its two main components — the directional controller and the velocity controller. It illustrated the necessary modifications to apply EmotioNav for operation in 3D environments, which included a solution to compute slope from current sensor data. The modifications also account for the kinematic capabilities of the mother robot.

The evaluation has shown that an application of EmotioNav in 3D environments is possible, but depends on the ability of the mother robot to classify non traversable obstacles. With the employed modification to EmotioNav the mother robot manages to explore major areas of a field of rubble autonomously. However, operator interaction can be still required in rare cases when the robot got stuck.

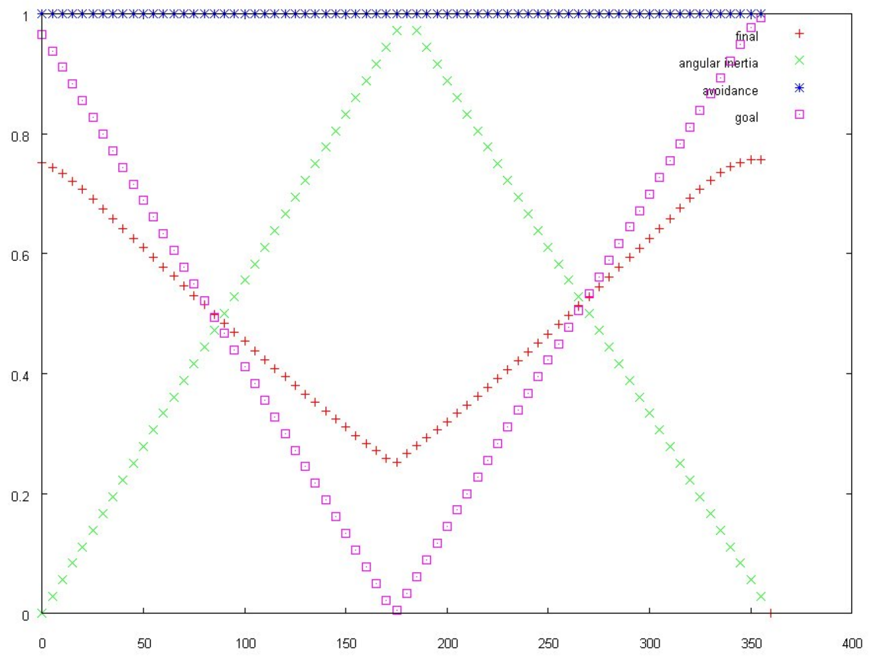


(a) Travel on a flat surface

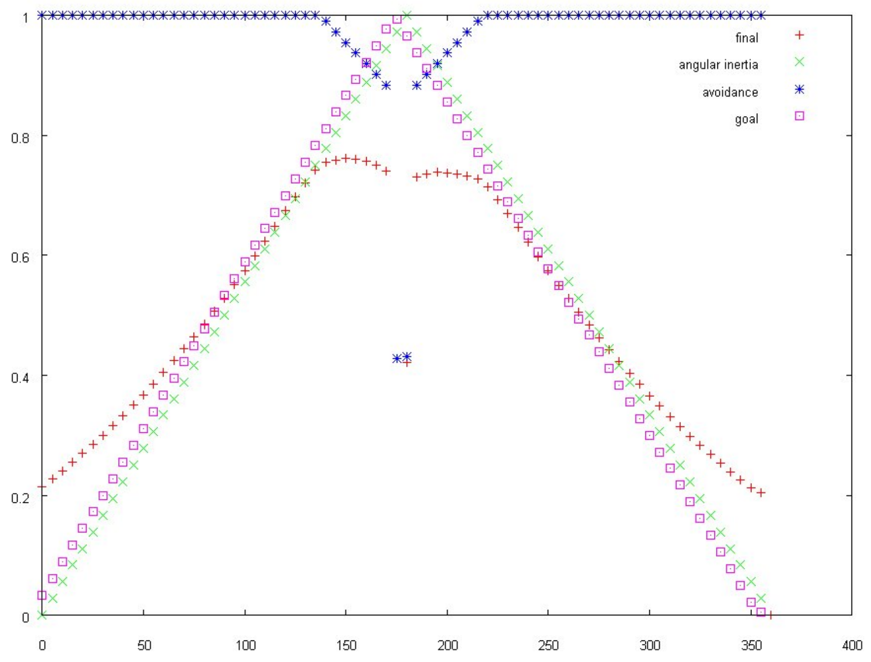


(b) Facing an edge

Figure 8.7: Directional controller output for standard situations



(a) Travel on a declining surface

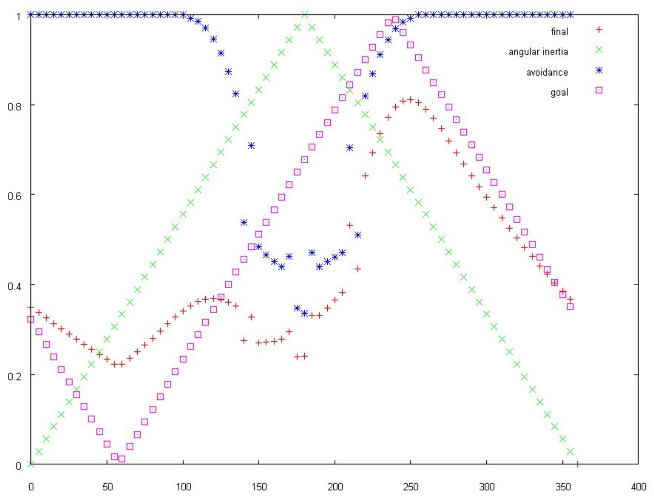


(b) Travel on an inclining surface

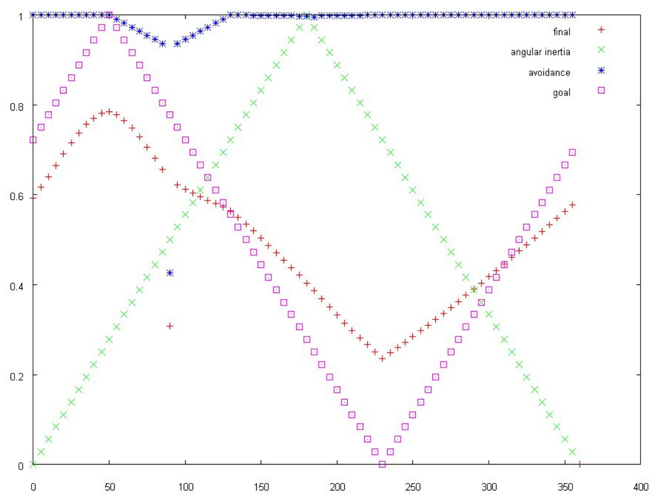
Figure 8.8: Directional controller output for standard situations

(a) Steep ramp ahead

(b) Downwards ramp

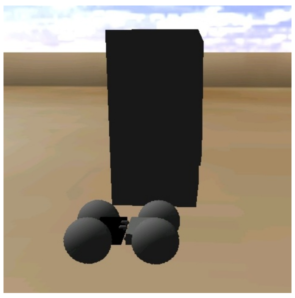


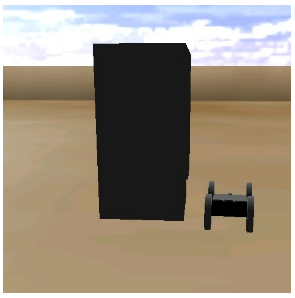
(c) Steep ramp directional controller results

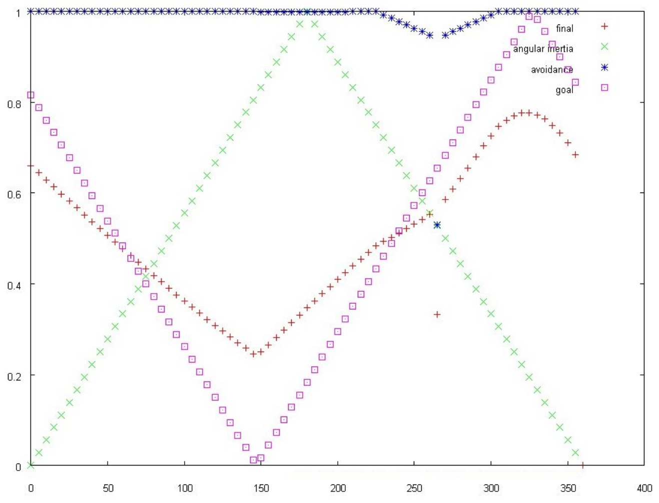


(d) Downwards ramp directional controller results

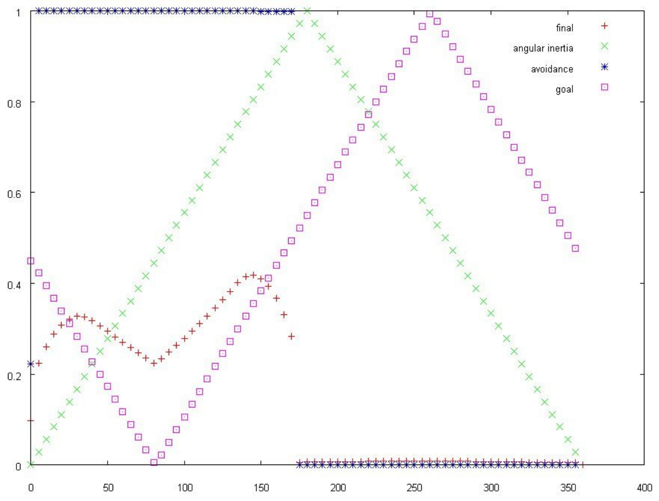
Figure 8.9: Situations with incline


(a) Distant obstacle


(b) Close obstacle

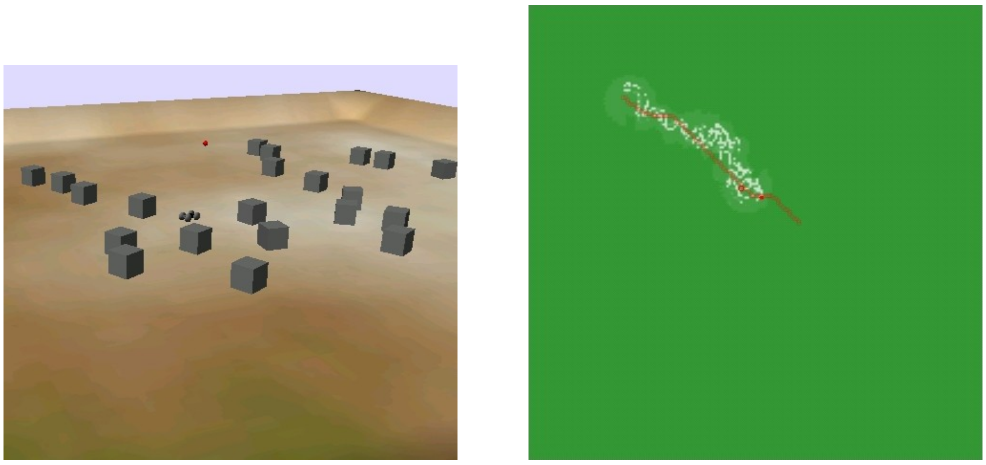


(c) Distant obstacle directional controller results



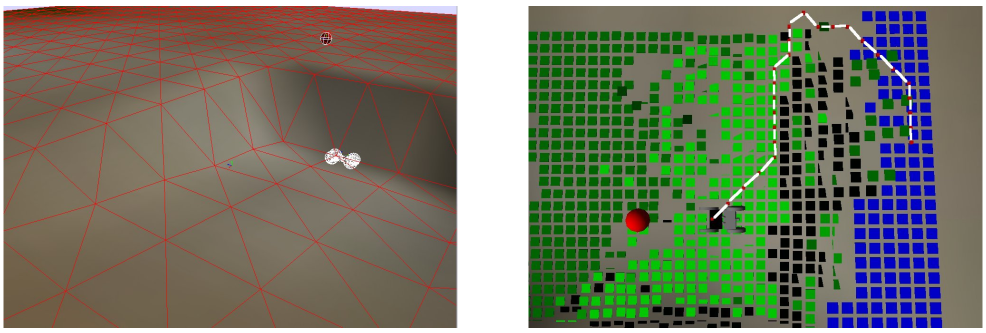
(d) Close obstacle directional controller results

Figure 8.10: Obstacle at varying side distances



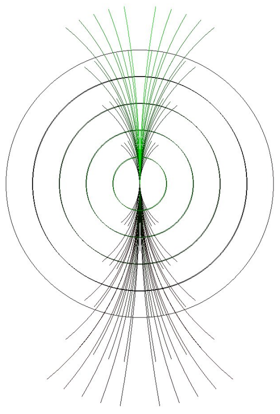
(a) Environment to explore (b) Exploration path

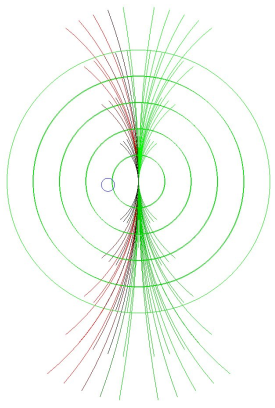
Figure 8.11: Path following to the target area

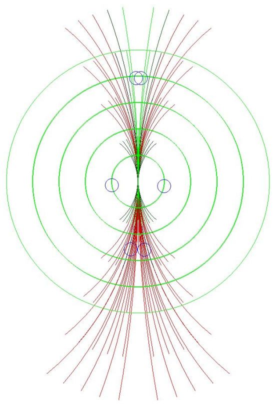


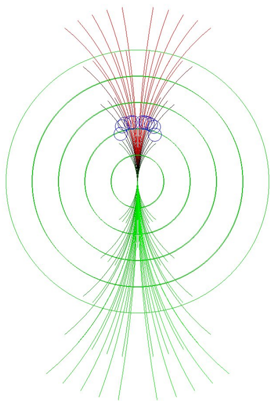
(a) Robot in a drop (b) Planned path to frontier

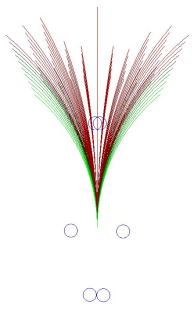
Figure 8.12: Path following to the next exploration frontier


(a) No obstacles


(b) Obstacle at left hand side


(c) Surrounded by obstacles


(d) Close obstacles at the front


(e) Distant obstacles at front

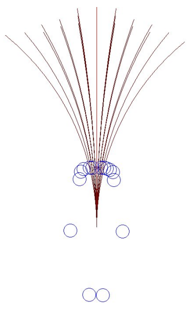
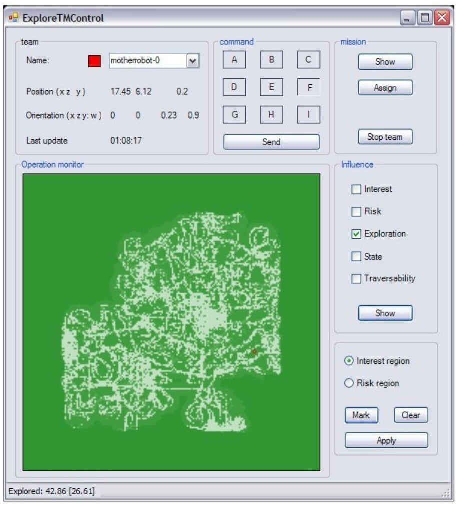

(f) Close obstacles at front

Figure 8.13: Velocity controller outputs



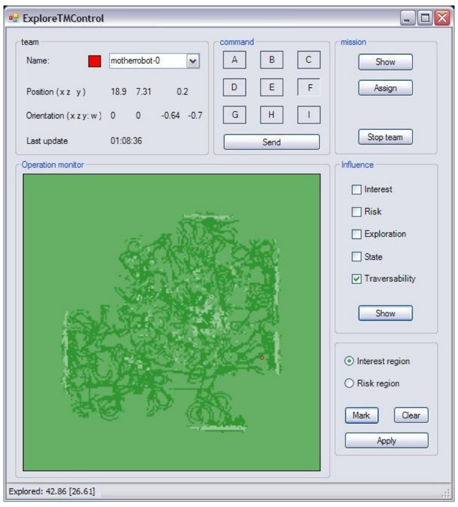
ExploreTMControl

Name: motherrobot-0
Position (x z y) 17.45 6.12 0.2
Orientation (x z y: w) 0 0 0.23 0.9
Last update 01:08:17

Influence: [✓ Exploration]

Explored: 42.86 [26.61]

(a) Explored area without complications



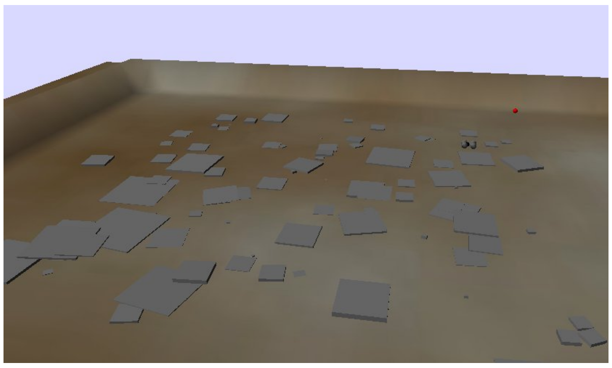
ExploreTMControl

Name: motherrobot-0
Position (x z y) 18.9 7.31 0.2
Orientation (x z y: w) 0 0 -0.64 -0.7
Last update 01:08:36

Influence: [✓ Traversability]

Explored: 42.86 [26.61]

(b) Traversability of explored area



(c) Operation area

Figure 8.14: Field of rubble and exploration

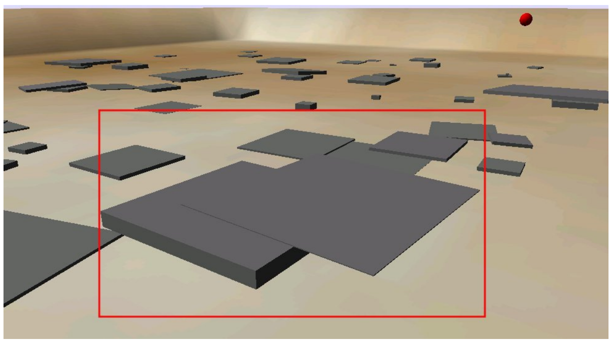


Figure 8.15: Problematic section

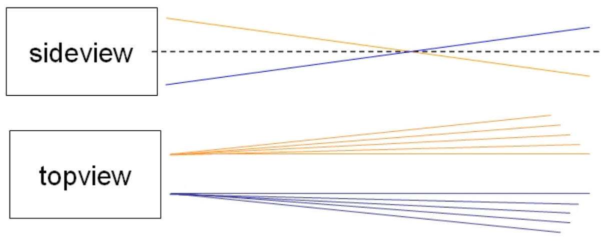


Figure 8.16: Possible asymmetric sensor configuration

Chapter 9

Exploration

The goal of the mother robots is the exploration of an environment to search for victims. Exploration can be done in a variety of ways on a spectrum of manual to fully autonomous. This project tries to achieve autonomous exploration with the design of two exploration strategies for the grandmother and mother robots — the global and local exploration strategies.

This chapter presents the details of the global and local exploration strategies within ExploreTM, which provides the control mechanisms for the hierarchy of robots.

9.1 Partially informed exploration

The general approach to investigating the operation area is based on partially informed exploration. The grandmother robot and mother robots will not have any prior information about the terrain to explore, but their operators might have. The operators will likely have a general idea about the structure of the terrain, which includes knowledge about areas which either promise a higher success rate to find victims or which just pose a threat to the continuous operation of the robots - this assumes previous experience of an operator and remains a subjective aspect. However, this project allows operators to encode their knowledge about the environment and provide advice for the grandmother robot.

In addition, the exploring robots will accumulate knowledge about the search area and send it at regular intervals (refer Chapter 3) to the grandmother robot. The grandmother collects and combines this information and uses it to optimise the global exploration strategy.

The design of the communication structure for operator, grandmother and mother robots allows the exploration strategy to be influenced by subjective and collected information. This project selected five different types of influencing factors which can be transmitted between these communication partners:

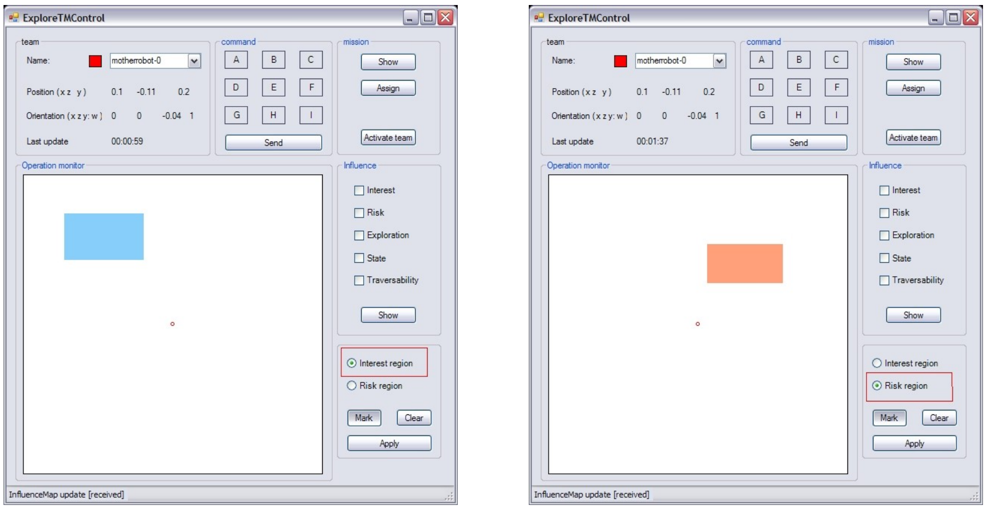
(i) degree of interest to explore a region

(ii) risk of operation

(iii) traversability

(iv) current state of the robots

(v) exploration status



Name: motherrobot-0
Position (x z y) 0.1 -0.11 0.2
Orientation (x z y: w) 0 0 -0.04 1
Last update 00:00:59
[✓ Interest region] Risk region
InfluenceMap update [received]

(a) Defining interest

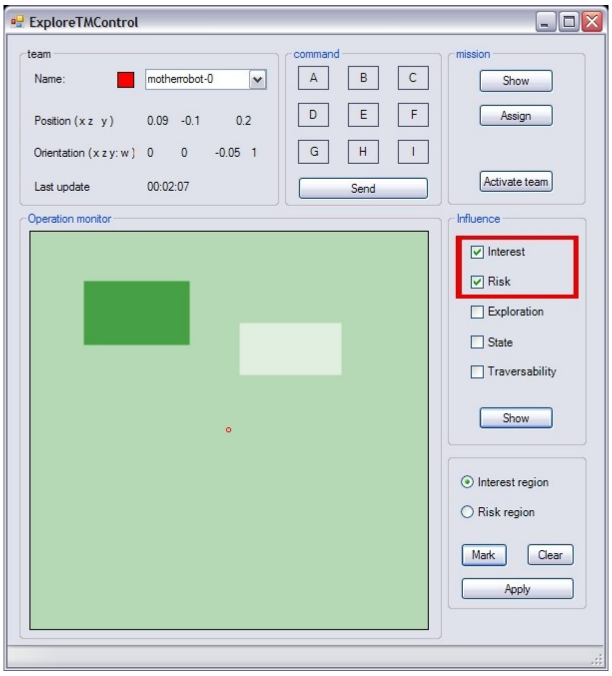
Name: motherrobot-0
Position (x z y) 0.1 -0.11 0.2
Orientation (x z y: w) 0 0 -0.04 1
Last update 00:01:37
Interest region [✓ Risk region]
InfluenceMap update [received]

(b) Defining risk

Figure 9.1: Operator actions

Operators can define the *risk* and *interest* for specific areas before or during the search operation (see Figure 9.1). This sets an initial focus for the search, and specific areas can be treated with preference. Some areas can also incur a higher risk of losing the exploring robots. Therefore, defining higher risk areas allows this to be taken into account during an operation.

The risk measure aims at increasing the probability of a longer operational state of the robots. These two influencing factors, interest and risk, are generic and apply for wide open search strategies as well as for confined spaces. An example of an influence map is presented in Figure 9.2 — darker green values represent a higher utility of exploring the region.



ExploreTMControl

team
Name: motherrobot-0
Position (x z y) 0.09 -0.1 0.2
Orientation (x z y: w) 0 0 -0.05 1
Last update 00:02:07

Influence
☑ Interest
☑ Risk
☐ Exploration
☐ State
☐ Traversability

◉ Interest region
○ Risk region

Figure 9.2: The influence map after marking interest and risk areas

The measure of *traversability* classifies explored areas (refer to Chapter 7); a default value applies to unexplored areas. This traversability classification allows path planning through unexplored terrains and allows the generation of an optimised path through known terrain.

A description of *state* accounts for temporary (in)capabilities of individual robots. Though the group of mother robots consists of homogenous devices with equal capabilities, the performance can vary during an operation. For example, battery discharge can limit the reach of an individual mother robot. Other factors such as the state of wheels or gearboxes, especially after drops in the environment, can also play a major role in limiting the speed or the reach of a robot. This project uses Euclidean distance from the robot's position to a target location to derive a utility measure of the location due to the state. Currently, the state influences only the selection of target areas, but not path planning, because path planning already includes a distance measure. The integration of the robot's actual hardware state will be a subject for future projects.

Exploration status is a normalised measure to describe the degree of exploration of a grid cell, i.e. 0 for an unexplored cell and 1 for a fully explored cell. In the MLS map, vertical space is actually unconstrained, so that a grid cell should never reach the exploration status of 1. However, this project expects a maximum height to be covered, so that the exploration status for a grid cell (given the cell has been explored at all) is currently computed in terms of the cell density as follows:

(1) Calculation of a cell density value φ

$$\varphi = \frac{\sum_{i=0}^{n} 2\sigma_n}{h_{max} - h_{min}}$$

where

n	number of space blocks in grid cell
$2\sigma_n$	vertical extent of a space block with index n
h_{min}, h_{max}	lowest/highest height level of all space blocks in current grid cell

(2) Exploration status ε:

$$\varepsilon = \min(1, \varphi^2 \frac{1 + n_{occ} + \alpha \cdot n_{free}}{E[N] + 1})$$

where

n_{free}, n_{occ}	number of free or occupied space blocks
α	constant, currently $\alpha = 0.4$
$E[N]$	expected maximum number of blocks per cell, currently $E[N] = 3$

The density parameter φ is a normalised value which describes the ratio between the height covered by space block descriptions and the maximum height difference covered by these blocks. The exploration status combines the density and the number of blocks in the cell. Note, that the expected number of blocks takes into account the kinematic constraints of Chapter 6. The final exploration function weights the free space block count by a constant $\alpha<1$. This reflects the assumption that knowledge about occupied space represents a higher quality of exploration than knowledge about free space blocks — a safely traversable area has at least a single occupied space block representing ground. The added pseudo count (+1 in numerator and denominator) rewards the initial exploration of a cell — even if just a free space block was computed.

The presented equations are an initial effort to provide a 3D exploration status for a 2D map to allow a map presentation in a 2D operator interface. As such it requires further evaluation to show the usefulness for multi-layered 3D environments. An evaluation could not be performed in this project due to time constraints.

9.1.1 Application of influence maps

This project uses so-called influence maps (see Chapter 2) for computation, storage and exchange of factors which can influence the global exploration strategy of the team of robots. Therefore, the grandmother robot maintains an influence map that is associated with the graph structure.

The influence map in this project is a 2D grid map and the stored values represent the five influence values associated with each grid cell of the graph (refer Chapter 7). The integration of these values into a planning operation requires the computation of a single, normalised value for each cell to represent the desirability or utility of visiting this cell.

Before computing this single value, some of the influence factors are rescaled. Rescaling allows an individual non linear weighting to be applied to individual factors, e.g. by using a degradation function for the state (with Euclidean distance as its input value).

The objective function as described in Chapter 8 is reused to compute the final influence value that represents the utility of a cell, and weights

Influence type	weight	rescaling
Interest:	0.5	none
Risk:	0.9	1 - <value>
Exploration:	1	1 - <value>
State:	0.5	$\frac{1}{1+\sqrt{\lvert 3-\text{<value>}\rvert}}$
Traversability:	0.5	none

Figure 9.3: Weighting of influence factors

are associated with each type of influence. The weights are currently set to stress the importance of the exploration status as the main goal of the mother robot and also to give risk significant influence. Note that when a cell is fully explored the total influence will equal zero, independent of other influence factors. The associated weights and any value rescaling are listed in Figure 9.3. The weights could also be a subject to modulation in subsequent projects.

This project uses an influence map with a fixed grid size to keep the memory footprint of the influence map small. Therefore the influence map uses a lower resolution than the graph. The current implementation uses an influence map size of 200×200 cells, so that the minimum resolution and therefore a lower bound for the influence map is 1.56 m^2, considering the currently supported maximum size of the grid being 1000×1000 cells with a resolution of 625 cm^2.

As a result of the different resolutions between global map and influence map, a single value in the influence map represents the utility of a neighbourhood of grid cells. To compute the influence from traversability and exploration status, the worst case assumption applies; the influence is computed based on the lowest traversability and the lowest exploration status of a neighbourhood of grid cells, thus leading to a conservative planning approach. Note that the traversability value of each grid cell is computed under the same assumption but from the nodes contained in the cell.

9.2 Exploration strategies

The overall exploration strategy of the team of robots consists of global and local exploration strategies. The global exploration strategy is computed on the grandmother robot and assigns missions (centres of target regions) to the mother robots. Each of the mother robots then tries to fulfil their assigned mission using the local exploration strategy. As soon as a single mother robot has completed its mission, the overall process is restarted and the grandmother assigns new missions to all the mother robots (refer Chapter 3).

Due to time constraints this project did not try to find optimal exploration strategies, but implemented simple (placeholder) solutions for the global and local exploration strategies instead.

9.2.1 Global exploration strategy

The goal of the global exploration strategy is to allocate a mission to each mother robot that optimises the influence value, the spread of the regions, and avoids conflicts between the robots. The global exploration strategy also suggests an optimised path to each of the mother robots to reach the assigned target area.

Allocation of the mission set

The grandmother has to compute a set of missions for the team of mother robots and the allocation process of the mission to the mother robots includes the information provided by the influence map. The global exploration strategy uses following optimisation process:

1. Build a candidate set of missions for each mother robot and evaluate their total region influence with respect to each robot
2. Find the top k missions for each mother robot, where $k = \max(20, 3 \cdot n_{mr})$ and n_{mr} is the number of mother robots
3. Create a candidate set of all mission combinations (only from the top k missions) for the team of robots, but ignoring the individual allocation to robots

4. Select the best mission combination for the team of robots
5. Allocate the missions to the individual mother robots, so that the sum of distances between missions and mother robots is minimised
6. Plan path for each robot and its allocated mission
7. Identify conflicts between the computed paths
8. Resolve conflicts (not implemented)

In order to find the top k missions for each mother robot, the grandmother analyses the influence map for each of the mother robots and also creates a list of candidate regions for each. The computation of these candidate regions uses a moving window approach (often applied in image processing). The window has a total size of 64 m^2 and is moved with a step size of 4 m to find the most attractive regions for a mother robot.

The identification of possible path conflicts between the mother robots is straight forward by analysing the distance between planned paths. Nevertheless, this project does not implement the final conflict resolution step which could be done by changing the mission allocation (based on the knowledge about conflicting ones) and repeating the process until no conflicts exist.

9.2.2 Local exploration strategy

Each mother robot uses the local exploration strategy to fulfil the mission received from the grandmother robot. A mission represents a target area to explore and is defined by a centre coordinate and a radius. A mother robot will apply its local exploration strategy to this area.

Frontier-based exploration [Yam98] has been selected by this project as the local exploration strategy for all mother robots. This strategy will prefer large and open frontiers with minimum obstruction by obstacles. First of all, the continuous graph update process marks explored grid cells as frontier cells when they have at least one unexplored neighbour grid cell. This labelling allows the mother robot to select the best frontier region after performing the following process to compute a list of candidate regions:

(i) Region growing (currently by factor 3 for an individual cell) is performed on cells with a traversability value smaller than 0.2 (i.e. obstacles) and frontier cells that are found to be obstacles are removed from the frontier.

(ii) Each frontier cell is assigned to an individual frontier region. These regions are greedily merged, as long as their centres are no more than 0.5 m apart from each other.

(iii) The new frontier regions are clustered with a standard hierarchical clustering method [Mac03]. The clustering process stops when the centres of all regions are at least 2 m apart from each other.

(iv) Frontier regions with fewer than the average number of cells per region are removed from the list of frontier regions, as are regions outside the mission target area or closer than 1.5 m

(v) The closest remaining frontier region will be explored.

A mother robot will apply its local exploration strategy as long as the mission is considered to be not fulfilled. To evaluate this criteria the exploration status of an area may be constantly evaluated, and a mission could be considered fulfilled when a certain percentage of the region has been explored. In this project, however, an exploration mission is fulfilled when no new frontier can be found in the region. In addition, separate timeout thresholds are part of the global and the local exploration strategies to either recompute the overall mission (6 min) or a target frontier (1 min).

9.3 Evaluation

As mentioned, the currently employed strategies are suboptimal and should be viewed as placeholder strategies which can serve as basis for subsequent developments. The performance of the exploration strategy will also suffer from the current limitations of navigation such as the missing modulation of navigational parameters. Nevertheless, this project performed a simple evaluation of the fundamental characteristics of the strategies to show the overall validity of the approach to include operator advice, and also points to individual aspects and elements to consider for improving the strategies in future projects.

The main criteria for an exploration strategy is the ratio between coverage and exploration time. The performance was measured for a team of mother robots exploring a planar environment, which is surrounded by an obstacle wall. The scenario allows a maximum of 56 % (about 2185 m^2) of the total area (about 3900 m^2) to be explored by the mother robots. The global strategy in this evaluation will assign missions with a radius of 10 m and the initial mission assignment is manually triggered at the start of each operation, but it does not include operator advice.

A single robot needed about 1 h to achieve an area coverage close to the maximum possible and three mother robot required about 0.5 h to explore the same area (Figure 9.4). The analysis of exploration times between a single and three exploring mother robots confirms, that the current combination of global and local exploration strategy is suboptimal. Note that the percentage of coverage of the area (cells which have been seen) is higher than the actual degree of exploration, which is based on the newly introduced measure to compute the exploration status (in which space known to be occupied results in a higher degree of exploration than empty space).

An autonomously operating team of mother robots is able to cover large parts of an area without any direction. Due to current characteristics and limitations of the exploration strategy the robots will spend more time getting around obstructed areas of the presented scenario, which results in a better exploration of these areas (for example Figure 9.5). Thus, the operation environment affects the exploration time. Figure 9.6 illustrates the increase of exploration time for a field of rubble with a team of three mother robots. A few tests have also been performed with a single mother robot in a cube field plus added rubble (Figure 9.7). These test have indicated that an exploration of such an area involves an increase of time by a factor >1.6 compared to an operation in a planar environment. However, due to time constraints this finding could not be further validated.

While the exploration strategies are suboptimal, an operator can influence the operation and thus increases the efficiency of the operation. Figure 9.8a illustrates the coverage and exploration for an initially undirected exploration. After about 10 min the mother robots do not gain significant new information about the area and the intervention of an operator (Figure 9.8b), and the setting of a new interest region help to refocus the exploration and increase the area coverage of the team of mother robots.

9.4 Limitations

Multiple mother robots do not share their maps and only the grandmother robot can access all information about the environment, so that the exploration of the mother robots can partially overlap; even once they have gotten to the target regions the mother robots might have to go outside the target area because the direct path to a frontier is obstructed.

An advanced centralised exploration strategy has to deal with multiple robots and needs to organise them so that the search activity does not interfere but is complementary. A centralised approach gives the opportunity to control the individual behaviour of the robots to a great degree, but this project takes a changeable and unpredictable environment into account and allows the mother robots to follow their individual exploration strategy. Furthermore, the global map with data from all robots is only maintained on the grandmother robot and currently not propagated to the mother robots. This aspect and the missing conflict management result in a suboptimal strategy that does not prevent the overlap of search activities.

In addition, the current selection of frontier regions as part of the local exploration strategy can be improved. Though region growing is enabled for obstacles, Figure 9.9 shows a frontier line (green) that extends between an obstructed section (red) and the unexplored area (black). This can result in the selection of frontier regions that cannot be reached. Simply increasing the obstacle growth factor can help, but might also lead to narrow passages being hidden.

An individual mother robot will trigger an mission assignment for whole team of robots when no frontier cell can be found or the completion of the mission takes too long. However, for some missions, although the target is not explored well, no frontier can be found, because the area has a high coverage already or is largely obstructed. Currently this triggers an immediate new request of a mission assignment which can lead to looping.

Subsequent global strategies should therefore incorporate a more sophisticated analysis of the exploration area to avoid such situations. In particular, the project uses 2D influence maps and a 2D operator interface, and the current local exploration strategy uses a 2D measure of coverage. A successful 3D exploration strategy must take multiple terrain levels into account and the data structures and exploration strategies need to be adapted.

9.5 Possible solutions and extensions

The accessibility of global knowledge for the mother robots needs to be improved and several options exists. In general, either detailed map information could be forwarded or only influence map could be used to transport information. To share detailed map information with the team of robots, the grandmother robot can forward the most current global map to the mother robots at regular intervals or the mother robots themselves could broadcast their most recent findings. Clearly, this places high demands on the communication channel bandwidth and thus the use of influence maps only should be investigated. Sharing global knowledge aims at increasing the efficiency of the team of mother robots and the overlap of search could then be avoided without a centralised conflict management on the grandmother robot.

Influence maps should be forwarded to the mother robots in any case as a next logical development step. This will allow an improved local exploration with the selection of exploration regions, which promise to be of higher value for the total operation.

With an enhanced operator interface the current representation of influence map can be expanded to three dimensions. Such an extension of the data structure should not significantly increase the memory requirements as long as dynamic allocation techniques are applied since the maximum number of nodes (surface patches) per grid cell is almost always less than four.

The exploration of a static environment can be performed by visiting locations once (though revisiting might be an advantage as well), but changing environments might require areas to be revisited. Therefore, the exploration status and eventually the map data itself of a single cell could be timestamped, so that exploration status and map data might decay to eventually influence the mother robots to revisit areas.

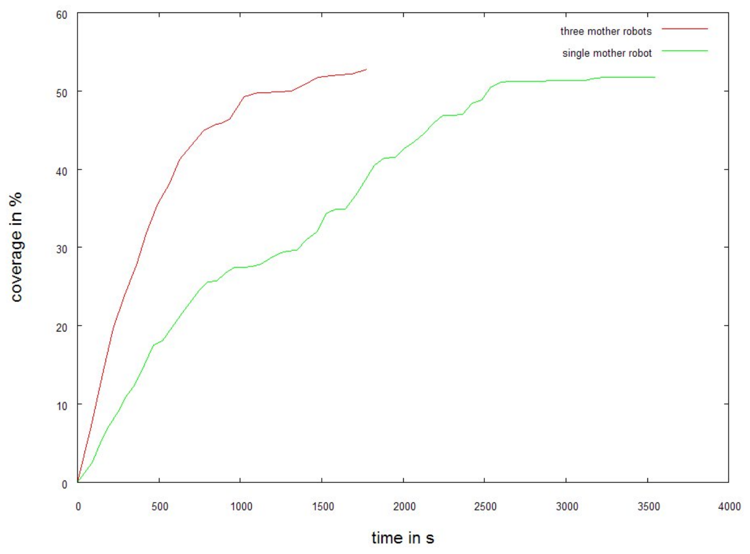
The sensor information that can be retrieved from the mother robots is limited. To gain more information from the mother robot, an (online) analysis of velocity and IMS data history could be performed. Such an analysis might lead to a more sophisticated terrain classification and allow an estimate of the time needed for the exploration of close regions. This information could subsequently be used to optimise the overall exploration strategy; using this data to optimise velocity control could be a further benefit.

The current implementation of the local exploration strategy provides tools for a more advanced analysis of frontier regions using image processing techniques for the computation of so-called moments. Frontier region have a more sophisticated geometric description through characteristics such as elongation, compactness and eccentricity. Currently frontier regions are clustered based on the distances of frontier cells to the region centre. As an extension of the current project, the optimisation process could take a geometrical analysis into account to improve clustering and selection of frontier regions for the group of robots.

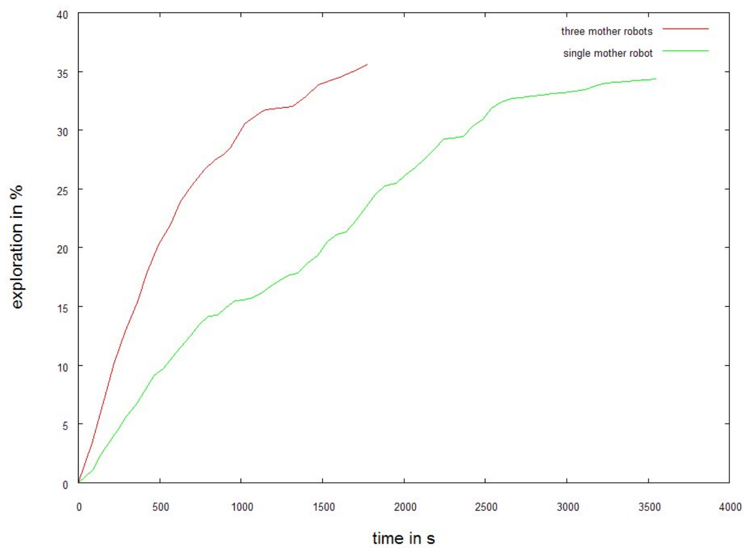
9.6 Summary

This chapter presented this project's approach to applying an exploration strategy for a group of robots. The exploration strategy can integrate prior information and uses influence maps as its main data structure, so that prior and current information can be combined. Consequently, the global exploration strategy is affected by different influencing factors, which allow an optimisation of the search. The local exploration strategy uses frontier-based exploration, but is also influenced by navigation strategy and the parameter settings of EmotioNav.

The combination of global and local exploration strategy defines the overall exploration strategy of the mother robots. The presented evaluation indicates that such combination is effective, but currently lacks efficiency and thus should be a subject of future research efforts.

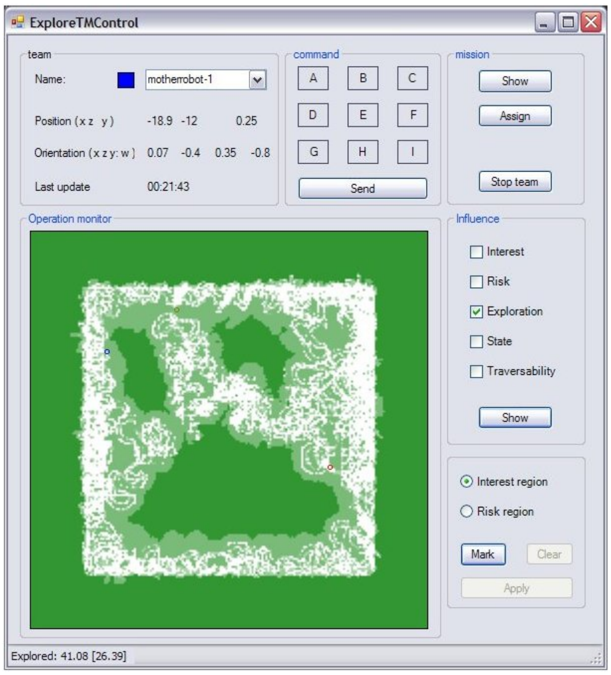


(a) Coverage



(b) Exploration

Figure 9.4: Coverage and exploration status using different numbers of mother robots



Name: motherrobot-1

Position (x z y) -18.9 -12 0.25

Orientation (x z y: w) 0.07 -0.4 0.35 -0.8

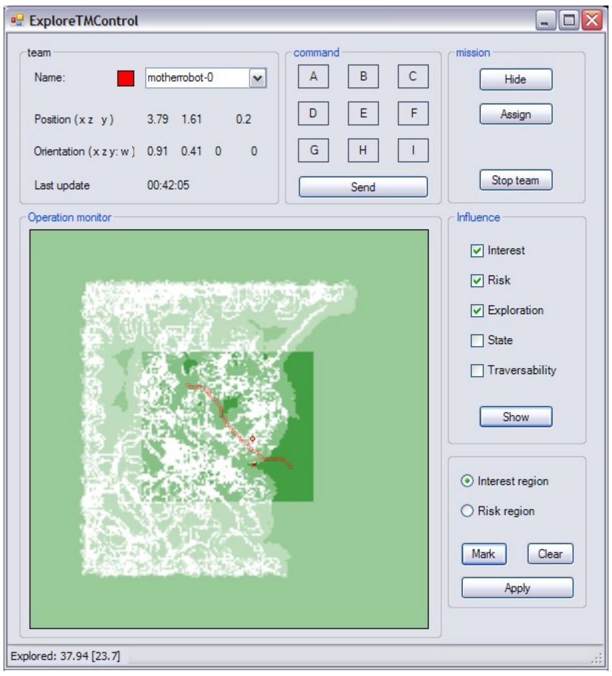
Last update 00:21:43

☐ Interest
☐ Risk
☑ Exploration
☐ State
☐ Traversability

◉ Interest region
○ Risk region

Explored: 41.08 [26.39]

(a) Three mother robots



Name: motherrobot-0

Position (x z y) 3.79 1.61 0.2

Orientation (x z y: w) 0.91 0.41 0 0

Last update 00:42:05

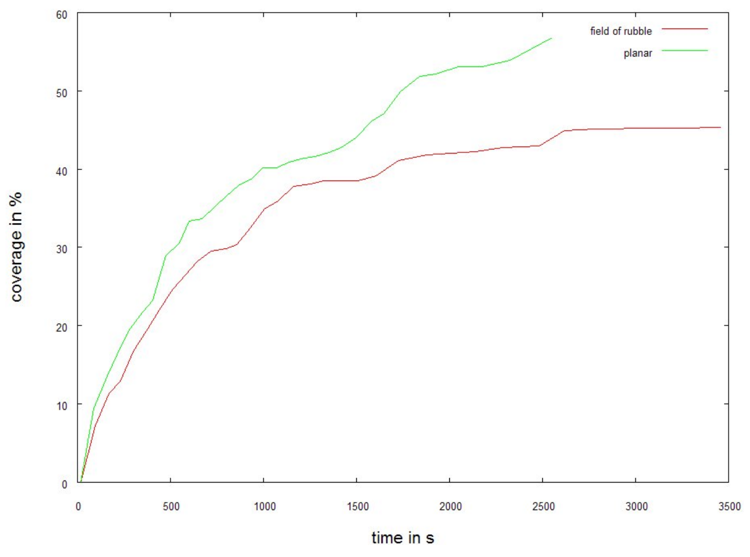
☑ Interest
☑ Risk
☑ Exploration
☐ State
☐ Traversability

◉ Interest region
○ Risk region

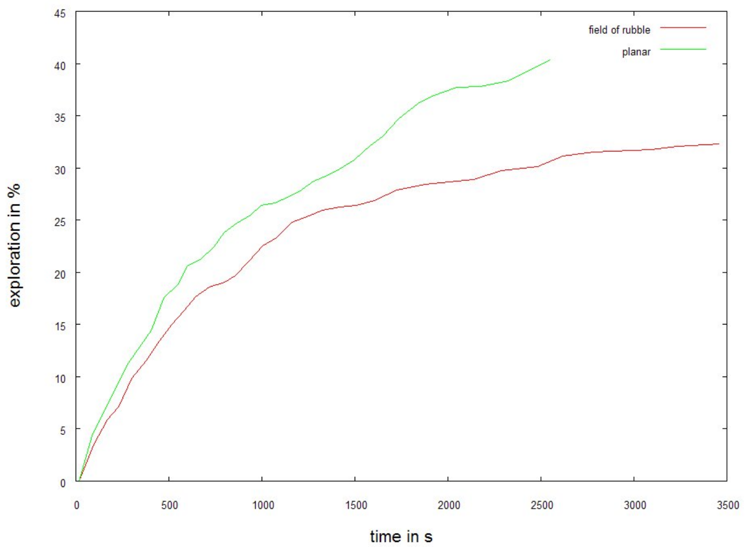
Explored: 37.94 [23.7]

(b) Single mother robot

Figure 9.5: Exploration with a different number of mother robots and operator advice

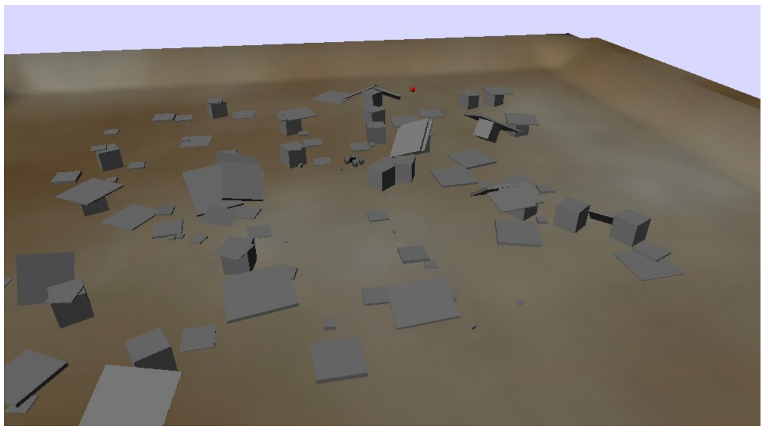


(a) Coverage

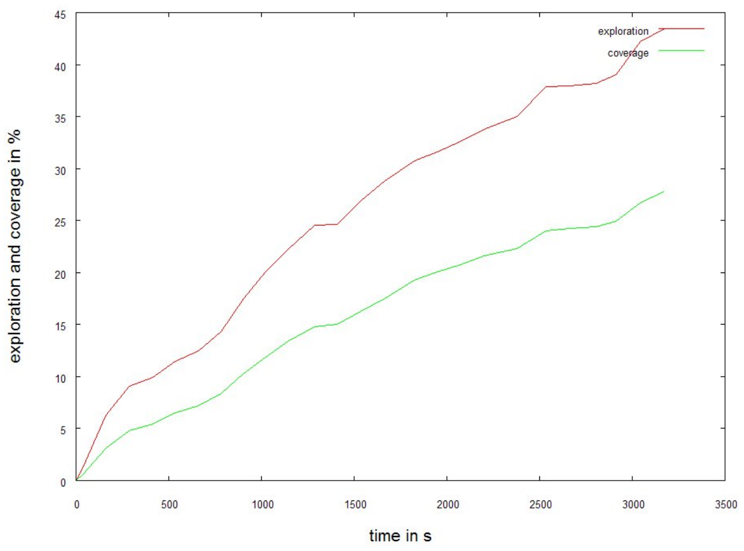


(b) Exploration

Figure 9.6: Three mother robots in a planar area and field of rubble

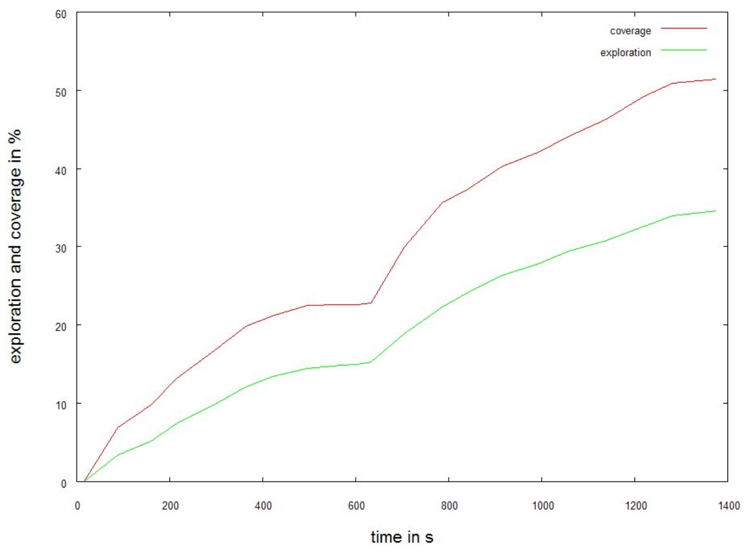


(a) Cube field plus rubble

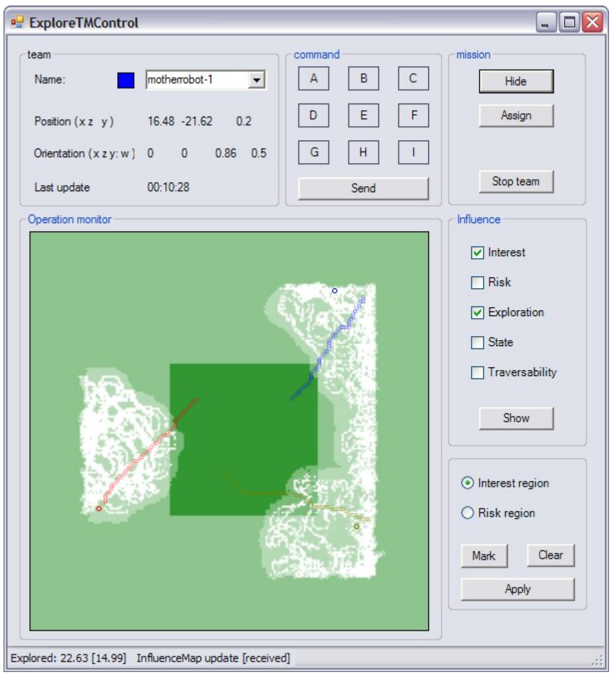


(b) Exploration time of a single mother robot

Figure 9.7: Exploration of single robot in obstructed terrain



(a) Effects of operator influence



Name: motherrobot-1

Position (x z y) 16.48 -21.62 0.2

Orientation (x z y: w) 0 0 0.86 0.5

Last update 00:10:28

- ✓ Interest
- ☐ Risk
- ✓ Exploration
- ☐ State
- ☐ Traversability

- ● Interest region
- ○ Risk region

Explored: 22.63 [14.99] InfluenceMap update [received]

(b) Operator influence

Figure 9.8: Total influence including the operator's advice, also showing the paths planned shortly after the advice was given

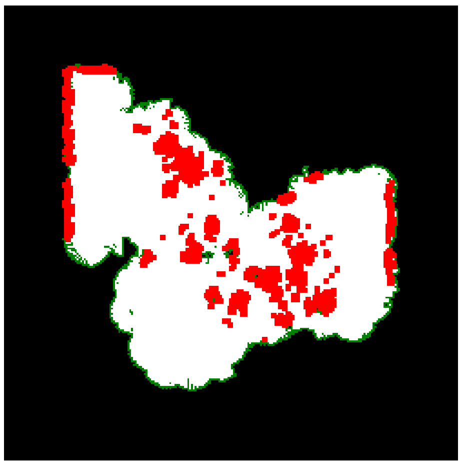


Figure 9.9: Cube field with rubble surrounded by obstacle wall

Chapter 10

Conclusions

This chapter outlines the contributions of this thesis and presents the conclusion from the current project. This chapter will also provide directions for future research.

10.1 Conclusion

The aim of this thesis was to provide a centralised control mechanism for a team of robots, specifically the grandmother and mother robots of [Car07]. To achieve this goal, it was necessary to set up a simulation which allowed the test of the individual control elements: mapping, path planning and navigation, as well as exploration techniques.

Simulation This thesis set up a simulation, created a model of the mother robot, embedded it into a simulation and tested it in a variety of operation environments. The simulation allowed the study of the dynamics and limitations of the mother robot including its control software and provided a powerful tool to improve the robots, so that it could meet challenges of USAR.

The simulation in this project was enhanced to easily set up a multi-robot team consisting of grandmother and multiple mother robots, so that a study of the control algorithm 'ExploreTM', its individual modules and collaborative properties was facilitated. Used throughout this project, the enhancements allowed a more efficient development and evaluation of algorithms for search and rescue operations.

The simulation has shown to be faithful in most situations and was applicable throughout the development of the control algorithm. By leading to the identification of major limitations in the current hardware design of the mother robots, the simulation has proven to be an essential part of the development process. With the ability to make immediate configuration changes, the simulation is therefore a valuable and low cost tool to identify early design limitations.

Hardware Design This thesis identified major limitations of the mother robots. The change of the hardware design was beyond the intended scope of this project, but was necessary to improve the mapping ability and increase robustness of the mother robot to allow operation in challenging environments. Financial costs and the requirement of symmetry for inverse operation have a great influence on the design of the mother robot and this thesis introduced changes to the hardware design which can be implemented with minimal costs and effort.

The limited perception still sets a challenge for autonomous operation and has severe implications for navigation in complex environments. Even with the modified range finder configuration of this project, the mother robot remains limited to the detection of small drops in the environment and the inclusion of the mounted camera will not change this situation. The navigation mechanism therefore requires an operation of the mother robots on the basis of optimistic assumptions, so that the mother robots can either explore their environment at increased risk or limit their field of action. With greater risk taking, the final performance of a mother robot relies to a large part on the sturdiness of the device.

Hence, from a variety of evaluations it can be concluded that the design of the mother robots needs reevaluation from a holistic viewpoint to allow safer and more predictable autonomous operation in cluttered environments.

System design This project served as an holistic approach to develop and validate a design for autonomous control for a hierarchy of robots. This thesis suggested a general high level control structure for the team of robots and provided the design for a message based system for communication between the different team members. It also implemented a solution to include operator advice into an autonomous control structure. Though the individual elements of control have to deal with the specific limitations of the hardware platform, the overall system design has been shown to be a successful and effective approach to allow for autonomous operation and exploration with the grandmother and mother robots.

Control algorithm The control system 'ExploreTM' is a result of this thesis and combines new and existing algorithms for mapping, path planning, navigation and exploration. ExploreTM allows autonomous operation of the individual mother robot and autonomous control of the team of robots based on a single grandmother and multiple mother robots. It also fulfils one of the major requirements of USAR robots by accounting for inverted operation. All algorithms have been evaluated in simulation to show that they fulfil the time performance constraints for application on the mother robots. The following paragraphs will address the individual algorithms of the control framework.

Mapping Chapter 6 presented a novel approach for mapping that is based on MLS maps. It uses descriptions of not only occupied but also free space and allows the current robots to operate with a small set of range finders by using incremental data collection. Though mapping of 3D environments usually sets high computational constraints, this thesis shows in Chapter 6 that resourceful 3D mapping can be performed with the low cost mother robot under some assumption about localisation. This can be done with sufficient accuracy for subsequent planning operations.

Path planning This thesis presented a distributed path planning approach in Chapter 7 and adopted a traversability measure to provide a fuzzy terrain classification which included the kinematic constraints of the mother robots.

Classification and path planning depend on the quality of the mapping results, but terrain classification showed good results, so that effective path planning through the mapped 3D environments was possible. The evaluations in this thesis lead to the conclusion that the real mother robot will be able to perform continuous path planning during autonomous operation. The results of Chapters 8 and 9 also confirm that for the mother robot a hybrid solution for navigation including a path planner is a feasible and valuable alternative to a purely reactive approach.

Navigation With modifications to the existing EmotioNav, this thesis makes a significant contribution to allow operation of the mother robots in a 3D environment. The modified navigation described in Chapter 8 enables the mother robot to operate autonomously and under a limited threat of failure in a complex environment such as a field of rubble. While operating on inclining and declining surfaces, it will still be able to distinguish traversable and non-traversable terrain.

Exploration An exploration strategy for the team of robots has been implemented in Chapter 9 which takes partial information about the environment into account. This thesis suggests combing global exploration strategy for the team of robots with a commonly used local exploration strategy. The implemented exploration strategy is a valid, though as expected, a suboptimal strategy for the team of robots. However, Chapter 9 successfully confirmed that autonomous and effective exploration for the hierarchy of robots can be achieved with the overall system design.

10.2 Future research

This thesis presented limitations of the current hardware and newly developed algorithms, and presented possible solutions to remove those limitations or expand on the current state. Future research projects could address one or more of the illustrated ideas, but the following presentation gives a summary of the main research directions for future developments.

10.2.1 Real world application

As a result of the currently broken real mother robot, this project based much of its hardware analysis on a mother robot which performed in simulation. Due to known limitations of the simulation it is of major importance to validate the findings and evaluate their impact on autonomous operation of the mother robots in a real world scenario.

Applying the presented software solutions to the real mother robot requires further efforts implementing filter techniques. The current mapping techniques operate on the assumption of no horizontal localisation error. Thus, while mapping is the central algorithm of ExploreTM, one of the solutions discussed in Chapter 5 to achieve a low localisation error needs to be implemented. Further, the integration of multiple maps on the grandmother robot needs to cope with the remaining localisation error.

The issue of sensor noise needs to be generally addressed by evolving a faithful sensor model for the employed devices, and neural nets offer a solution. A final mapping solution should include current advances in SLAM and consider an application of constrained SLAM.

10.2.2 Redesign of the mother robot

To allow flexible and safe operation of the mother robots, the hardware design is fundamental. Safe operation results from an optimal use of available range finders and the sturdiness of the robotic platform. To deal with drops in the environment and to negotiate a cluttered environment three issues have to be resolved:

The drive The wheeled mother robot still has problems negotiating 'traversable' obstacles, depending on the angle of attack, current velocity and other influencing factors. A better suspension or even a change of the operation platform should be evaluated to increase the stability of the mother robots during operation and to gain a more predictable control command output in problem situations.

Sensor configuration As one of the major limitations, the perception of the robot needs to be improved to allow operation and mapping. To operate the mother robot in a hostile environment the sensor configuration needs to allow the detection of significant drops and a simplified detection of ramps. This can either be achieved by optimising the current sensor configuration, or expanding on the existing one. A simplified detection of ramps has already been suggested in Chapter 8 but might require a change of paradigm away from the vertically symmetric mother robot design.

Split robot body The split robot body design has negative implications for traversing obstacles in the simulation, as well as chances of ending up stuck in a twisted state. The seriousness of this finding has to be verified in a real scenario and if the issues can be confirmed, an evaluation has to show whether the navigational control can cope with this limitation or if a redesign is required.

10.2.3 Navigation in cluttered environments

EmotioNav has been included into this thesis and has been modified to work in three dimensions. Future research should look at an integration of emotions and parameter modulation as well as researching the effect of a 3D operation environment. However, determining whether this adds value in the USAR application is important.

10.2.4 Collaboration

To ensure the survival of victims in a USAR operation, time is a critical factor. The controlled application of a team of robots helps to speed up the overall time to find victims, but more efficient exploration techniques have to be developed. Exploration of an area will profit from an improved communication between team members and an increase of information exchanged. The current project has set an exploration model in place as a base on which future research should study and develop more intelligent solutions for a collaboration of the team of robots for search in cluttered environments.

10.2.5 Improvement of the operation interface

Though autonomous operation is the ultimate goal, it is important for an operator to be able to monitor and map an environment as well as override the current autonomous control. Because the robots operate with

3D data and positions can be on different levels in a MLS map, a human operator needs to be able to visualise situations in 3D. Thus, an interface is needed which visualises the current 3D map data in an appropriate way to the operator. This approach has to consider limited network bandwidth and computational constraints of the grandmother robot as well as the capabilities of the operator's device.

10.3 Summary

This thesis is a significant contribution towards the overall development of a hierarchical team of USAR robots at Victoria University of Wellington. With ExploreTM, this thesis introduced a variety of algorithms to solve the challenges of mobile robotics arising from the need of autonomous operation in a 3D application environment with a low cost robotic architecture. These algorithms eventually allowed the successful implementation of an autonomous control mechanism for the grandmother and mother robots. This thesis also established a powerful tool for the analysis and research of robot hardware and software design in a safe and low cost environment. This tool simplifies evaluation and tuning of the implemented parameterized solutions to a major degree, and will serve as an ideal entry point for the subsequent development of the robot architecture consisting of grandmother, mother robots and daughter robots.

One paper has been published in the development process of this project [RCA07] awarded with the "Best poster award" on ENZCON 2007. This success reflects the fact that this project has in many cases exceeded the project objectives by contributing in the areas of both hardware and software design for the hierarchy of robots as well as providing a flexible tool for future research.

Acronyms

CCR	Concurrency and Coordination Runtime
DGPS	Differential Global Positioning System
EGNOS	European Geostationary Navigation Overlay Service
GPS	Global Positioning System
IMS	Inertial Measurement System
INS	Inertial Navigation System
LRF	Laser Range Finder
MEMS	Micro-Electro-Mechanical System
MLS map	Multi Level Surface Map
MRT	Microsoft Robotics Toolkit
ODE	Open Dynamics Engine
SBAS	Satellite-Based Augmentation Systems
SLAM	Simultaneous Localisation and Mapping
USAR	Urban Search and Rescue

VFH	Vector Field Histogram
XML	Extensible Markup Language

Bibliography

[AGE07] AGEIA, *Ageia physx*, "`http://www.ageia.com`", July 2007, last accessed on 14/7/07.

[Ark98] Ronald C. Arkin, *Behavior-based robotics*, 1rst ed., MIT Press, 1998.

[BDW06] Tim Bailey and Hugh Durrant-Whyte, *Simultaneous localisation and mapping (slam): Part ii state of the art*, IEEE Robotics & Automation Magazine **13** (2006), no. 2, 108–117.

[BK91] J. Borenstein and Y. Koren, *The vector field histogram-fast obstacle avoidance for mobile robots*, Robotics and Automation, IEEE Transactions on **7** (Jun 1991), no. 3, 278–288.

[BK99] O. Brock and O. Khatib, *High-speed navigation using the global dynamic window approach*, Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on **1** (1999), 341–346 vol.1.

[BMF$^+$00] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, *Collaborative multi-robot exploration*, Proceedings ICRA '00 IEEE International Conference on Robotics and Automation, vol. 1, 24-28 April 2000, pp. 476–481.

[Bro86] R.A. Brooks, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation (now IEEE International Conference on Robotics and Automation) **RA-2** (1986), no. 1, 14–23.

[BT90] G.A. Bekey and R. Tomovic, *Biologically based robot control*, Proceedings of the Twelfth Annual Intl. Conference of the IEEE, 1-4 Nov 1990, pp. 1938–1939.

[Car07] Dale A. Carnegie, *A three-tier hierarchical robotic system for urban search and rescue applications*, Proceedings of the 2007 IEEE International Workshop on Safety, Security and Rescue Robotics, September 2007.

[CFS06] Joseph Carsten, David Ferguson, and Anthony (Tony) Stentz, *3d field d: Improved path planning and replanning in three dimensions*, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06), October 2006, pp. 3381 – 3386.

[CGB01] G. Chen and D.A. Grejner-Brzezinska, *Land-vehicle navigation using multiple model carrier phase dgps/ins*, American Control Conference, 2001. Proceedings of the 2001 **3** (2001), 2327–2332 vol.3.

[CMBG07] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, *A survey of commercial & open source unmanned vehicle simulators*, Robotics and Automation, 2007 IEEE International Conference on (10-14 April 2007), 852–857.

[Cra97] Michael Cramer, *Gps/ins integration*, "`http://www.ifp.uni-stuttgart.de/publications/phowo97/cramer.pdf`", 1997, last accessed on 03/07/08.

[Cyb07] Cyberbotics, *Cyberbotics - professionel mobile robot simulation*, "`http://www.cyberbotics.com`", July 2007, last accessed on 12/07/07.

[Dep08] Department of Homeland Security, *Department of homeland security, national institute of standards & technology - urban search and rescue robot performance standards*, "`http://www.isd.mel.nist.gov/US&R_Robot_Standards/`", February 2008, last accessed on 20/02/08.

[DJ00] Gregory Dudek and Michael Jenkin, *Computational principles of mobile robotics*, Cambridge University Press, 2000.

[DWB06] Hugh Durrant-Whyte and Tim Bailey, *Simultaneous localisation and mapping (slam): Part i the essential algorithms*, IEEE Robotics & Automation Magazine **13** (2006), no. 3, 99–110.

[EG08] Earthquake Commission and GNS Science, *Geonet*, "`http://www.geonet.org.nz/about/`", April 2008, last accessed 26/04/08.

[Elf89] A. Elfes, *Using occupancy grids for mobile robot perception and navigation*, IEEE Computer, vol. 22, June 1989, pp. 46–57.

[ESA08] ESA, *Egnos european geostationary navigation overlay service*, "`http://www.esa.int/esaNA/GGG63950NDC_egnos_0.html`", February 2008, last accessed 02/02/08.

[Eur08] European Commision, *Galileo european satellite navigation system*, European Commision "`http://ec.europa.eu/dgs/energy_transport/galileo/index_en.htm`", February 2008, last accessed on 2/2/08.

[FBT97] D. Fox, W. Burgard, and S. Thrun, *The dynamic window approach to collision avoidance*, Robotics & Automation Magazine, IEEE **4** (Mar 1997), no. 1, 23–33.

[FEM08] FEMA, *Department of homeland security federal emergency management agency (fema) - urban search and rescue*, "`http://www.fema.gov/emergency/usr/`", March 2008, last accessed 01/03/08.

[FKK$^+$06] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, *Distributed multirobot exploration and mapping*, IEEE Proceedings, vol. 94, July 2006, pp. 1325–1339.

[FS06] David I. Ferguson and Anthony Stentz, *Multi-resolution field d*, IAS, 2006, pp. 65–74.

[Gal81] F.E. Gallas, *Land search and rescue*, Federated Mountain Clubs New Zealand, 1981.

[GHH$^+$07] B. Gerkey, R. Hedges, A. Howard, K. Stoy, and R. Vaughan, *The player/stage project*, "`http://playerstage.sourceforge.net`", July 2007, last accessed 11/07/07.

[GL06] Shuzhi Sam Ge and Frank L. Lewis, *Autonomous mobile robots: Sensing, control, decision-making and applications*, Boca Raton, FL: CRC/Taylor & Francis, 2006.

[GMAM06] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, *Path planning for mobile robot navigation using voronoi diagram and fast marching*, Proceedings 2006 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, October 2006, pp. 2376–2381.

[Gro07] Precision Products Group, *Hemisphere gps*, '`http://www.hemispheregps.com`', June 2007, last accessed 01/06/08.

[GWA01] Mohinder S. Grewal, Lawrences R. Weill, and Angus P. Andrews, *Global positioning systems, inertial navigation, and integration*, A John Wiley & Sons, Inc. Publication, 2001.

[Hei93] Loren Heiny, *Advanced graphics programming using c/c++*, Wiley, 1993.

[HSS03] Andrew Howard, Sajid Siddiqi, and Gaurav S. Sukhatme, *An experimental study of localization using wireless ethernet*, Proceedings fo the 4th International Conference on Field and Service Robotics, July 2003.

[Jon04] Joseph L. Jones, *Robot programming - a practical guide to behaviour-based robotics*, McGraw-Hill, 2004.

[JSPB07] Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard, *Autonomous exploration for 3d map learning*, Fachgespräche Autonome Mobile Systeme (AMS) (2007).

[KB91] Y. Koren and J. Borenstein, *Potential field methods and their inherent limitations for mobile robot navigation*, Proceedings of the

IEEE Conference on Robotics and Automation, 1991, pp. 1398–1404.

[KL02] S. Koenig and M. Likhachev, *Improved fast replanning for robot navigation in unknown terrain*, Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on **1** (2002), 968–975 vol.1.

[KLC+03] Seong-Baek Kim, Seung-Yong Lee, Ji-Hoon Choi, Kyoung-Ho Choi, and Byung-Tae Jang, *A bimodal approach for gps and imu integration for land vehicle applications*, Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th **4** (2003), 2750–2753 Vol.4.

[KLF04] Sven Koenig, Maxim Likhachev, and David Furcy, *Lifelong planning a**, Artif. Intell. **155** (2004), no. 1-2, 93–146.

[KWLG07] Sardha Wijesoma Kwang Wee Lee and Javier Ibanez Guzman, *A constrained slam approach to robust and accurate localisation of autonomous ground vehicles*, Robotics and Autonomous Systems **55** (2007), no. 7, 527–540.

[LAJ04] C. Leung and A. Al-Jumaily, *A hybrid system for multi-agent exploration*, Proceedings 2004 IEEE International Conference on Fuzzy Systems, vol. 1, July 2004, pp. 209–213.

[LFG+05] Maxim Likhachev, David Ferguson, Geoffrey Gordon, Anthony (Tony) Stentz, and Sebastian Thrun, *Anytime dynamic a*: An anytime, replanning algorithm*, Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), June 2005.

[LJ08] Christopher P. Lee-Johnson, *Emotion-based parameter modulation for a mobile robot planning and control system*, Ph.D. thesis, Victoria University of Wellington, 2008, Submitted in July 2008.

[LJC07] Christopher P. Lee-Johnson and Dale A. Carnegie, *Emotion-based parameter modulation for a hierarchical mobile robot planning and control architecture*, Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on (Oct. 29 2007-Nov. 2 2007), 2839–2844.

[LJCC07] Christopher P. Lee-Johnson, Praneel Chand, and Dale A. Carnegie, *Applications of an adaptive hierarchical mobile robot navigation system*, Proceedings of the 2007 Australasian Conference on Robotics & Automation (Matthew Dunbabin and Mandyam Srinivasan, eds.), 2007.

[Mac03] David J. C. MacKay, *Information theory, inference, and learning algorithms*, Cambridge University Press, 2003.

[Mat08] Mathworks, *Matlab - the language of technical computing*, "`http://www.mathworks.com/products/matlab/`", March 2008, last accessed on 28/03/08.

[MBT+06] Mark Maimone, Jeffrey Biesiadecki, Edward Tunstel, Yang Cheng, and Chris Leger, *Intelligence for space robotics*, ch. Chapter 3 Surface navigation and mobility intelligence on the Mars Exploration Rovers, pp. 45–69, TSI Press, 2006.

[Mic06] Microsoft, *Microsoft robotics studio*, "`http://msdn2.microsoft.com/en-us/robotics/default.aspx`", 2006, last accessed 20/03/08.

[Mic07] ———, *Ccr introduction*, '`http://msdn.microsoft.com/en-us/library/bb648752.aspx`', Aug 2007, last accessed 05/05/08.

[Mil06] Ian Millington, *Articificial intelligence for games*, 1rst ed., Morgan Kaufmann, 2006.

[Min08] Ministry of Civil Defence & Emergency Management, *New zealand search & rescue - urban search and rescue*, "`http://www.usar.govt.nz`", March 2008, last accessed 01/03/08.

[MM04] Javier Minguez and L. Montano, *Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios*, Robotics and Automation, IEEE Transactions on **20** (Feb. 2004), no. 1, 45–59.

[MOB07] MOBILEROBOTS, *Mobilerobots*, "`http://www.activrobots.com`", September 2007, last accessed on 01/09/07.

[MPV00] Mauro Di Marco, Domenico Prattichizzo, and Antonio Vicino, *approximation of uncertain height fields for outdoor navigation*, Proceedings of the 39th IEEE Conference on Decision and Control, 2000, pp. 839–844.

[MRH$^+$02] M. Montemerlo, N. Roy, D. Hahnel, C. Stachniss, S. Thrun, and J. Glover, *Carmen - the carnegie mellon robot navigation toolkit*, "`http://carmen.sourceforge.net`", 2002, last accessed on 10/07/07.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, *Fast-SLAM: A factored solution to the simultaneous localization and mapping problem*, Proceedings of the AAAI National Conference on Artificial Intelligence (Edmonton, Canada), AAAI, 2002.

[Mur00] Robin R. Murphy, *Introduction to ai robotics*, 1rst ed., MIT Press, Cambridge, Massachusetts, London, England, 2000.

[Neh03] Ulrich Nehmzow, *Mobile robotics a practical introduction*, 2nd ed., Springer, 2003.

[RCA07] Thomas M. Roehr, Dale A. Carnegie, and Peter Andreae, *Developing a robust control system for a team of autonomous mobile robots*, Proceedings of the Fourteenth Electronics New Zealand Conference, 2007, pp. 273–278.

[RN03] Stuart Russell and Peter Norvig, *Artificial intelligence - a modern approach*, 2nd ed., Prentice Hall Series in Artificial Intelligence, Prentice Hall, Upper Saddle River, New Jersey 07468, 2003.

[SD03] R. Sim and G. Dudek, *Effective exploration strategies for the construction of visual maps*, Proceedings 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, vol. 4, 27-31 October 2003, pp. 3224–3231.

[SDC07] Dongqing Shi, D. Dunlap, and E.G. Collins, *A comparison between a fuzzy behavioral algorithm and a vector polar histogram algorithm for mobile robot navigation*, Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on (20-23 June 2007), 260–265.

[SN04] R. Siegwart and I. R. Nourbakhsh, *Introduction to autonomous mobile robots*, MIT Press, 2004.

[Ste95] A. Stentz, *The focussed d* algorithm for real-time replanning*, Proceedings 1995 Intl. Joint Conference on Artificial Intelligence, August 1995, pp. 1652–1659.

[Sto75] Lawrence D. Stone, *Theory of optimal search*, Academic Press, 1975.

[SZ06] A. Sgorbissa and R. Zaccaria, *Nav: Navigation without localization*, Proceedings 2006 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, October 2006, pp. 1761–1766.

[TBF00] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d-mapping*, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, April 2000, pp. 321–328.

[TBF05] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, MIT Press, Cambridge, Massachusetts, London, England, 2005.

[TPB06] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard, *Multi-level surface maps for outdoor terrain mapping and loop closing*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2006, pp. 2276–2282.

[Tri07] Rudolph Triebel, *Three-dimensional perception for mobile robots*, Ph.D. thesis, Alberts-Ludwigs-University Freiburg, May 2007.

[TW97] D.H. Titterton and J.L. Weston, *Strapdown intertial navigation technology*, IEEE Radar, Sonar, Navigation and Avionics Series 5, Peter Peregrinus Ltd., 1997.

[UB98] I. Ulrich and J. Borenstein, *Vfh+: reliable obstacle avoidance for fast mobile robots*, Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on **2** (16-20 May 1998), 1572–1577 vol.2.

[USN03] Christopher Urmson, Reid Simmons, and Issa Nesnas, *A generic framework for robotic navigation*, IEEE Aerospace Conference 2003, March 2003.

[VDLM06] Kimon P. Valavanis, Lefteris Doitsidis, Matt Long, and Robin Roberson Murphy, *A case study of fuzzy-logic-based robot navigation*, IEEE Robotics and Automation Magazine **13** (2006), no. 3, 93–107.

[VM06] D. Vikerimark and J. Minguez, *Reactive obstacle avoidance for mobile robots that operate in confined 3d workspaces*, Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean (16-19 May 2006), 1246–1251.

[Wat93] Alan Watt, *3d computer graphics*, 2nd ed., Addison-Wesley, 1993.

[WC06] David A. Williamson and Dale A. Carnegie, *Embedded platform for search and rescue application*, Proceedings of the Intl. Conference on Autonomous Robots and Agents, 2006, pp. 373–378.

[Wik08] Wikipedia, *Sichuan earthquake*, "`http://en.wikipedia.org/wiki/2008_Sichuan_earthquake`", May 2008, last accessed on 03/08/08.

[Wil07] David A. Williamson, *The development of a mother agent for a hierarchical multi-robot urban search and rescue system*, Master's thesis, Victoria University of Wellington, 2007.

[Yam97] Brian Yamauchi, *A frontier-based approach for autonomous exploration*, IEEE International Symposium on Computational Intelligence in Robotics, 10-11 July 1997.

[Yam98] ______, *Frontier-based exploration using multiple robots*, Proceedings of the Second International Conference on Autonomous Agents, May 1998.

[Yan07] Xi Peng Yang, *Developing a real time tracking solution in wlan*, Master's thesis, Victoria University of Wellington, 2007.

[YMD08] Jack Yeazel, Joe Mehaffey, and Dale DePriest, *gpsinformation.net*, "`http://www.gpsinformation.net/main/altitude.htm`", January 2008, last accessed 06/01/08.

[YS07] Qingmei Yang and Jianmin Sun, *A location method for autonomous vehicle based on integrated gps/ins*, Vehicular Electronics and Safety, 2007. ICVES. IEEE International Conference on (2007), 1–4.

[ZM00] Paul Zarchan and Howard Musoff, *Fundamentals of kalman filtering - a practical approach*, Progress in Astronautics and Aeronautics, vol. 190, American Institute of Aeronautics and Astronautics, Inc., 2000.