Fachhochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Informatik und Medien

Deutsches Zentrum
für Luft- und Raumfahrt

Institute for Robotics and Mechatronics

Bachelor Thesis

Improving Local Navigation by Application of Scan Matching Techniques in Mobile Robotics

Created by: Jakob Hasse

October the 4th, 2013

Supervisors, FH Brandenburg: Dipl.-Inform. Ingo Boersch
Prof. Dr.-Ing. Jochen Heinsohn
Supervisors, DLR: Dipl.-Math. Christian Rink M. Sc.
Dipl.-Math. techn. Daniel Seth

Abstract

This thesis is about the use of registration algorithms for navigation of a mobile robot. Generally, the use of these algorithms for the localization and a modification of the navigation system are examined. The Lib3D framework of the Robotics and Machatronics Center of the DLR serves as experimental evaluation environment. The main goal here is to improve the navigation through narrow environments. In the current state the robot often corrects its position a few times until it drives through the narrow environment. Through a more accurate pose estimation the uncertainty before driving through a narrow environment should be lowered. So the robot can try alternative ways to not move and relocalize again. The used registration algorithms have already been implemented in the Lib3D framework. A concept is created that depicts how the algorithms should be applied and evaluated, in order to find the best one, which can be integrated into Lib3D. This concept serves as a rough plan for the implementation. Soon in this implementation it turns out that the ICP is the best choice for the integration. It is integrated into the navigation system of Lib3D and an exact evaluation is performed. On the one hand the ICP is mostly more accurate, on the other hand the results vary strongly, what makes them less meaningful. The Conclusion is that a good statement about whether the ICP should be used or not can not be made at this point. Further examinations are necessary.

Zusammenfassung

Diese Arbeit behandelt die Auswirkungen des Einsatzes von Registrierungsalgorithmen auf die Navigation eines mobilen Roboters. Hauptsächlich geht es dabei um den Einsatz dieser Algorithmen in der Lokalisierung und eine Modifikation der Navigationskomponente des Lib3D-Frameworks des Robotik und Mechatronik Zentrums des DLR. Speziell geht es darum, dem Roboter die Fahrt durch eine enge Stelle zu erleichtern. Mit der momentan implementierten Monte Carlo Lokalisation korrigiert der Roboter seine Position oft mehrere Male. So soll durch eine genauere Lokalisierung mithilfe der Registrierung die Unsicherheit vor dem Befahren der engen Stelle kleiner werden. Das soll dazu führen, dass dem Roboter mitunter alternative Fahrmanöver geboten werden, sodass er seine Position nicht erneut korrigieren muss. Die in Frage kommenden Registrierungsalgorithmen sind alle schon in dem Lib3D-Framework implementiert. Es wird ein Konzept erstellt, wie die Algorithmen angewendet und bewertet werden, um später eine Entscheidung zu treffen, welche sich für die Integration in Lib3D eignen. Danach wird anhand dieses Konzepts die Implementierung beschrieben, bei der sich frühzeitig herausstellt, dass der Iterative Closest Point Algorithmus bis jetzt die beste Wahl ist. Er wird in die Navigationskomponente von Lib3D integriert und getestet. Abschließend wird die Anwendung des ICP in Lib3D ausgewertet. Zwar ist die Lokalisierung mit dem ICP oft genauer, doch andererseits variiert sie sehr stark, wodurch die Aussagekraft der Auswertung kleiner wird. Eine generelle Entscheidung, ob der ICP an der Stelle sinnvoll eingesetzt ist, kann also nicht gefällt werden.

Contents

1 Introduction **3**
- 1.1 Goal . . . 4
- 1.2 Structure . . . 4

2 Related work **5**

3 Fundamentals **7**
- 3.1 Programming tools and frameworks . . . 7
- 3.2 Mobile Robot Environment . . . 8
- 3.3 Simulated hardware . . . 8

4 Concept **11**
- 4.1 Registration approaches . . . 11
- 4.2 Retrieving the data . . . 11
- 4.3 Application of the algorithms and rough evaluation . . . 12
- 4.4 Integration into MRE . . . 12

5 Implementation **15**
- 5.1 Example scene . . . 15
- 5.2 Retrieving the template data . . . 15
- 5.3 Retrieving the sensor data . . . 18
- 5.4 Application of the algorithms and rough evaluation . . . 22
  - 5.4.1 ICP . . . 22
  - 5.4.2 MCR . . . 24
  - 5.4.3 Image processing methods for depth feature calculation . . . 25
- 5.5 Integration of the algorithms into MRE . . . 25
  - 5.5.1 Base structure for the implementation . . . 25
  - 5.5.2 Determining the application of the registration . . . 26
- 5.6 Optimizing the input data of the ICP . . . 26
- 5.7 Changes of the navigation system . . . 28
- 5.8 Evaluation of the use of registration algorithms . . . 29
  - 5.8.1 Runtime . . . 29
  - 5.8.2 Achieved accuracy . . . 31

6 Conclusion **35**

List of Figures

3.1 The KUKA omniRob 9

4.1 Integration into the MRE 13
4.2 Navigation problem before narrow environment 14

5.1 The test environment 16
5.2 The random sampled points 16
5.3 The three chosen measurement positions 17
5.4 The template data 18
5.5 Normal and reduced classified feature points as template 19
5.6 The view fields of the depth sensors. 19
5.7 Calculated feature points of a measurement 20
5.8 Feature detection fails on edge 21
5.9 Features of image processing methods 21
5.10 Positions of the rough evaluation 22
5.11 Matching to the template data 23
5.12 Unsuccessful matching to the template data 24
5.13 Unsuccessful feature point matching 25
5.14 Structure of the algorithms 27
5.15 The navigation system 29
5.16 The analyzed ways 30
5.17 Times of way 1 and 2 30
5.18 The deviance 31
5.19 The angular deviance 32
5.20 The matching planes 34

1 Introduction

In the industry mobile robots today are about to step over the edge of being only research projects. They are on the way to become good applicable, reliable and safe systems that help human workers.

A key factor of mobile robots is their navigation system. The mobile robot navigation, as it is relevant for this thesis, consists of 3 tasks: localization, path planning and path executing. There exists another view that navigation is also the processing of a request of an external instance to go to a certain point or area [10]. But this is a very high level task and is not covered in this thesis. Furthermore, there is a distinction between *global* and *local* navigation. According to Batalin et al. [3], *global* navigation is the overall planning of the path from a start position to a goal position that may not be seen directly by the robot. A map of the global environment is necessary here. *Local* navigation is the steering of the robot in the near environment at the current time. What is *near* depends of the properties of the robot and its setting and has to be determined in respect to possible scenarios. Another important characteristic is that the navigation in this case typically is not complete (except for the case where the robot steers directly to its goal pose).

A very challenging part of local navigation is to drive the mobile robot safely, especially collision free, through a narrow area. Why this is so difficult can be observed when humans try to carry a bulky freight through a narrow door that is just wide enough. The humans do this very slowly and carefully intuitively. Collision free carrying is the ultimate goal here on the one hand. On the other hand humans suffer from uncertainty and inaccuracy. They want to be able to stop before a collision happens, even if they cannot observe all edges of the bulky freight simultaneously and their movements are not infinitely accurate. That explains why they become so slow. But humans mostly find a good compromise between velocity and safety.

Mobile robots suffer from uncertainty and inaccuracy, too. To respect that, mobile robots often become very slow, like humans in the situation above, when driving through narrow areas. That was observed by the author when working with a Pioneer 3 robot with the ARIA framework. When it traveled through door frames it became relatively slow. But even with relatively high side clearance (the clearance right and left of the robot to an obstacle) to the door frame the robot became very slow. The problem was not only the door frame itself, but also the high uncertainty of the robot localization. The simulation of the KUKA omniRob of the Robotic and Mechatronics Center (RMC) of the German Aerospace Center (DLR) shows problems, too, even though with a different behavior. If it is located before the narrow area (a simulated door frame) through that it should drive, it mostly relocalizes and moves again. That happens because the coordinates of its pose estimation are not close enough to a temporary goal. But the robot has to be

very precise at this point because the path to the next temporary goal goes through the narrow area. So the robot often needs a few attempts until it is accurate enough. What both situations have in common is the uncertainty of the pose estimation that is, amongst others, slowing down the driving velocity or forces the robot to move closer to the current way point. It seems that a more accurate pose estimation could help to solve this problem and to find a better compromise between velocity and safety.

1.1 Goal

A common localization technique of mobile robots today is the Monte Carlo Localization (MCL) [11]. It became very popular because it is fairly accurate and robust enough for most good accessible areas. However, the given examples above work with the MCL and show that it sometimes is too inaccurate for the conditions of a narrow area. The idea of this thesis is to focus on the application of registration algorithms in combination with MCL while moving through narrow passages. One main task is to analyze the impact it has to the pose estimation and thus to the navigation system of the robot. If the registration algorithms will make the average pose estimation on each step more accurate, the navigation system will be changed to benefit from the more accurate estimation.

The goal is to improve at least one of the two aspects:

1. Collision free driving
2. Driving velocity and run time of the navigation algorithms.

The driving velocity and the run time of the navigation algorithms can depend on each other: It may be that the robot stands still on a place and waits for an excessively long running algorithm so that it loses time at that moment. The time the robot needs may also be outperformed by later time savings because of a more accurate initial pose estimation.

Dynamic environments will not be treated here.

1.2 Structure

The structure of this thesis is as follows: first the fundamentals of this thesis will be described, including related work, a description of registration and the used soft- and hardware. Then, the algorithms for the application will be introduced and a concept will be developed for extracting and converting data and using the algorithms. After that, the actual implementation of this concept follows. Finally, there will be a conclusion, evaluating the three depicted ways of registration. The runtime performance, speed of the robot and the applicability by comparing all with the MCL will be analyzed.

2 Related work

Registration or scan matching in mobile robotics is mainly the matching of two shapes, which have to be partly or fully overlapping. The terms registration and scan matching are equivalent in this thesis. The shapes usually are either 2-D or 3-D, depending on the sensors and their arrangement in the space. In this thesis 3-D data will be used for the registration. The navigation of the robot, in contrast, will be 2-D. Two registration algorithms will be used.

A widely used registration algorithm is the Iterative Closest Point algorithm (ICP) [4]. The ICP is a local method for matching and aligning two shapes. So an initial guess of the pose of the data is necessary. Rusinkiewicz and Levoy provide an overview of efficient variants [9]. Another used algorithm is the Monte Carlo Registration MCR [6]. It is a global registration method that is intended to work with reduced and classified feature points of a template and data set. It searches the complete rotational space for the most accurate match of template and data and then determines the translation. The search of the rotations is performed by a particle filter. A reduction drops the majority of the points and only the most important ones remain. Correspondences between two points are only assigned if the points belong to the same feature class. This minimizes the runtime of the algorithm.

Scan matching to improve the pose accuracy has already been used with 2-D data. Röwekämper et al. [7] propose a scan matching approach with the ICP, by taking the position guess of the MCL as initial pose estimation. The results are promising, the error of the system never exceeded 17mm/0.53 deg at all, even not in dynamic environments. Most of the times the position error was only a few millimeters. Scan matching is used to improve the accuracy of an already roughly reached pose. After reaching that pose, it is not of importance anymore. In contrast, the proceeding described in this thesis should improve the accuracy of the estimated pose to enhance the driving of the robot to a next goal. Also, the idea in this thesis is to integrate the out coming data of the scan matching process back into the MCL so that new particles can be sampled. Another difference is the measurement of the real robot poses for the evaluation. Röwekämper et. al. use an external precise motion capture system, while the measurement used in this thesis will be based on the data provided by a simulation.

3 Fundamentals

This chapter will list the programming tools, the frameworks and the simulated hardware that this thesis relies on.

3.1 Programming tools and frameworks

The programming language was C++. The tools used for creating and modifying source code were a DLR version of Eclipse and the gcc for compiling. The most important frameworks for this thesis were the Lib3D, OpenCV and the BOOST Test Library.

Lib3D

The framework used for solving the robotic tasks was the Lib3D (L3D) framework, developed by the DLR-RMC. It is a robotic framework providing storing and processing of data from a mobile robot and its environment and many related tasks. Especially, it contains some of the most important components for mobile robotic tasks: Localization, path planning and execution, mapping and sensor simulation. It also contains implementations of the important algorithms used in this thesis, the ICP and the MCR.

The navigation system of L3D is organized by several steps:

1. At first, a graph, based on the map, is generated, that contains a set of possible way points for the trajectory of the robot.
2. The robot localizes itself with the MCL.
3. Then an optimal path is searched along the way points of the graph and the first way point is set as current way point.
4. The robot tries to move to the current way point, then localizes itself and checks whether it is close enough to the current way point.
5. If it is not close enough, it tries to move closer to the current way point and continues with step 4. Otherwise it goes on with step 6.
6. If the current achieved way point is the goal, the navigation is done and the program returns. If not, the robot sets the next way point as current way point and continues with step 4.

Two poses are essential in mobile robotics and have to be considered: The real pose of the robot and the pose estimation of the robot. The pose of the robot in the real world is

usually unknown since there is no way to receive it directly. If the robot is simulated, as is always the case in this thesis, the real pose is known. Hence it will be called SimPose because it is generated by the *simulator*. The estimated pose will be called MLPose from then on. It is the *M*aximum *L*ikelihood estimate of the MCL. Generally, there could also be other localization algorithms than the MCL which is used in L3D. The robot perceives the environment through the sensors and then estimates its pose with the MCL, which is using sensor data. The pose accuracy depends on many factors such as the underlying algorithms, the sensor and map data quality, sensor calibration, the structure of the environment and the sensor coverage of the environment, to name a few examples.

L3D contains several data types for data storage. Most algorithms that are implemented in L3D use this storage types. On of them is the point storage, which stores points and is required for the ICP, for example.

BOOST Test Library

The BOOST Test Library is a unit testing framework that was used for testing some parts of the newly developed components for this thesis. It is part of the BOOST library and, amongst other things, supports easy test case implementation, test suites, fixtures and can expect exceptions [8].

OpenCV

OpenCV is an image processing framework for multiple platforms that has a C++ interface, amongst others [2]. In this thesis a combination of Blur, Canny and Dilation filters, applied in this order, was used. The Canny filter can detect edges in an image.

3.2 Mobile Robot Environment

The Mobile Robot Environment, hence called MRE, is essential for this thesis. It relies on the L3D framework and forms the basic user interface for working with a robot or its simulation. It is a C++ project that assembles all the necessary parts of the framework together. The simulation is one of the possible modes to execute MRE. While the MRE is starting, parameters can be passed to set additional features or to let the robot perform certain tasks.

A separate viewer exists to make the robot and its environment visible.

3.3 Simulated hardware

The hardware on which the MRE should run in the future is the KUKA omniRob. It is an omnidirectional mobile robot that serves as research platform. As can be seen in Figure 3.1, it has a light weight robot arm for manipulation purposes. The standard sensors are 2-D laser scanners. The omniRob has been extended DLR. One of these is fundamental for this thesis: It has 8 additional depth sensors, each with a 64 x 50 pixel resolution.

Figure 3.1: The KUKA omniRob without (left) and with (right) the modifications of the DLR. Source: DLR internal

4 Concept

This section proposes three solutions to the problem of making the position estimation more accurate. It will also propose a plan for the data acquisition as well as the steps that will be made until the evaluation can be done in MRE. Then it will describe how an optimization of the navigation system can be achieved with the help of a more accurate pose estimation.

4.1 Registration approaches

The three proposed approaches are mainly chosen as the underlying algorithms were already implemented in the L3D framework. It shall be examined whether the pose estimation becomes more reliable and accurate by the application of them. Furthermore, the impact the approaches have on the behavior of the robot shall be evaluated.

1. The first approach consists of the ICP. The necessary initial pose estimation can be delivered by the MCL. Because the robot translations are only in 2-D, an ICP variant will be used that only takes rotations around the Z-axis into account.

2. Another approach is based on using the MCR.

3. The third approach uses feature detection directly on the depth images. After the features are detected, either the MCR or the ICP can be applied for the position estimation.

4.2 Retrieving the data

In general, all algorithms need two data sets: the known template data from the map and the sensor data.

The template data is always a point cloud of the narrow area of the test environment (Figure 5.1). It already fits for the ICP that only needs a plain point cloud. Additionally the MCR is optimized for (reduced) feature points, so its point cloud template gets extra treatment to extract feature points. Afterward the feature points may be reduced.

The first possibility to get point clouds from the model is to sample points randomly on the surface of all polygons. A second possibility is to simulate depth sensor measurements of the polygon model. Generally all data sets have to be cropped so that only the points near and in the narrow area retain. This enhances performance and also sets the focus only on the narrow area.

Finally, the sensor data for the matching process has to be retrieved. This is also done by simulating depth sensor data. To create a realistic simulation, it has to be inaccurate. Otherwise the registration would not make any sense. For best results the depth images are taken out of the robot simulation in a run of the MRE. Since they contain the real coordinates, they get the coordinates of the current MLPose, as if the robot created the measurement on the basis of the MLPose.

4.3 Application of the algorithms and rough evaluation

Because the data often has to be transformed and converted before it is ready to use, several binaries have to be used to make the data fit to the final algorithm. For example, before a depth image can be matched with the template data by using the ICP, it has to be converted into a point cloud first. To check how well the ICP performed, the output of it, a transformation matrix, has to be applied to the depth image. Then the differences can be shown in a viewer program.

The rough evaluation is done by visual inspection of the result compared with the template data in a viewer. As a result there shall be one or more algorithms that fit best for this problem.

4.4 Integration into MRE

The algorithms resulting from the rough evaluation will be integrated into MRE, together with the enhancements of the navigation system. The MRE is will not only be used with the simulation but also with the real robot, which weights almost 400 kg. Furthermore other people at the DLR-RMC will work with the MRE and thus with the created software. So there are certain requirements for components that are developed:

1. Functionality: The program has to fulfill the functionality, in this case the registration and the enhancement of the navigation system, correctly.

2. Stability: Software crashes have to be avoided at all time. Especially when the MRE runs on a real robot, crashes and bugs can have catastrophic consequences.

3. Changeability: The algorithms can be changed and replaced easily, even at runtime.

4. Maintainability: The source code of the components can be easily understood, so that small changes and debugging can be done with minimal costs.

5. Testability: The components can be tested with minimal costs, for ensuring the functionality and for comparing the algorithms.

Point 1 and 2 can be achieved by using unit tests, even though it is (nearly) impossible to achieve error-free code. The use of the design pattern *Strategy* provides points 3, 4 and 5 because it encapsulates the whole algorithm and makes it interchangeable [5]. How

it does this will be discussed in detail in section 5.4. Documenting the code properly is very important for point 4.

The template data for the narrow areas will be stored with global coordinates. So it can be used like a global map. If the registration algorithm is applied, this template will be passed to it, together with the sensor data and the current estimated MLPose from the MCL. The registration algorithm should try to match the sensor data with the template. The resulting transformation will be applied to the passed MLPose which will be passed back to the particle filter of L3D again.

For all the algorithms described so far, the decision has to be made in every step, whether one of them should be applied or not. To reduce CPU run time and power consumption, the algorithms should only be applied when the robot is near the narrow area. Figure 4.1 shows the movement-update cycle of MRE with the registration algorithm option: as initial guess of the pose, the MCL is used. It returns the MLPose that is used as input if the registration algorithm shall be applied. If yes, then the registration algorithm takes that position and returns a new position that shall be more accurate. If not, the movement-update cycle goes on normally and the pose of the MCL remains valid as the most accurate one.

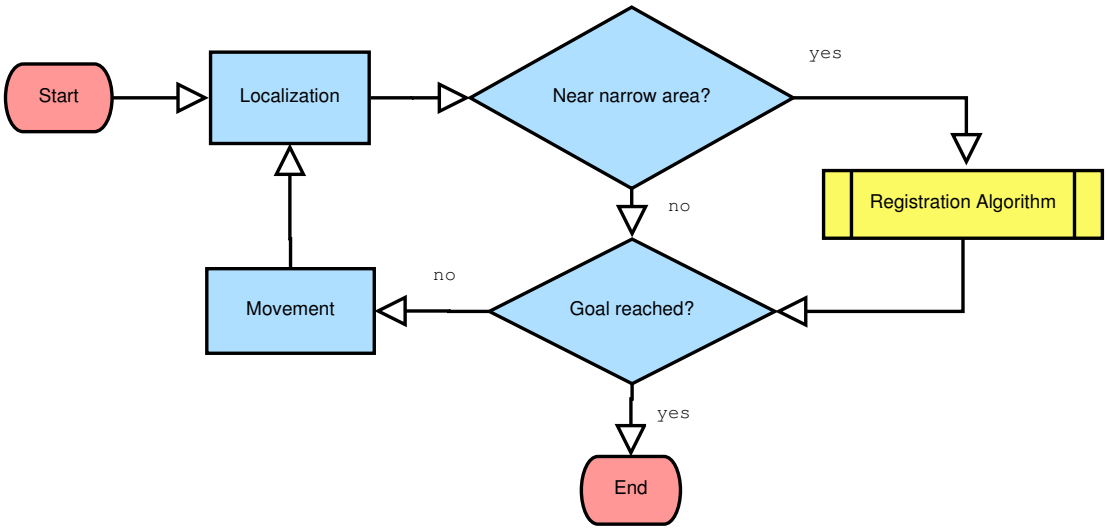

Figure 4.1: A simplified diagram of the integration of the registration algorithm into the MRE.

To evaluate the impact of the newly used registration algorithms, a few modifications will be made to the MRE. At first a time measurement will be included to see how the runtime will change. Furthermore, there shall be a measurement of the accuracy of the position. A simple approach can be to just take the deviations of the position of the robot in the simulator and the position estimation of the registration. On the one hand there is the euclidean distance between them. On the other hand there is the deviation of the angles.

For suiting the need of flexibility and testability, the design patterns of the Gang of Four [5] were considered. After a coarse study of the them, the Strategy design pattern was contemplated for solving that task. The testing shall be done with the C++ BOOST Unit testing framework.

If one or more of the algorithms succeeds, the navigation system can be optimized for situations depicted in Figure 4.2. The figure shows a situation in which the robot is coming from below left and tries to reach a given way point of the planned path (black). After reaching the way point, a next way point on the right-hand side (not in the picture) is tried to be reached. Three possible ways for this situation are depicted, each marked with a different color. The diamonds represent way points and the arrows possible movements of the robot, together they form a possible path. The light pink circle is a range around the given way point the robot tries to reach. Its size depends on the side clearance the robot has to achieve on the way to the next way point on the right side. If the robot's MLPose is within that circle after it tried to reach this way point, it goes on to the next step. The robot then tries to reach the next way point of the planned path. If the robot's MLPose is outside of that circle, it first tries to reach the currently given way point again until it is within that circle. One example is the red way in the Figure. But there are situations when the robot's position is outside that circle while the robot could drive safely to the next way point nevertheless. Such a situation is the green way. There the robot can go ahead to the next planned way point without violating the critical side clearance.

Another situation is when the position of the robot in the reality is already within the circle but the MLPose is outside because of its lack of accuracy. In this situation a more accurate position estimation could help too to prevent the robot from correcting its position and relocalizing again.

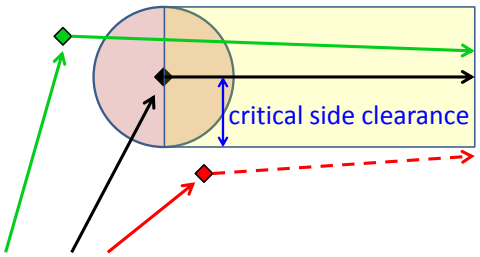

Figure 4.2: Navigation problem in front of the narrow environment.

5 Implementation

This section describes the realization of the preceding concept. Therefore it is also divided into three major steps: The retrieving of the template data covers the two steps stated above, trying to sample random points on a surface model and trying to simulate depth measurements. After that, there is a description of the application of the basic algorithms (ICP, MCR and OpenCV algorithm) to the chosen template data. Finally, there will be the integration of the best fitting algorithm(s) and template data into the MRE and a detailed evaluation of them.

5.1 Example scene

For evaluation concerns, a map of the DLR Mobile Laboratory was used for generating data on which the algorithms could be applied (see Figure 5.1). The format of that model was the open inventor file format [1]. It has a narrow area simulating a door frame. Another narrow area is the door at the far end of the laboratory. A third one is the floor beneath it where there is a box, blocking part of the environment.

5.2 Retrieving the template data

As stated in the Concept section, there are two possibilities for retrieving the template data: Sampling random points on the surface of the model of the testing environment or simulating depth measurements on it.

Sampling of random points

The file format of the model of the test environment is open inventor, at first it has to be converted into an L3D specific file, containing only triangles. The conversion processing is performed within the L3D framework.

Then points are sampled over each triangle. To have an equal share of points over the whole added up size of the surface of all triangles, a sampling factor is introduced. The sampling factor is multiplied with the surface of each triangle. The resulting number is the quantity of sampled points on the current triangle.

Figure 5.2 illustrates the point cloud of the test environment. Two major problems occurred with this model. The first one is obvious: the share of the points produced by the sensors was different. Vertical planes are covered by more points per square meter, while horizontal planes such as the floor have a relatively low coverage. The initial idea was to try this kind of template anyway. Indeed it turned out that the ICP was matching well in that model.

Figure 5.1: This is the test environment for the robot. The first narrow passage is marked.

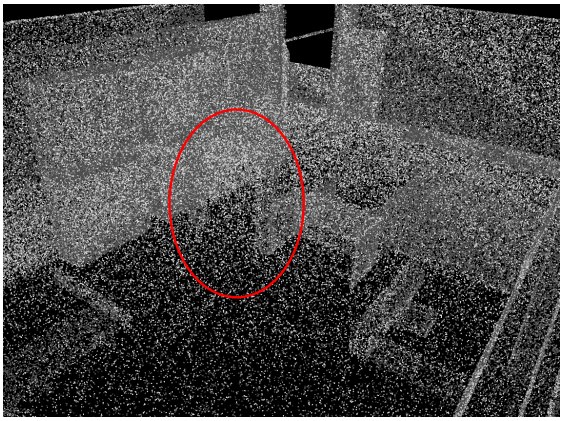

Figure 5.2: The random sampled points of the test environment. The narrow area is marked.

The other problem occurs if a plane is not visible. Nevertheless points are sampled on this plane. Data points, always coming from planes that can be seen, might associate with the points of the invisible plane. These false matches can disturb the ICP, as well as the MCL and should be avoided. Since the weakness comes from the calculation method itself, it was decided to go over to another method at this point.

Simulating depth measurements

The simulation environment of L3D has a simulator for depth images. After the preceding approach failed, it was used for the simulation of depth images to get the template data.

The idea was simple: Three basic measurement points were chosen manually.

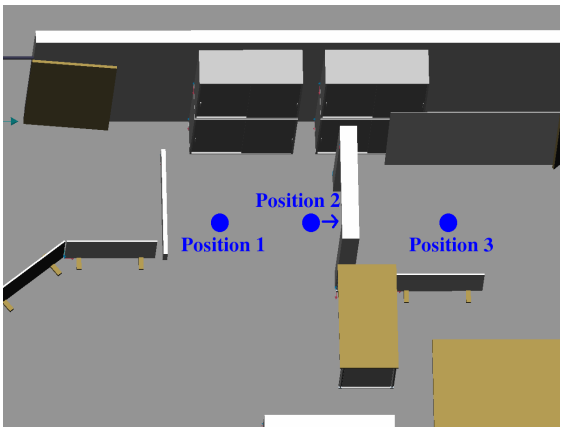

Figure 5.3: The three chosen measurement positions.

In Figure 5.3 three positions are depicted. There was one major reason for that: The robot positions in front of the obstacle could not be predicted. But we assume that the robot will in fact move through the narrow environment. When it arrives, it will probably take the first measurement while standing in front of the obstacle as for position 1 or 3. In addition, it will probably take one measurement while standing in the narrow environment. So the three positions shown in Figure 5.3 were chosen. Of course it can be at a quite different place, but if the template model is dense enough and covers most corners, that would not be a big problem.

Then 12 measurements were done from every point with the measurement simulator, heading around the point in 30 degree steps. Each measurement consisted of 1 depth image. So a sufficiently dense template with global coordinates could be created. The robot simulation was not used here to get a reasonable sensor coverage of the environment. At this point the measurement simulation for depth images did not add Gaussian noise to get a plain template. In contrast the measurement simulation of the robot in MRE will add Gaussian noise to simulate realistic depth images. When this step was finished, the data was cropped: Only points that were closer than 1.5 m to the narrow area remained. For determining that, only the X- and Y-coordinates of the points and the narrow environment were considered. So the environment in which the points around the

narrow area remain is cylindrical. The final step was to save the template, as depicted in Figure 5.4, in a file.

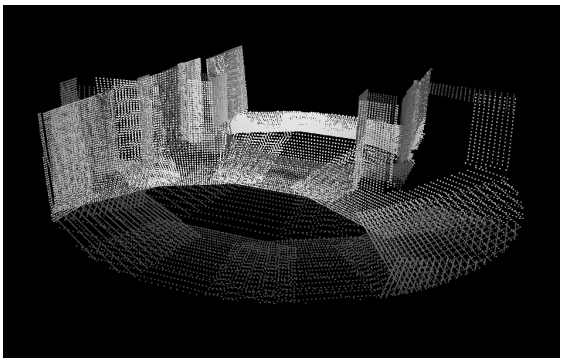

Figure 5.4: The template data (compare Figure 5.1).

Obviously the density of points on the floor is now different to that of the walls, for example. Another important fact is the elimination of most of the planes that are invisible for the sensors and would only lead to false matching. This includes the parallel planes of points too. These were the reason for the undesirable artifacts of the feature calculation described in the preceding section. Above all, this data is more realistic than the simply random sampled points because it is similar to the data coming from the sensors. Because scan data matches best with template data that is equally shaped, this template was chosen for the application of the registration algorithms.

For the MCR a template which only contains reduced feature points was calculated. It is based on the same template that was just produced for the ICP and the calculations were done by programs of the L3D framework. The feature calculation is based on the curvature each point builds with its neighbors. At first feature points were calculated on the template (Figure 5.5 left). For the reduction the feature points were classified, depending on the feature value, while 7 classes were used. Then the reduction process drops points depending on a threshold ratio and their feature value. In this case the threshold was 0.1, which means that only 10 percent of the points remain. The points of different classes are treated differently. The more points a class has, the more points are reduced. As a result, only the most important points, thus the points with the rarest features, remain.

Because this template is much more realistic than the randomly sampled points, it was chosen for further evaluation with the registration algorithms.

5.3 Retrieving the sensor data

For the ICP the depth images coming from the sensors in the simulation just had to be converted into a point storage. A converter in L3D did that work.

The first step of the data for the MCR was the same like for the ICP. But then features had to be calculated on that point cloud. The calculation became problematic since the

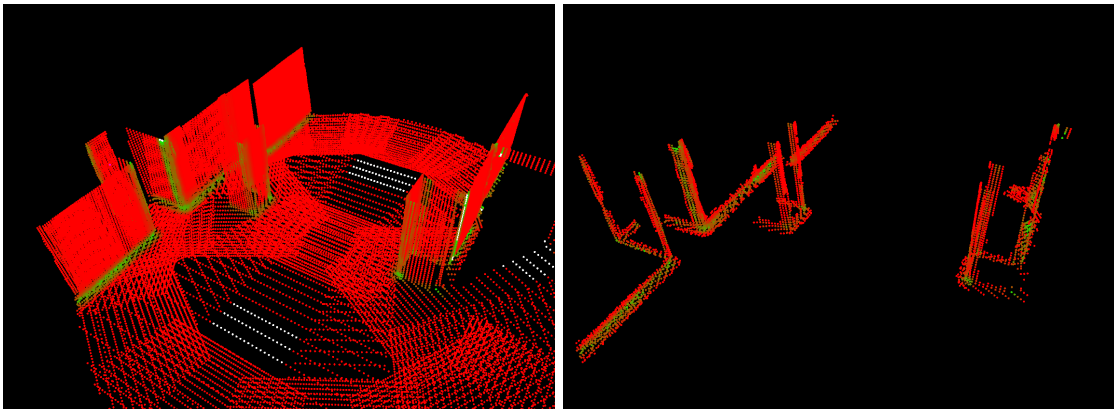

Figure 5.5: The left picture shows the feature points, calculated out of the normal template point cloud. The right one shows the classified and reduced feature points. The color mapping is continuous and ranges from red for low curvature features to green for high curvature features.

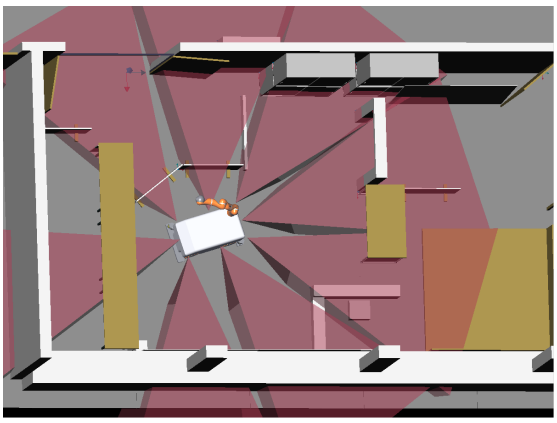

Figure 5.6: The omniRob in the simulation. The view fields of the depth sensors are marked red transparent.

sensors do not have a full coverage of the environment. Especially the very near areas have a poor sensor coverage (see Figure 5.6). If an obstacle is close to the robot, it is likely to remain only partly seen (or even unseen).

In the right picture of Figure 5.7 very few remaining points can be seen in the reduced template cloud. Actually it is meant to drop many points and let only the most important remain. This should be the advantage of the feature calculation because less points means less runtime. But since the sensors do not have a good coverage of the near environment (Figure 5.6), they often miss areas where the most important features are.

Another problem was located in the underlying feature calculation based on geometric relations between the points. It appears when a depth image is taken across the edge of an obstacle like in Figure Figure 5.7, while the other side of the edge is not visible for the camera.

Within the red marked area an edge of the obstacle in Figure 5.7 (part of the simulated door frame) can be seen that does not have correct features in the calculated feature point cloud. In the reduced feature point cloud the points in that area, although very important, are discarded. The features at the edge actually have to be edge related. But the other side of this edge is not visible, so they are discarded because the feature calculation algorithm needs points in the near environment, otherwise they became plane related. Figure 5.8 shows that schematically: You can see that the vertical side of the obstacle is on the blind side of the sensor and thus not covered with points. In this instance the yellow marked points get wrong (plane related) features. Generally they can also get no features at all this situation if they have too few neighbors.

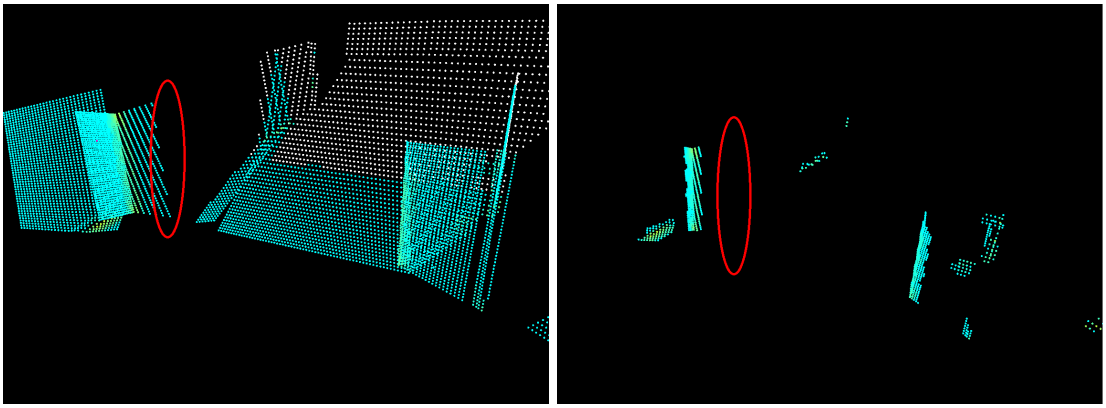

Figure 5.7: The left picture shows all the calculated feature points of a depth measurement the right only the classified and reduced ones. The color mapping of the measurement data is light blue for plane related features, moving on to yellow for edge related features. Points that do not have a feature are white.

The image processing methods for feature calculation directly on the depth images was actually meant to solve that problem. In fact the proposed filter could detect exactly these edges the geometric feature calculation could not find. But this were almost the only features detected by this method, what can be seen in Figure 5.9. Especially the

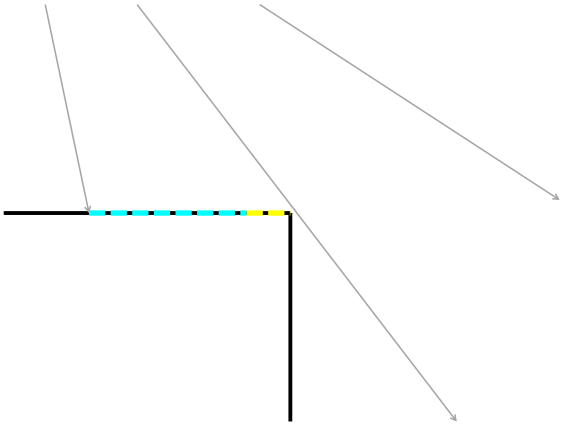

Figure 5.8: You can see the beams (Grey) of the camera hitting an obstacle (black) partly. The broken line represents the points that appear. Light blue means normal features can be calculated later. Yellow means wrong features will be calculated later or no features at all.

normal concave and convex edges of obstacles that are fully covered do not get features.

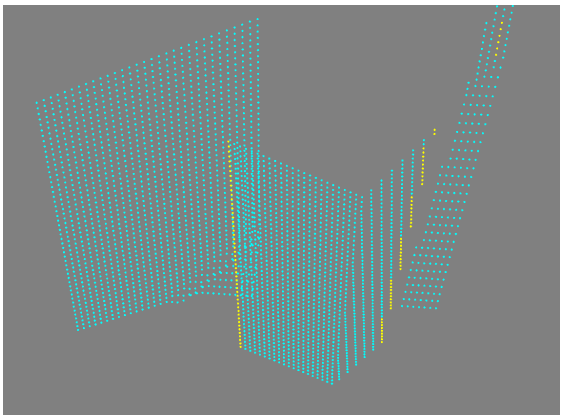

Figure 5.9: The feature values that are calculated directly on the depth image with image processing methods. Points corresponding with edges are yellow, all other points are light blue.

Both, the geometric feature calculation and the image processing method, seem to complement each other. The combination of both methods is not covered in this thesis. Nevertheless it could be the base of a future proceeding and should be evaluated.

5.4 Application of the algorithms and rough evaluation

At the beginning this part mainly consisted of using several binaries for processing the data and then applying and evaluating the algorithms. Later it turned out that it was sufficient to implement components directly and then just test them with Unit tests. The design pattern *Strategy* isolates the component with the specific algorithm, so that it does not have to rely on external dependencies. Generally spoken, it provides *loose coupling*. The result is an easy testable component, amongst others. Besides, unit tests are highly automated, many test cases can be written with less expense. All test cases can be run altogether or specific test cases can be run - each with only one call. They can be used more flexibly than testing with binaries and the tests are faster to use and reproduce.

Figure 5.10: The robot positions planned for the rough evaluation. The narrow area is in the center. From these positions the depth images were taken.

For the evaluation different positions of the robot, facing the obstacle, were chosen. In Figure 5.10 you see the general positions that were used for the evaluation. They are only approximated, the used positions in the reality differ from them. That is because you can not simply reproduce realistic data on a certain position. In the reality the robot has knowledge from the previous data acquisitions and position estimations. Thus only an approximate guess is possible, if the SimPose has to be on a certain position. You can set the SimPose and shift and twist the measurements taken from that SimPose. That was also done for creating extreme test conditions for the registration. However, the normal deviance of the position and the angle were mostly not enough to irritate the ICP.

5.4.1 ICP

The proceeding for the ICP was relatively simple compared to the other ones. It just takes two normal point clouds, one as template, another one as data for matching with that template. The template of the previous section (see section 5.2) was used and the

data was extracted from a run in MRE.

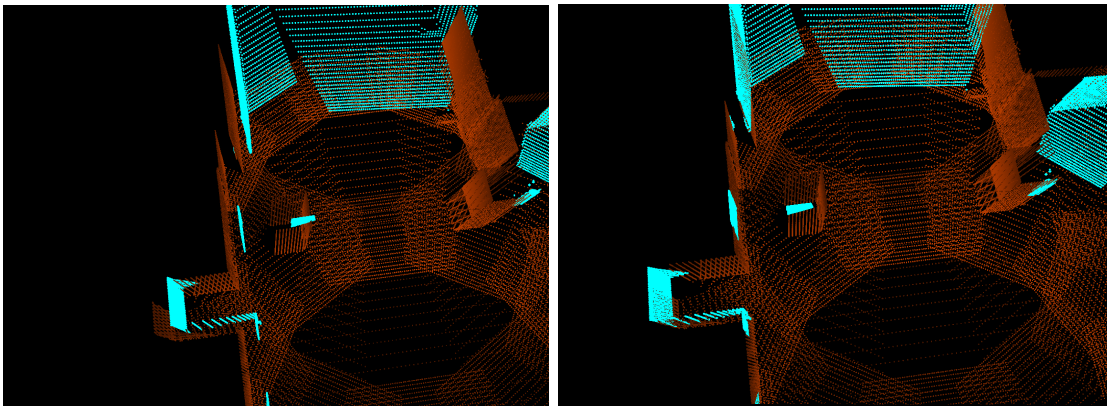

Figure 5.11: The camera and template data before and after matching with the template data. The template data is shaded red and the matched camera data is shaded light blue.

Table 5.1: This shows the values of the positions, corresponding to Figure 5.11. The Z values are left out because the movements did only happen in 2-D. MLPose is normally set by the MCL, but if registration is applied, it is set by the registration.

	X in mm	Y in mm	angle in degrees
SimPose	3864.27	8124.88	153.24
MLPose	3960.78	8156.07	153.66
MLPose after applying ICP	3864.06	8125.3	153.24

Figure 5.11 shows a typical situation of the robot standing in front of the narrow area at Position 1. The data is taken right out of the simulation in the MRE. As can be seen in the corresponding Table 5.1, a shift on the X-axis of around 96 mm and a shift on the Y-axis of around 31 mm is visible. There is also a little twist smaller than one degree. All three components match very well after applying the ICP. The deviance of the new MLPose are smaller than 1 mm and 0.01 degrees.

This was the case for most situations of the simulation that were examined. Only in a minority of the situations there were bad results in the tests. An example can be seen in Figure 5.12: In this case only three camera measurements are corresponding to the template point cloud at all. Unfortunately two of them only overlap with the floor and not with the walls. That is a problem since the ICP absolutely needs valid correspondences to walls to make a right shift in the horizontal directions. The floor is flat horizontally, so it does not make a big difference where the points of the measurement match there. The only measurement where points can correctly match with the template points is the

one that you can see in the center left. It bears the whole load of horizontal matching what is obviously not enough at that point.

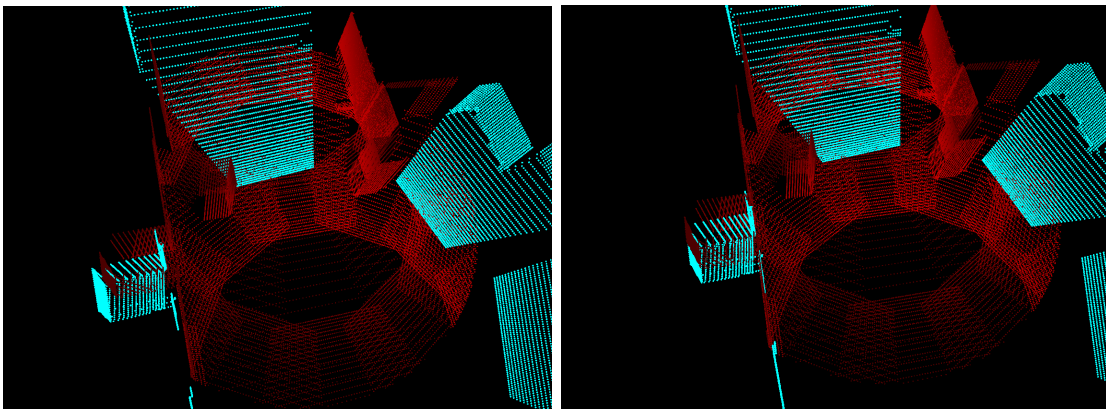

Figure 5.12: The camera and template data before and after an unsuccessful matching with the template data. The template data is shaded red and the matched camera data is shaded light blue.

The result can be seen in Table 5.2: The shift on the X-axis is overcompensated, while the shift on the Y-axis and the angle are reduced in the right direction after all.

Table 5.2: This shows the values of the positions, corresponding to Figure 5.12. For further details refer to the description in Table 5.1.

	X in mm	Y in mm	angle in degrees
SimPose	3952.49	7541.55	-86.51
MLPose	3917.89	7450.02	-83.89
MLPose after applying ICP	3970.37	7493.82	-84.10

However, these situations did not occur often in this preliminary application. So it was decided to take the ICP as a candidate for the integration into MRE.

5.4.2 MCR

Because of the two major problems described in section 5.3, the MCR failed to deliver correct transformations. Mostly they were not even close to the simulated position of the robot or to the position the MCL was estimating. This can be seen in Figure 5.13. The initial SimPose and MLPose estimation for the data on the right picture are the same as in Figure 5.11 / Table 5.1. On the right picture both pose estimations can be seen after the application of the MCR and the resulting transformation to the MCL. The data set is far more inaccurate than before the application.

Thus far, the MCR was not chosen for the integration into MRE.

However, the mentioned problems were also the reason for choosing the third proceeding, the calculation of features directly on the depth images.

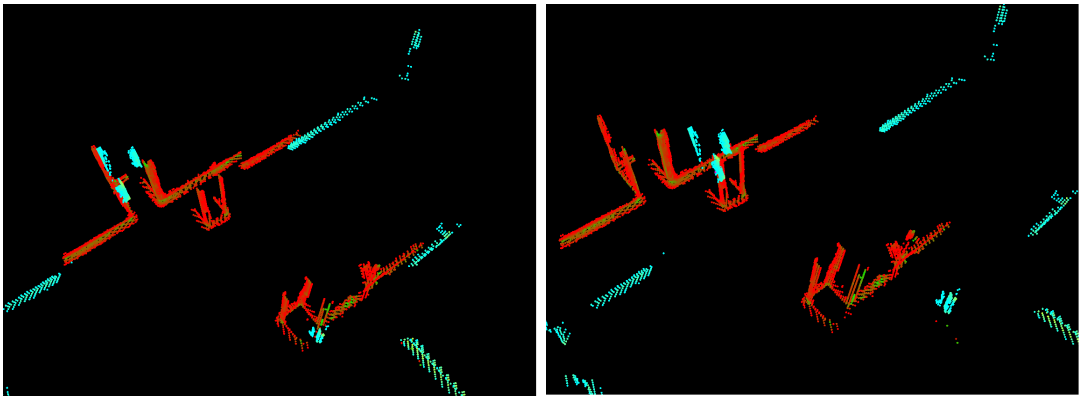

Figure 5.13: An unsuccessful feature point matching. On the left the initial pose of both data sets is depicted, on the right the pose estimation of the MCR is applied the the measurement data. The template feature points are red to green. The measurement points are yellow to blue.

5.4.3 Image processing methods for depth feature calculation

This approach should have been divided into two steps. The first one would have been taking the depth image and calculating features directly on it with image processing methods. After the features were detected and the underlying depth image was transformed into a feature point cloud, the usual algorithms like MCR or ICP could have been applied. But because the problems depicted in section 5.3 showed up so early, an application of the image processing methods was obsolete.

5.5 Integration of the algorithms into MRE

Finally, the MRE should be used for an evaluation of the robot's behavior within a real world simulation. This includes the integration of the registration algorithms as well as the measurement of the deviance, as stated in section 4.4. At the beginning of this section, the basic structure for the integration and the data storage will be explained. Then a mechanism for determining when to apply the algorithm and an improvement will be shown.

5.5.1 Base structure for the implementation

A good structure is important because it enhances readability, reusability and testability. As proposed in section 4.4, the *Strategy* design pattern was used for the integration because of several advantages in flexibility and testing. Instead of implementing all the

algorithm-specific code in the class CommandLoop, it only has a variable pointing to a concrete class that contains the algorithm-specific code. Figure 5.14 depicts this variable of type `RegAlgorithmICP`, stored in the class CommandLoop.

But only implementing the algorithm is not enough. The function that applies the registration, `invokeRegAlgorithm()`, takes a special storage type for the passed depth images. Therefore an extra function was developed for converting the depth images to points by iterating through the list. It also does the optimization, if depth images are not necessary (see section 5.6).

For the evaluation there were done some important changes too. First a time measurement was introduced for the method that does the steering from a start point to a goal point (for example results, see Figure 5.17). Another issue was the measurement of the deviance of the MLPose (before and after applying the ICP) and the SimPose. That was also done in the class CommandLoop because it has easy access to the position of the robot (MLPose and SimPose) and to the corrected MLPose of the registration.

The template data is stored in a class called Obstacle (see Figure 5.14) were it is read at the initialization of MRE. Obstacle mainly aggregates the map and path data. It is a good accessible data structure and so the template data is fitting well.

Getting the point storage's needed more processing because the cameras only provide depth images whereas the registration algorithm implementations of L3D need point clouds. In addition, these depth images are stored in a list instead of the DepthImageStorage the registration algorithms and the conversions expect. Therefore a function was implemented to iterate through the list. While doing that it first looks whether the current depth image is needed at all (see section 5.6). If yes, the depth image is copied into a given DepthImageStorage. After that the depth images could be converted to point clouds.

At runtime the CommandLoop picks the template data out of the Obstacle and passes it, together with the current measurement data, to the chosen algorithm.

5.5.2 Determining the application of the registration

Now a basic structure is determined as well as the data calculation and storage. When MRE runs, the registration algorithm has to be triggered when the robot is near enough to the obstacle.

The first try then was to take the euclidean distance between the robot's MCL position and the obstacle's position as a measurement for deciding whether to apply the algorithm or not. This is very easy to implement, but can also cause trouble. If the robot drives near a narrow area but does not go through it, the algorithm possibly is applied but not needed. That would result in more runtime without any benefit.

5.6 Optimizing the input data of the ICP

The ICP implementation of the DLR already has an optimization: There is a radius for the correspondences of each point. Another potentially correspondent then is only taken

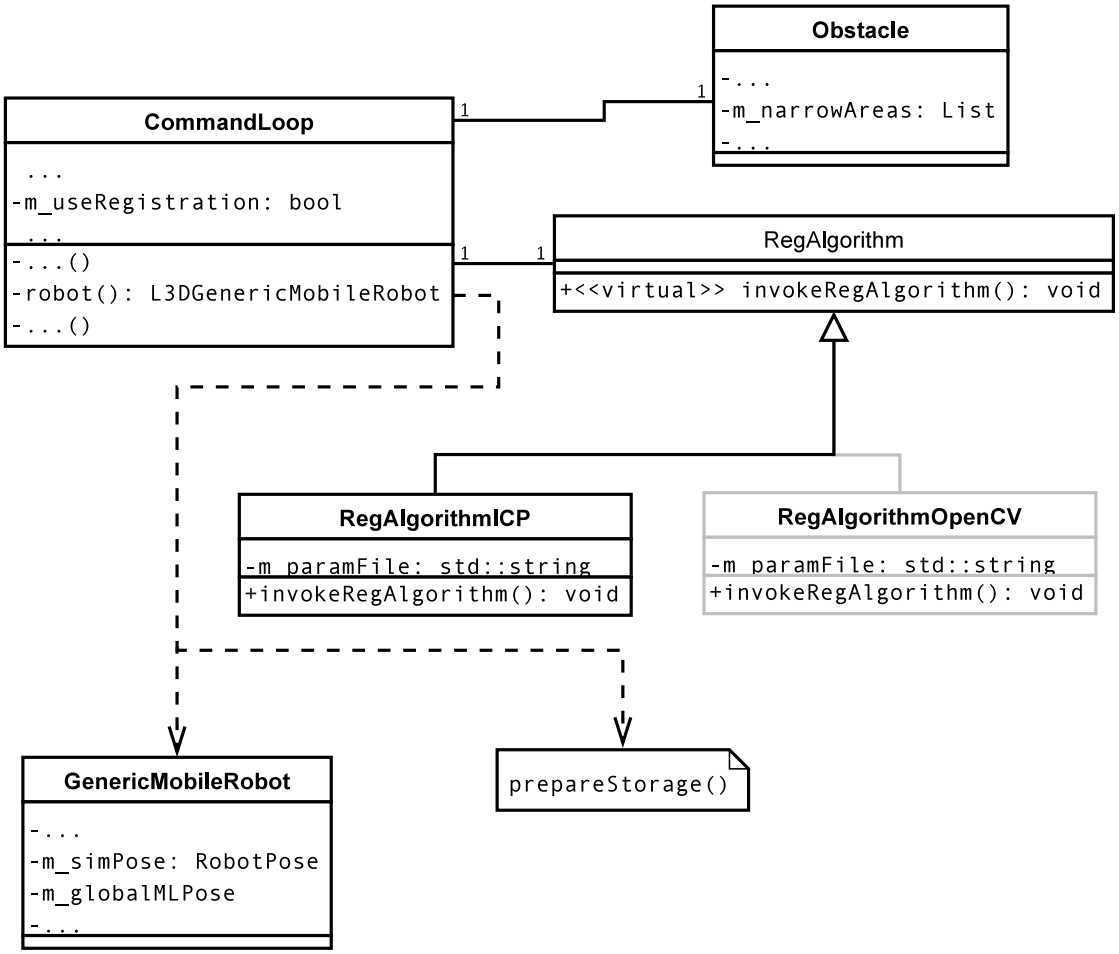

Figure 5.14: The structure of the most important involved components. The registration algorithms are implemented using the design pattern Strategy in which every algorithm has an own class. The diagram is simplified: The parameters of the functions are left out. CommandLoop, GenericMobileRobot and Obstacle have much more functions and responsibilities. `RegAlgorithmOpenCV` could be another possible implementation of `RegAlgorithm` and could be run and maintained independently of `RegAlgorithmICP`.

into account, if the euclidean distance to it is smaller than this radius. The result is a much shorter runtime without drawbacks, if the radius is carefully selected.

However, the template data normally only covers the narrow area and a bit of its surroundings. If the robot is not directly in the narrow passage its sensors often return depth images from areas not covered by the template data. To find out whether this has an impact on the runtime of the ICP, runtime tests were created and conducted.

Table 5.3: Example of the evaluation of the ICP with the three front cameras and all cameras. The second row shows the robot position after applying the algorithm

	Three front cameras	All cameras
Runtime in seconds	1.04	2.78
robot position	3918.71, 9110.31	3918.71, 9110.31

Table 5.3 depicts that the estimated robot position does not change at all on the one hand. On the other hand the runtime decreases significantly when only the three front sensors are used. This test is also done in other situations where only the three front sensors are needed. The relative runtime reduction of around 70 % is nearly the same at all the tests. But this case only observes the application of the ICP alone. An implementation of this optimization was included into the `prepareStorage()` function. The runtime of the improved function `prepareStorage()` together with the ICP is only slightly shorter than the runtime of both components without the optimization. Furthermore, the optimization was delivering wrong depth images when it was applied into the MRE, although unit tests showed a different behavior of it when it was isolated. This difference in behavior could not be examined, thus a solution was not found. So the optimization was removed again and the examination of it remains a task for a future analysis.

5.7 Changes of the navigation system

The section 4.4 stated a way to manipulate the robot's driving so that it does not drive and relocalize unnecessarily. The depicted way is only applied if the registration process took place because the MCL alone is too inaccurate. In Figure 5.15 the branch "Way to next way point clear?" can be seen down in the center. This was the only modification, after the registration. It bypasses the following branch "Close enough to current way point?" if the robot can drive directly to the next way point without collisions. Otherwise movement and relocalization cycles are possible if the robot is not close enough to the current way point. After setting the next way point the robot tries to reach it in the next step automatically.

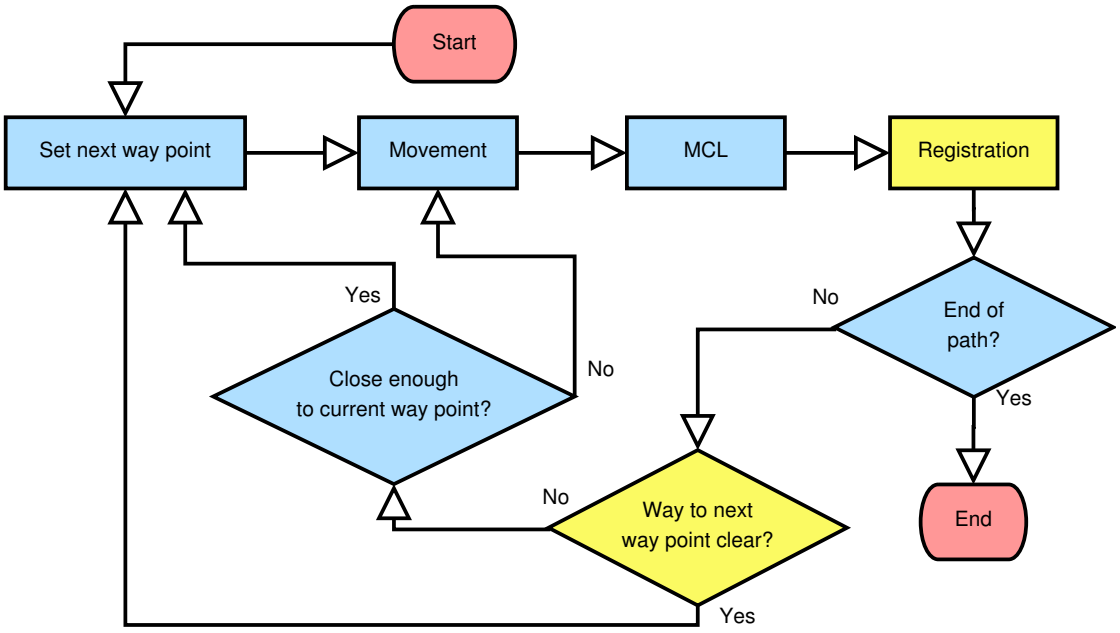

Figure 5.15: A simplified diagram of the navigation system with the optimization, assuming that the registration is used in every step.

5.8 Evaluation of the use of registration algorithms

The evaluation in this thesis focuses on the ICP. The other ones failed at registering the data correctly or the input data for them could not be calculated correctly. One problem occurred when running very many test runs (100 and more). Sometimes the robot strays from the normal path, delivering completely false results for the MLPose from then on until it by chance is on the path again. Since it appears both in the runs with and without registration, the origin of this behavior probably does not lie in the implemented component. The corresponding results to such situations were sorted out when a data set was used for an evaluation.

Mainly, two ways will be analyzed, depicted in Figure 5.16. For each way the robot has to travel through the narrow area. Each way will be evaluated with and without registration.

5.8.1 Runtime

The runtime was one major issue for this thesis. The idea was to make a very accurate position guess to decide whether the path to the next way point is free and then skip the current way point. So the repeated relocalization steps in front of the obstacle should be omitted. Even if the registration algorithm costs some runtime, the more accurate position estimation should save more runtime in a later stage.

What stands out in Figure 5.17, is the high variance of the time measurements. In

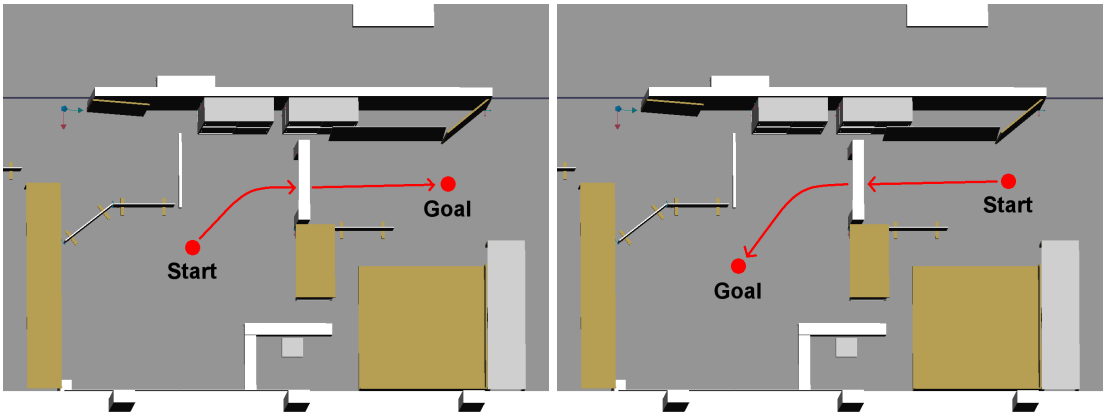

Figure 5.16: Both analyzed ways, *way 1* (left) and *way 2* (right).

Table 5.4 the corresponding means show that the robot on average needs around 3 seconds longer on *way 1* and around 7 seconds longer on *way 2*. But with this high variances the results are not convincing enough. Nevertheless it has to be taken into consideration that the ICP needs additional runtime of around 1 to 2 seconds in every step near the narrow area. At this point it can be stated that the application brings a little advantage on *way 1*, but it does not outweigh the additional runtime the ICP needs. On *way 2* the application does not bring any advantage of run time.

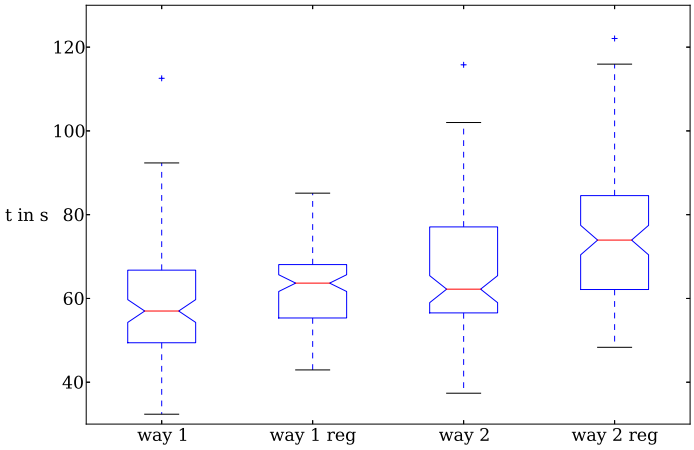

Figure 5.17: Time measurements of *way 1* and *way 2*. Each way is driven 100 times.

Table 5.4: The average times

	without registration	with registration
way 1	59.65	62.93
way 2	67.33	74.37

5.8.2 Achieved accuracy

Generally, the accuracy of the pose estimation on the X-axis was enhanced by using the ICP.

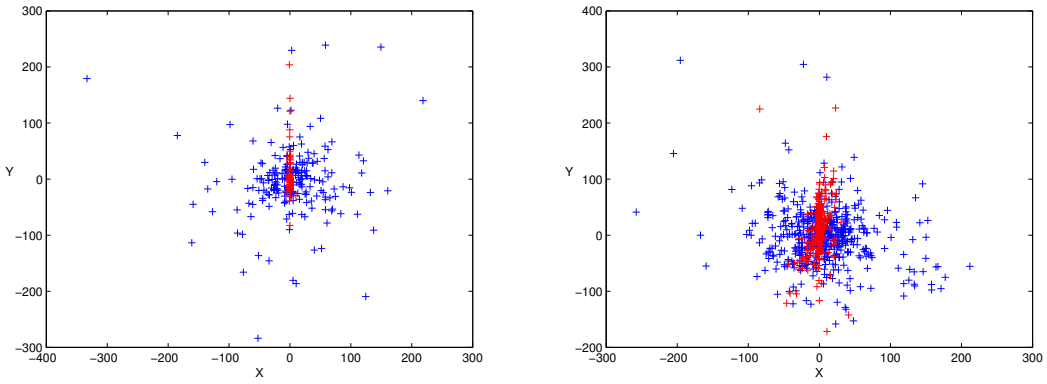

Figure 5.18: Deviance of X- and Y-axis of 21 runs on *way 1* (upper Figure) and 101 runs on *way 2* (lower Figure) in mm. The deviance with registration are red, the deviance without registration, only with MCL, are blue. Note: the number of measurements with MCL is more than the number of measurements with registration. The deviance is recorded in every movement-update cycle.

Table 5.5: The standard deviations of the angle measurements

	without registration	with registration
way 1	3.4833	1.4741
way 2	3.4087	1.3430

Figure 5.18 shows that the Y-component of the pose is not matched very good at all, while the accuracy of the X-axis is mostly improved. The matching works very good for holding enough space to the obstacle left and right. But in the driving direction it becomes inaccurate. The reason is that most of the surface space in the narrow environment is perfectly shaped to match points on the X-axis. In Figure 5.20 it can be seen that the majority of the surfaces are good for matching points on the X-axis, while only

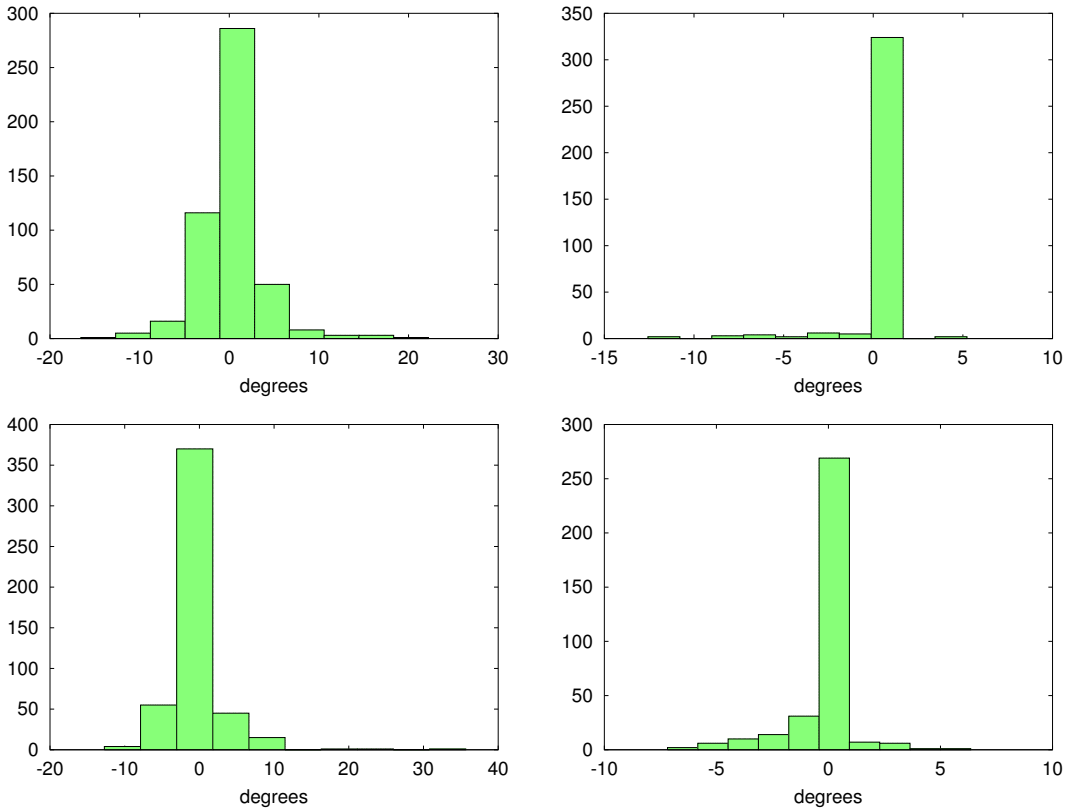

Figure 5.19: The variance of the angles without (left) and with (right) registration on *way 1* (top) and *way 2* (bottom). *Way 1* was driven 100 times and *way 2* 101 times.

few surfaces exist where points could be matched on the Y-axis. The environment of the robot is well structured in this case compared to a straight corridor. Nevertheless the sensor coverage is often not dense enough and especially the characteristic obstacle is missed many times. It is important to notice that it is *not an axis specific* problem of the ICP but a problem of the *environment*. If the whole map would have been turned 90 degrees around the Z-axis, the problematic axis now would be the X-axis and not the Y-axis. If the map would have been turned around 45 degrees, both components, X and Y, would have roughly the same deviance, while the deviance would still tend only in one direction: the driving direction through the obstacle and its opposite direction. As soon as the environment has a different structure, the result would be different too. Considering the results above, an unstructured wall with a simple door frame will probably cause more problems on matching the critical deviance to the right and the left of the robot.

Another issue is that the X-component is matching worse on *way 2* than on *way 1*. The problem might be that the robot has only few space around its start position. it can not make an accurate first position estimation with active localization then. The following position estimations might not be good either until a bad position estimation determines the first initial pose of the sensor data for the ICP. The exact reason could not be found before this thesis was finished.

The angle deviation was getting lower on both ways (see Figure 5.19). That is positive since a position error caused by angle deviation gets higher, the longer the distance the robot travels is. Only on *way 1* the vast majority of the deviance seem unrealistically low, while there are some significant freak values on the other side. This also explains the relatively high standard deviation for *way 1* with registration in Table 5.5 Whether these values are right should be analyzed at least in a later evaluation with the real robot.

In the case of *way 1* the problem of the translational deviance might not be as bad as in other situations. The MLPose is matched relatively good in the directions that are critical for the side clearance of the robot. The angles show good corrections, too. What is matched worse is only the direction the robot is driving to. That direction is normally not so critical because the path planning has already assured that it is clear (in a static environment). Otherwise the path would have been planned another way. So the manipulated MLPose does not have to be as accurate in the driving direction as in the different ones and as the angles.

Howsoever, the achieved accuracy is generally not reliable and high enough. So the registration should not stay integrated in the MRE in this condition, at least not as a default procedure. Is definitely has to be improved and further tested until it works more accurate and more reliable.

This result contradicts the more promising results in subsection 5.4.1. There the ICP was mostly matching the template and sensor data well, even on the Y-axis. The reason is that only few perspectives were chosen in subsection 5.4.1. In all perspectives the sensor data always covered at least a part of the characteristic shape of the simulated door frame. In the MRE the robot was mostly closer to the narrow environment so that the sensors often missed the simulated door frame completely. A better study of the robot's behavior in MRE before determining of the poses for the rough evaluation might have cleared this difficulties earlier.

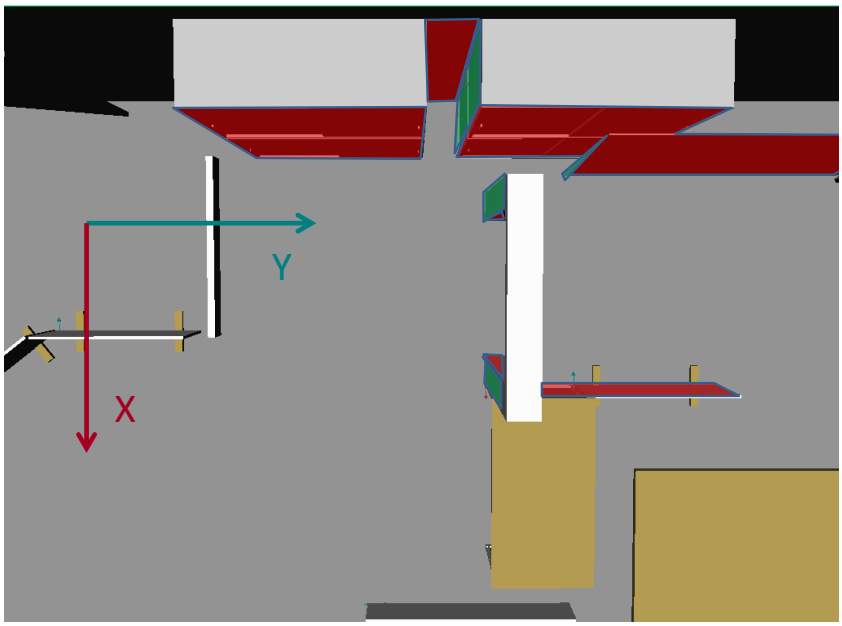

Figure 5.20: Surfaces where points can match the X-component are marked red, the surfaces where points can match their Y-component are marked green.

6 Conclusion

The goal was the analysis of the impact of registration methods on the navigation system especially when the robot drives through narrow environments. An implementation of the ICP for this purpose was made, what is the basis for an evaluation of runtime and pose deviance at the end.

What is already obvious in section 5.3 is the relatively low coverage of the depth sensors in the near environment of the robot. But naturally the robot needs to perceive near obstacles in a narrow environment. And high coverage of them is often crucial for the scan matching process. For example the robot often misses the characteristic shape of the simulated door frame that usually provides good results for the ICP - if it is visible. As a consequence it has to be evaluated whether the registration can be successfully used for local navigation in narrow areas with the current state of the sensor coverage.

The MCR and the image processing method show quite bad results at the early stage of this work. This does not mean that they are generally bad for scan matching in mobile robotics. But in this case they suffer especially from the low sensor coverage in the near environment of the robot. If the coverage was better, they probably would perform better. Furthermore, a combination of the MCL with the image processing methods would make sense. The geometric feature detection has problems with jumps (from one object to another) in the depth image, while the Canny Edge detector detects exactly these jumps. Both methods show opposing disadvantages that could be cleared if they were combined.

Another problem that turned out during the evaluation of this thesis is the high variance of the robot's driven path. The behavior of the robot on a certain path is very different from run to run. Sometimes it needs more than twice as much movement-update cycles. This is visible in the time measurements in Figure 5.17. The times vary very much too. The high variance makes all results less worthy, so final statements cannot be made.

The reader might have noted that the *Strategy* design pattern now only provides one concrete algorithm subclass for the ICP. But that does not mean the *Strategy* design pattern is useless here. On the one hand it was already useful for the unit testing. On the other hand the algorithm can be changed easily in the future, without big influence of other classes. It is the compromise that deals best with the requirements. If they change in the future, the situation must be analyzed and it should be decided whether the pattern will still be necessary.

The optimization of the input data for the ICP could not be finished for this thesis. If it worked, a few seconds might be saved for every run of the registration. But it will probably not be enough to give the registration algorithms a significant advantage in runtime. The problem with the inaccuracy will not be affected at all.

Furthermore, the collision avoidance over the whole driven path could not be analyzed. This is a major problem since collision avoidance is essential and has the highest priority in most scenarios in mobile robotics. So the next step should be to check collision avoidance when the implementation of a collision check is available.

However, the results of the analysis in this thesis are not sufficient to come to a full conclusion, whether the registration algorithms should be implemented or not. The data basis with the high variance becomes problematic. A further analysis should be made to verify the results of this thesis with a bigger or more stable data basis.

Bibliography

[1] Open inventor mentor on the visualization sciences group's website. http://oivdoc92.vsg3d.com/content/121-open-inventor-file-format, 2013.

[2] Opencv documentation. http://opencv.org/, 2013.

[3] Maxim A. Batalin, Gaurav S. Sikhatme, and Myron Hattig. Mobile robot navigation using a sensor network. 2004.

[4] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *IEEE Transactions PAMI*, 14(2):239–256, February 1992.

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1994.

[6] Christian Rink, Zoltan-Csaba Marton, Daniel Seth, Tim Bodenmüller, and Michael Suppa. Feature based particle filter registration of 3d surface models and its application in robotics. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS (accepted)*, 2013.

[7] Jörg Röwekämper, Christoph Sprunk, Wolfram Burgard, Gian Diego Tipaldi, Cyrill Stachniss, and Patrick Pfaff. On the position accuracy of mobile robot localization based on particle filters combined with scan matching. volume 4. IEEE, 1997.

[8] Gennadiy Rozental. Boost.test documentation. http://www.boost.org/doc/libs/1_53_0/libs/test/doc/html/index.html, 2007.

[9] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[10] Roland Stenzel. *Steuerungsarchitekturen für autonome mobile Roboter*. PhD thesis, RWTH Aachen, 2002.

[11] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, MA, 2005.