Master Thesis

Active SLAM in Crowded Environments

Author(s):
Mammolo, Dario

Publication Date:
2019-03

Permanent Link:

https://doi.org/10.3929/ethz-b-000341706 →

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master Thesis

Active SLAM in Crowded Environments

Spring Term 2019

Supervised by:
Dr. Jen Jen Chung
Daniel Dugas
Dr. Mark Pfeiffer

Author:
Dario Mammolo

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

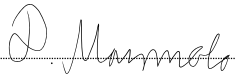
Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Mammolo	Dario

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 25.03.2019	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Abstract

The safe navigation and execution of tasks, such as assisting humans, in a crowded environment is still a difficult task for mobile robots and is still not fully solved. As a first step towards solving this issue, we present in this thesis an active SLAM algorithm for static environments, which we then extend to crowded environments. The goal of active SLAM is to give a robot the ability to explore an unknown environment, even if it is crowded, in an optimal way. We use a utility function based on Rényi's general theory of entropy, which automates the trade off between exploration and exploitation without the necessity to manually tune parameters [1]. We show that the free scalar parameter in Rényi's entropy was not related correctly to the pose uncertainty. To improve this, we present a relation, which scales to the map resolution. We further show that moving people can be detected in 2D range data by using an adaptive breakpoint detector for clustering possible candidates. We then track the clustered objects using individual Kalman Filters. The tracks are then distinguished using counter, size and velocity checks. Further, we use an adaptive people ellipse, which adapts its form depending on the current velocity and moving direction, to merge leg clusters, obtained from a 2D LIDAR, for better tracking of people. Finally, we show results both in a simulation and real environment for static and crowded cases. We can show that our approach achieves better results than a simple shortest frontier strategy and that the detection and tracking of moving people in a crowded environment clearly improves the performance of our active SLAM framework.

Chapter 1

Introduction

The goal in mobile robotics is to have a system, which is capable to move autonomously in real environments, especially if these environments are unknown. One of the essential parts to achieve this goal is having a working simultaneous localisation and mapping (SLAM) system. Since the last twenty to thirty years, SLAM has been a focus research area in mobile robotics. There exist a big variety of SLAM systems with pros and cons. Current state-of-the-art algorithms are able to run online on mobile robots. Passive SLAM systems do not navigate a robot to explore an unknown environment. Active SLAM however, tries to enable the optimal exploration of an unknown environment. To perform active SLAM, an exploration strategy is necessary to generate possible future actions the robot can choose. These actions have to be evaluated for their utility based on the optimal decrease in map and pose uncertainty, such that the resulting map is of high quality.

1.1 CROWDBOT Project

This thesis is based on the CROWDBOT project[1], which has the goal of enabling mobile robots to navigate autonomously and assist humans in crowded areas. One part of this project is to achieve this goal for the Pepper robot from SoftBank Robotics[2].

1.2 Goal of Thesis

SLAM algorithms are susceptible to dynamic objects in the environment as localisation and mapping depend on the different viewpoints of a static environment. As crowded areas are highly dynamic, it is essential to have a SLAM algorithm, which is working in dynamic environments. Having a SLAM system working in different types of dynamic environments is still an unsolved problem in research. The goal of this thesis is to make a big step into the autonomy of a mobile robot by developing an active SLAM algorithm for crowded environments. In this thesis we develop an active SLAM algorithm for static environments, which we then extend for crowded environments.

[1] `https://cordis.europa.eu/project/rcn/212487/factsheet/en`, (Accessed on 19.12.2018)
[2] `https://www.softbankrobotics.com/emea/en/pepper`, (Accessed on 19.03.2019)

1.3 Structure of Report

Chapter 2 relates works done in active SLAM and dynamic environments. We describe similar works that have been done, but also show interesting approaches to solve certain issues, which are worth mentioning. In Chapter 3 we first describe the active SLAM framework built and used in this thesis. We then show how we handled crowded environments in simulation and reality. The last part of this chapter gives further information on the simulation environment used. Chapter 4 presents results gathered in simulation and reality. Chapter 5 concludes this thesis and gives a small outlook on what further improvements could be done.

Chapter 2

Related Work

Most of the research in the area of SLAM has been done for static environments where the robot is steered by human actions. Because of this it is hard to find a work that tries to solve the active SLAM problem for dynamic environments. There are a few active SLAM algorithms for static environments, which are summarised in Section 2.1. There are also some approaches on solving the SLAM problem for dynamic environments, which are discussed in Section 2.2.
One approach to solve the active SLAM problem for dynamic environments has been done in [2]. They developed their own active SLAM algorithm for dealing with dynamic problems. The main idea was to use a monocular camera pointing towards the ceiling. The camera is used to detect ceiling features and to estimate the pose of the robot using an iterative closest point (ICP) algorithm with the detected features. The usage of a camera, which is pointing to the ceiling, prevents having moving objects, like humans, disturbing robot sensors. But in our opinion this approach has some general disadvantages, like the need for additional sensors for navigation, a higher computation usage and the need for structures on the ceiling. In this thesis we show that active SLAM is possible in crowded environments by only using 2D laser scans.

2.1 Active SLAM in Static Environments

Active SLAM for static environments has been studied more than for dynamic environments. In most approaches the active part can be used for different SLAM frameworks. In general the goal is to decrease the pose uncertainty of the robot and the map uncertainties simultaneously and in an optimal way. A good survey on the state-of-the-art in active SLAM and SLAM in general can be found in [3]. They describe active SLAM as a decision-making problem to exploration-exploitation decisions. To solve this problem there are several frameworks, which can be used. One of these is the Theory of Optimal Experimental Design (TOED) [4]. An example applied to active SLAM can be found in [5]. It allows selecting future robot actions based on the predicted pose and map uncertainties. Another framework is the application of information theory [6] to the decision problem. The decision making here is in general guided by the notion of information gain. An example, where it was applied to, is [1] where a utility function based on Rényi's general theory of entropy [7] is used. One main advantage of this utility function is that it automatically trades off between exploration and exploitation. Because of this advantage, it has been also used in this thesis. Another example is the work of Stachniss et al. [8] where a Rao-Blackwellized particle filter has been used for solving the SLAM problem using occupancy grid maps as map representation. Map and pose uncer-

tainties are described using the Shannon entropy. Carrillo et al. [1] showed in their work that utility functions that only use Shannon's entropy, neglect the fact that the entropies for map and pose uncertainties are two completely different quantities. There also exist control theoretic approaches to the active SLAM problem, which use Model Predictive Control [9]. One of the first approaches of active SLAM to Pose SLAM was done in [10]. The advantage in the computation of the utility is that with Pose SLAM only one map exists instead of multiple maps, as generated for example using a particle filter SLAM system. A further development of the work of Valencia et al. [10] is described in [11]. A growing RRT* tree is used as a set of candidate paths, which will be used for the computation of their utility functions. A further work, which only used the Shannon entropy in the utility computation, is [12]. This approach can be used for any graph-based SLAM algorithm. The work proposes the use of a Topological Feature Graph (TFG) as map representation, where landmarks are connected together if they belong to the same object.

2.2 SLAM in Dynamic Environments

Active SLAM for crowded environments has until now not been a focus in research, therefore we will focus in this section on applications for SLAM in dynamic environments. Usually, the developed approaches can be easily included or adapted for active SLAM. Dynamic objects are mostly handled in two ways. Either one tries to detect them and treats them as outliers or tracks each dynamic object separately, using for example multi-target tracking. An overview on multi-target tracking can be found in the work of Luo et al. [13]. One of the first approaches using SLAM and detection and tracking of moving objects (DATMO) in dynamic outdoor environments is described in [14] for the application using vehicles. The detection of moving vehicles is based on the inconsistencies between observed free space and occupied space in the local grid map, which has been built so far. Detected moving objects are then tracked by a multiple hypothesis tracker (MHT) coupled with an adaptive interacting multiple model filter. Bahraini et al. [15] propose to use a multilevel-RANSAC algorithm for classifying detected objects into stationary and moving objects. The detected objects are extracted from 2D laser scans using an adaptive breakpoint detector [16]. The objects classified as moving are then integrated into an EKF SLAM algorithm. Another approach called SLAMIDE [17] includes the detected dynamic objects into the least-square formulation of SLAM. To include dynamic objects into SLAM, they combine sliding window optimisation and least-squares together with generalised expectation maximisation to do reversible model selection and data association. Including dynamic objects into SLAM helps with localisation in highly dynamic environments. Another approach focuses more on the long-term mapping for low dynamic environments [18]. It is called the dynamic pose graph SLAM and uses iSAM [19] as the underlying SLAM state estimation engine. Laser scans for the same area of an environment at different times are compared to perform change detection. If a change has occurred, old poses and scans that no longer match the current state are removed from the pose graph. An important part for SLAM in dynamic environments can be DATMO. Mertz et al. [20] describe a DATMO system for 2D and 3D laser scanners where the core algorithm of DATMO is described in [21] and [22]. Objects in the scan data are clustered into segments using a basic region growing algorithm. They then extract features (lines and corners) from the segments, such that the tracking using Kalman Filters is simplified. Yin et al. [23] include the detection of dynamic objects into the scan matching process for graph-based SLAM. The scan matching consists of three phases. The first phase executes a Hough Transform based segmentation to extract and group line features, the second phase is an occupancy-analysis based moving

object detection, which detects and discards moving objects and the third phase does a linear regression based feature matching to estimate the roto-translation parameters. This approach is based on line features in indoor environments and is for example not suitable for moving people. A more exotic approach to detecting moving objects is based on deep learning methods [24]. In fact, this approach is not restricted to moving objects as the geometrical structure of laser scan points is learned and can be used to detect objects independent of the moving state. A Convolutional Neural Network (CNN) is trained to detect wheelchairs and walkers. An extension of this work includes the detection of person in the 2D range data [25].

Chapter 3

Methodology

In this chapter the active SLAM approach for crowded environments is presented and explained in more detail. Section 3.1 describes the used Back-End and Front-End of the SLAM system. The following Sections 3.2 and 3.3 describe the exploration approach, like the chosen utility function, and the detection and tracking of moving people. In Figure 3.1 an overview of the developed system and their components is shown. Finally, Section 3.4 describes the simulation environment used in this thesis.

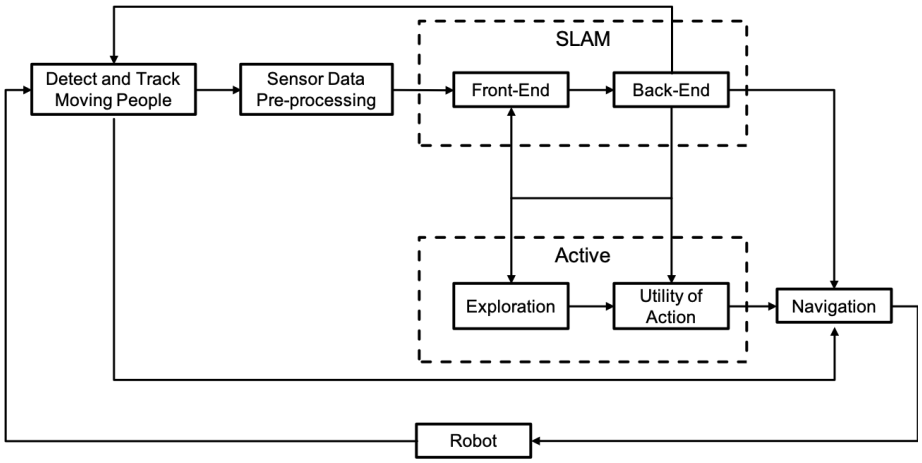


Figure 3.1: This is an overview of the whole active SLAM framework developed in this thesis. The boxes describe the main modules, which are necessary for active SLAM in crowded environments. Arrows show the flow of data between the modules.

3.1 SLAM Framework

The focus on the development of the SLAM framework was to build a state-of-the-art system. Because of the complexity of an active SLAM framework and the restricted time period of the thesis, we wanted to use components, which can be found open source, for SLAM and Navigation. For programming reasons the usage of a complete SLAM system, such as gmapping[1] [26][27] or ORB-SLAM [28][29], was not desired, as active SLAM needs access to internal information, which is

[1] `http://wiki.ros.org/gmapping`, (Accessed on 20.12.2018)

not readily exposed by the out-of-the-box frameworks. In addition, developing a custom framework gives extra flexibility in customising its characteristics to the problem at hand. Further, in prospect of the development of a solution for crowded environments, the system should be kept easy, such as only using 2D laser scans.

3.1.1 Front-End

As Front-End we use a laser scan matcher for the generation of laser odometry information. The scan matcher is an iterative closest point (ICP) variant using a point-to-line metric [30]. The implementation in this thesis is based on the implementation in the ROS package *laser_scan_matcher*[2], which uses this ICP variant for consecutive laser scans. A keyframe-based approach is used to prevent pose drift. The ICP variant, also called canonical scan matcher (CSM) is a very fast scan matcher. Much of the speed in the CSM comes from a smart correspondence-search procedure. As alternative the libpointmatcher developed from Pomerleau et al. [31] has also been tested as Front-End, but seemed to be slower. Further, an extensive parameter tuning of scan filters would have been necessary to get a good performance. For these reasons, the CSM has been chosen as it is already highly optimised for our application. The CSM has also an accurate closed-form solution implemented for the estimation of ICP's covariance [32].

3.1.2 Back-End

As Back-End an incremental graph-based SLAM algorithm called iSAM2 [33] is used. The graph is based on a factor graph implementation. See Figure 3.2 for an example of a factor graph for pose SLAM. The graph is built using factors, which describe the relation between the nodes. The nodes correspond to the position estimates in the world. Factors can be built from consecutive poses through laser odometry and through loop closings. iSAM2 is based on further improvements of iSAM [19] and Square Root SAM [34]. A nice tutorial on factor graphs and GTSAM[3] can be found in [35]. GTSAM is a smoothing and mapping (SAM) library from Georgia Tech, which has implemented all above-mentioned algorithms. iSAM2 has been chosen, as it is, for the nature of iterative algorithms, very fast and currently one of the state-of-the-art SLAM Back-Ends. iSAM2 allows us to easily access the marginal covariances of the current pose graph, which will be necessary for the computation of the utility function, as presented in Section 3.2.3.
In general the graph is built while the robot is moving. In the beginning, we initialise the graph with a prior factor on the starting pose. In simulation we use simulated localisation information to initialise the SLAM map frame to the same frame as the ground truth map frame, allowing us to easily perform comparisons. In a real environment, where an initial localisation is missing, the map frame is arbitrarily set to having its origin at the robots initial position. After the robot moves for a certain distance, which can be estimated using the laser odometry data, a new node is added to the graph. A new factor is defined between the previous and current node pose. These factors also contain pose uncertainty information, which can be retrieved by our laser scan matcher as mentioned in Section 3.1.1. While adding new nodes, we save the corresponding laser scans for map building and loop closing purposes. After every new added node, the system searches for loop closings. Loop closings are found by iterating through the pose graph and by checking if there exist poses in a range around the current pose, which do not exceed a certain threshold. For each found loop closing, we use the CSM to find the relative pose between the two laser scans of the loop closing poses. As initialisation, the relative pose between

[2] `http://wiki.ros.org/laser_scan_matcher`, (Accessed on 20.12.2018)
[3] `https://borg.cc.gatech.edu`, (Accessed on 20.12.2018)

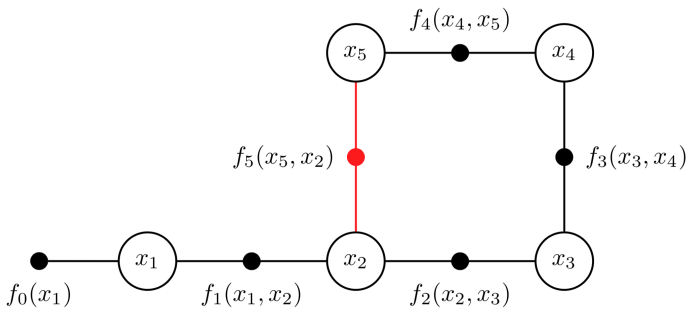


Figure 3.2: An example of a factor graph for pose SLAM, where x_i corresponds to the nodes of the factor graph and f_i to the factors. $f_0(x_1)$ is in this case a prior factor on x_1. $f_5(x_5, x_2)$ is a factor between x_5 and x_2, which describes a loop closing. Figure reference from [35].

them in the current graph is used. The resulting relative pose is used to build a new factor for the factor graph. After the graph has been searched for loop closings, the pose graph is optimised. For computational reasons we check after each scan matching step if enough time is left for further loop closing computations. After a scan match step, we skip a certain number of nodes in the graph, with the idea that we only compute one loop closing per graph intersection area.

3.1.3 Mapping

The map is represented by a 2D occupancy grid map and is computed using the occupancy grid mapping algorithm [36]. In general the update equation, if a map cell is in the perceptual field of our sensor, can be defined in the log odds notation as:

$$l(m_i|z_{1:t}, x_{1:t}) = \underbrace{l(m_i|z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i|z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} + \underbrace{l(m_i)}_{prior}, \tag{3.1}$$

where m_i is the i-th discrete random variable of the occupancy grid map, which describes the occupation of a cell. z and x are time-dependent measurement and state variables, respectively. The prior is the log odds value of the prior probability $P(m_i)$ at map initialisation. In our case we initialise all m_i as $P(m_i) = 0.5$ (unknown), which results in a value of zero for the prior. The recursive term is just the log odds value in the previous time step. The inverse sensor model term handles the update of the log odds value depending on the newest measurement z_t. To get a sensor model we first have to define a plausible sensor model. In Table 3.1 the chosen probabilities for our sensor model are shown. The value of $P(z = 1|m = 1)$ has been chosen like this, as there exists the possibility that a cell could be occupied by a window or a glass door for example, which is hard to be detected by 2D LIDARs. Laser beams can reflect on glass and induce wrong measurements.
In Table 3.2, the inverse sensor model is shown.

	$z = 1$	$z = 0$
$m = 1$	0.8	0.2
$m = 0$	0.01	0.99

Table 3.1: The values describe the sensor model, i.e. the probabilities $P(z|m)$.

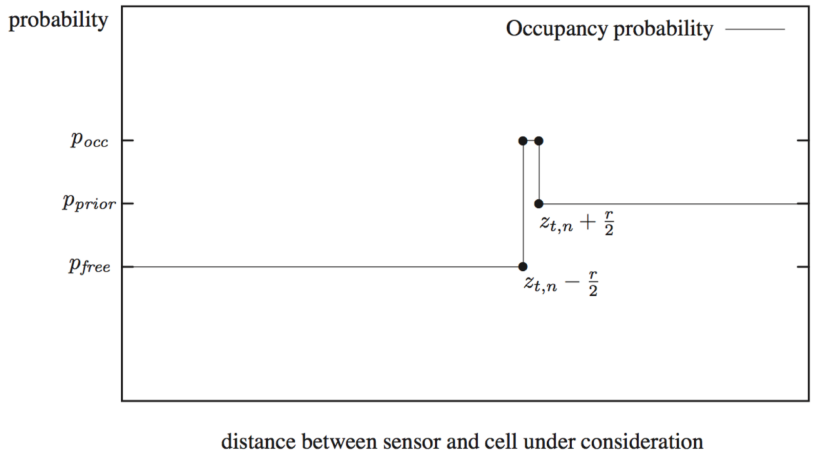


Figure 3.3: This plot shows the inverse sensor model for a laser beam hitting a wall at distance $z_{t,n}$, where r can be interpreted as the cell size of an occupancy grid map.[4]

	$z = 1$	$z = 0$
$m = 1$	0.998	0.168
$m = 0$	0.002	0.832

Table 3.2: The values describe the inverse sensor model, i.e. the probabilities $P(m|z)$.

In Figure 3.3 a diagram of the inverse sensor model is shown. Because laser measurements are not noise free and the fact that this noise can have an impact in which cell the measurement lies, one can define a region with the size r, in our case r corresponds to the map resolution, where the probability is updated as an occupied cell. In other words, this means that if a sensor measurement is not exactly in the centre of a map cell, the next closest cell is also updated as an occupied cell. This results in occupancy grid maps that tend to have thicker walls.
As the pose graph estimates can change after each optimisation, i.e. after adding a new node, the whole map needs to be recomputed. This is intractable for growing pose graphs. For this reason, the map is only updated incrementally after a new node is added. As this SLAM system is intended to be used in an active SLAM approach, recomputing the whole map can be done after a goal position of the robot has been reached and before the possible future goals and the corresponding utilities of the action paths are computed. If it is necessary to only use the SLAM system, the re-computation can be done after a fixed number of nodes. As the focus is to develop an active SLAM solution, no additional effort to the re-computation has been dedicated.

3.2 Autonomous Exploration

Autonomous exploration can be described as an exploration-exploitation problem. Ideally exploration and exploitation is done simultaneously and optimally in the

[4]Figure source: `http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam11-gridmaps.pdf`, (Accessed on 05.01.2019)

sense of exploring the unknown world as fast as possible while reducing uncertainties of robot pose estimates and generated maps optimally.

3.2.1 Frontier Exploration

Frontier [37] detection can be used to generate possible future goals for a mobile robot. Frontiers are generally regions on the border of explored and unexplored areas. In the case of occupancy grid maps, these frontiers are the regions between free and unknown cells. The implementation in this thesis is based on the ROS implementation *frontier_exploration*[5]. In general the frontier cells are detected by searching for free cells, which are only adjacent to unknown cells, and then clustered together. A cluster is only classified as frontier, if the size is greater than a certain threshold. This threshold is a design parameter and is also dependent on the map resolution and the environment in which the robot operates. In our case the threshold is set to 8 cells, which corresponds to a range of up to 0.4 m when using a map resolution of 0.05 m.
Frontier exploration in principle does not take into consideration the possibility of revisiting already observed regions to decrease robot pose uncertainties. During exploration, the paths from the current robot pose to the frontiers can pass already observed regions (but do not necessary have to) and as described in Section 3.2.3 possible future loop closings are considered in the computation of the utilities of a path. Of course this does not ensure an optimal decision making between exploration and exploitation, as there might be better paths, which are not considered by our strategy. For the sake of time in this thesis and for computational reasons, additional path generations for better active SLAM performance are left over for future study.

3.2.2 Navigation

For autonomous exploration it is essential to have a navigation system, which is capable to drive a robot from its current position to a goal position. We use the ROS package *move_base*[6] as our navigation system, which uses a global and local planner. The global planner is called *navfn*[7] and uses Dijkstra's algorithm to plan paths. The local planner is called *teb_local_planner*[8] and implements the method Timed Elastic Band [38][39], which optimises the execution time, separation from obstacles and compliance with kinodynamic constraints at runtime. A great benefit of the Timed Elastic Band planner is that dynamic obstacles can be included into the planning. Both planners need a 2D cost map[9] for planning. For the computation of the utility function in the next section, the paths from the current robot pose to the frontiers have to be computed. We use the global planner for the computation of these paths.

3.2.3 Utility of Paths

The utility function is based on a Shannon and Rényi entropy and has been developed by Carrillo et al. [1]. In this work a similar active SLAM system as in their work has been used. Compared to other active SLAM approaches, most of them only use the Shannon entropy. But this is in general not a good choice, as the map and pose entropies are fundamentally different quantities. The entropy of the pose

[5] `http://wiki.ros.org/frontier_exploration`, (Accessed on 20.12.2018)
[6] `http://wiki.ros.org/move_base`, (Accessed on 21.12.2018)
[7] `http://wiki.ros.org/navfn`, (Accessed on 21.12.2018)
[8] `http://wiki.ros.org/teb_local_planner`, (Accessed on 21.12.2018)
[9] `http://wiki.ros.org/costmap_2d`, (Accessed on 21.12.2018)

of the robot, which is a continuous random variable, can take any real value while the entropy of a discrete random variable, such as the grid cells in our map, takes only strictly non-negative values. As shown by Carrillo et al. [1], these two entropies can have vastly different numerical values in an autonomous exploration task.
Rényi's entropy can also be seen as the generalised Shannon's definition of entropy and is defined by:

$$H_\alpha[P(\boldsymbol{x})] = \frac{1}{1-\alpha}\log_2\left(\sum_{i=1}^{n} p_i^\alpha\right). \tag{3.2}$$

Shannon's entropy is a special case of Rényi's entropy in the limit of $\alpha \to 1$. Jumarie [40] presents a definition of mutual information as the difference between the Shannon entropy of the probability distribution $P(\boldsymbol{x})$ and Rényi's entropy of the same distribution with $\alpha = c$:

$$I_c[P(\boldsymbol{x})] = \underbrace{H_{\alpha=1}[P(\boldsymbol{x})]}_{\text{Shannon's entropy}} - \underbrace{H_{\alpha=c}[P(\boldsymbol{x})]}_{\text{Rényi's entropy}}, \tag{3.3}$$

where I_c is called the mutual information. The parameter α can be seen as a gain coefficient, which measures the efficiency of an observer who is considering the distribution $P(\boldsymbol{x})$. Based on the mutual information Carrillo et al. [1] defined a new utility function as:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \quad I_{c(\boldsymbol{a})}[P(\boldsymbol{m}|\boldsymbol{x},\boldsymbol{d})], \tag{3.4}$$

where $\boldsymbol{a}$ are the possible future actions and $P(\boldsymbol{m}|\boldsymbol{x},\boldsymbol{d})$ is the current distribution over possible maps. The value of $\alpha = c(\boldsymbol{a})$ depends on the possible future actions $\boldsymbol{a}$. $\boldsymbol{d}$ is the history of data, i.e. control inputs and sensor measurements. Note that computing Equation 3.4 does not require updating the map using possible future measurement, which would require propagating sensor and localisation uncertainties forward. Additionally the utility function is computed only over the regions of the map that will be visited during the action $\boldsymbol{a}$ as was similarly proposed by Stachniss et al. [8]. This leads to the formulation:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \sum_{m\in\boldsymbol{m}(\boldsymbol{a})} \underbrace{H[P(m|\boldsymbol{x},\boldsymbol{d})]}_{\text{Shannon's entropy}} - \underbrace{H_{\alpha(\boldsymbol{a})}[P(m|\boldsymbol{x},\boldsymbol{d})]}_{\text{Rényi's entropy}}, \tag{3.5}$$

where $\boldsymbol{m}(\boldsymbol{a})$ is the subset of the current map that may be visible to the robot when following the action plan. $\boldsymbol{m}(\boldsymbol{a})$ can be computed with standard ray tracing techniques, in this thesis Bresenham's line algorithm [41] has been used[10]. For the computation of the Shannon entropy in Equation 3.5 the same equation as for the Shannon entropy of the map distribution as given by Stachniss et al. [8] has been used:

$$H[P(m|\boldsymbol{x},\boldsymbol{d})] = -(P(m)log_2(P(m)) + (1-P(m))log_2(1-P(m))). \tag{3.6}$$

In general the utility function (see Equation 3.5) will choose actions, which will most improve the map estimate of the partially explored areas of the map, i.e. where $P(m) \neq \{0, 0.5, 1\}$. These situations arise when the robot has poor localisation during its pass through an area. For further details on properties of the utility function refer to [1].
The parameter $\alpha(\boldsymbol{a})$ is then related to the predicted uncertainty in the pose of the robot after taking action $\boldsymbol{a}$, such that it will decrease the information gain

[10] Implementation is based on algorithm found on `https://csustan.csustan.edu/~tom/Lecture-Notes/Graphics/Bresenham-Line/Bresenham-Line.pdf`, (Accessed on 21.12.2018)

when the pose uncertainty is high. When the robot has minimal uncertainty on the pose estimate then the information gain should be maximal. Similarly, when the uncertainty is high the information gain should be zero, which causes the robot to do exploitation in order to decrease its pose uncertainty by choosing the appropriate path. In the case of our utility function we want $\alpha \to 1$ when the uncertainty becomes infinite, since the two entropies cancel out. We want $\alpha \to \infty$ when the pose uncertainty approaches zero, since this minimises Rényi's entropy. A simple candidate for α is:

$$\alpha = 1 + \frac{1}{\sigma}, \tag{3.7}$$

where σ is a scalar representation of the predicted pose uncertainty. Carrillo et al. [1] propose to compute the uncertainty scalar σ by optimality criteria from the Theory of Optimal Experimental Design (TOED) [4]. They propose to use one of the three most commonly used criteria; A-optimality, D-optimality or E-optimality. The criteria can be applied to either the full covariance matrix or to the marginal covariance matrices from each node in the action pose graph. The action pose graph is explained further bellow. In this thesis the D-optimality has been used as it minimises the volume of the covariance matrix ellipsoid and is defined as:

$$\sigma = \det(\boldsymbol{\Sigma})^{1/n} = \exp\left(\frac{1}{n}\sum_{k=1}^{n}\log(\lambda_k)\right), \tag{3.8}$$

where n is the dimension of the covariance matrix and λ_k is the k-th eigenvalue of the covariance matrix.

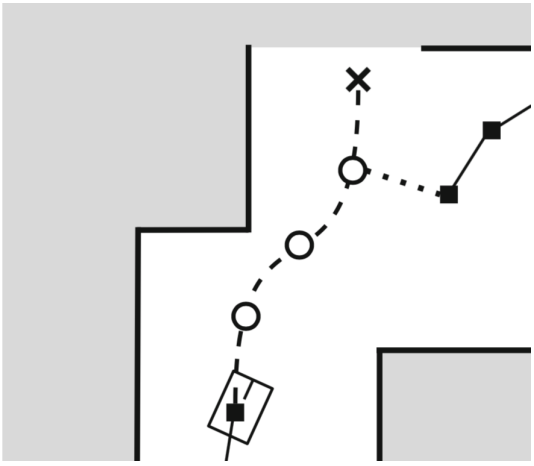
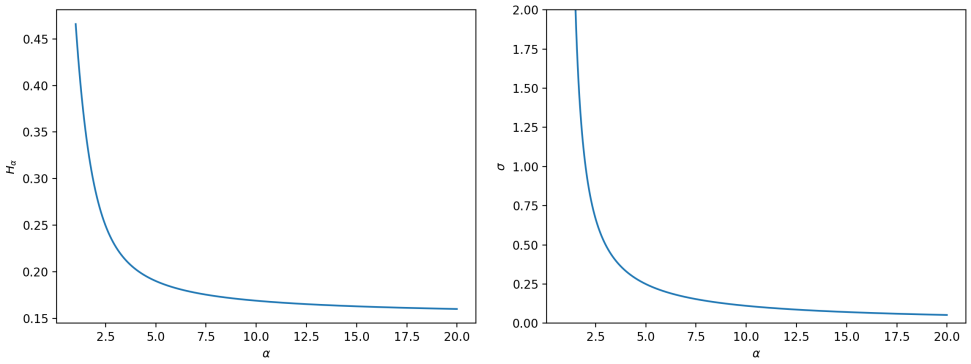


Figure 3.4: The action pose graph is a miniature graph, which represents the action plan. The black square boxes are nodes from the previously built factor graph during exploration. The cross describes the goal of the current action. The circle corresponds to future nodes added to the action pose graph along the action plan.[11]

For the computation of the path utilities, we need to define the uncertainty prediction of the future paths. As proposed by Carrillo et al. [1] we use a miniature pose graph representing the action plan, which is also called the action pose graph. We initialise the graph with an initial factor at the robot's current estimated location, with the covariance matrix taken from the most recent node in the graph. The marginal covariances can be extracted using GTSAM, which uses the method presented by Kaess and Dellaert [42]. To build our action pose graph, we interpolate

[11] Figure source: Carrillo et al. [1]



(a) Rényi's Entropy (b) α Function

Figure 3.5: (a) shows the plot of Rényi's entropy and (b) shows the plot of the α function.

the action plan, which we get from our global planner, with some fixed step size and add pose nodes along the length of the path. As odometry constraints between the nodes, we use fixed covariance values such that longer actions will lead to larger increase in uncertainty. In this thesis we chose average values, which result from our laser scan matcher. If the action plan takes the robot close to other nodes in the existing graph, the robot adds additional factors if the current distance in the estimated locations is smaller than a certain threshold, as has been already done in our SLAM framework. In [1] they use the number of FLIRT features in the laser scans to decide whether or not to add a new factor. To get the covariance matrices of the action pose graph, we use the iSAM2 library to optimise the action pose graph. Now we can compute the utilities and the best action $\boldsymbol{a}$ to take with Equation 3.5. For the computation of the D-optimality criteria we use the marginal covariance matrices as different nodes may have different α values. If the full covariance matrix is used, only a single α value would exist for the entire path. Additionally to the computation of the map subset $\boldsymbol{m}(\boldsymbol{a})$, we need to find the last node j in the action pose graph from which the cell was visible and then use that α_j to compute the information gain in that cell.

Analysis of α Function

In the proposed α function from Carrillo et al. [1], it was never justified why this function should work and if it has the desired effect. The function seems to be chosen only depending on the constraints of the utility function. But to be sure that the α function acts as desired, we need to analyse this more accurately. Let us do a simple numerical example to check if it is doing what is expected.
Assume we have a map cell with the probability $P(m) = 0.1$ of being occupied and a pose uncertainty with a standard deviation of the size of the map resolution, e.g. the standard deviation in x is equal to the map resolution and is zero for y and θ. This pose uncertainty results in $\sigma = 0.0025$. In Figure 3.5 you can see two plots, one for Rényi's entropy and the other for the α function using our numerical example values. For both functions we can see that for higher α values the graph is getting flatter. If we now try to find the corresponding α value for our σ value, we notice that it will be far outside of the plot range. If we compute the value we get $\alpha = 401$. If we now try to find the corresponding value for the Rényi entropy, we can see that if we have α values in this range, the Rényi entropy will be almost constant. This means that it will have almost no influence on the utility computation. We will only

Figure 3.6: The geometrical relation for the orientation error φ related to the map resolution (map_res) is illustrated here. l corresponds to the distance from the robot at which the error should be equal to the map resolution.

see a bigger influence, if we would have a lot of map cells with high uncertainties. But this should not be the case as we normally just plan actions in a sub-part of the environment we want to explore. If we would need to define a region for α where the effect on the utility has an influence, we could define this between $\alpha = 1$ and $\alpha = 20$.

Normalised α Function

We propose to use an α function, which scales depending on the map resolution. Our proposition uses a normalisation value to enforce the influence on the utility depending on how the normalisation is chosen. The function is defined as:

$$\alpha = 1 + \frac{\sigma_{norm}}{\sigma}, \tag{3.9}$$

where

$$\sigma_{norm} = \exp\left(\frac{1}{n}\left(2\log(map\_res^2) + \log\left(\underbrace{\tan^{-1}\left(\frac{map\_res}{l}\right)^2}_{\varphi}\right)\right)\right). \tag{3.10}$$

In Equation 3.10 we use the D-optimality (Equation 3.8) for the case where we would have a pose uncertainty with standard deviations of the map resolution in x, y and θ. For θ we defined an orientation error φ, which defines a distance l from the robot at which the error is equal to the map resolution. Figure 3.6 shows an illustration of the mathematical relation. In this thesis we used these parameters; $n = 3$, $map\_res = 0.05$ and $l = 10\,\text{m}$. Our α function causes α to have a value of 2, if we have a pose uncertainty in the range of the map resolution as defined for σ_{norm}.

3.3 Detection and Tracking of Moving People

In crowded environments the scene is often not static and is constantly changing. In this thesis we only focus on moving people detection and tracking. Changing environments like moving furniture, opening doors, etc. are not considered for simplicity. The detection of moving people in 2D laser scans is a hard task as it is even for a human hard to distinguish static objects from people only from laser scans. In the following sections we will present our method for detecting and tracking moving people and what extensions to real data are necessary.

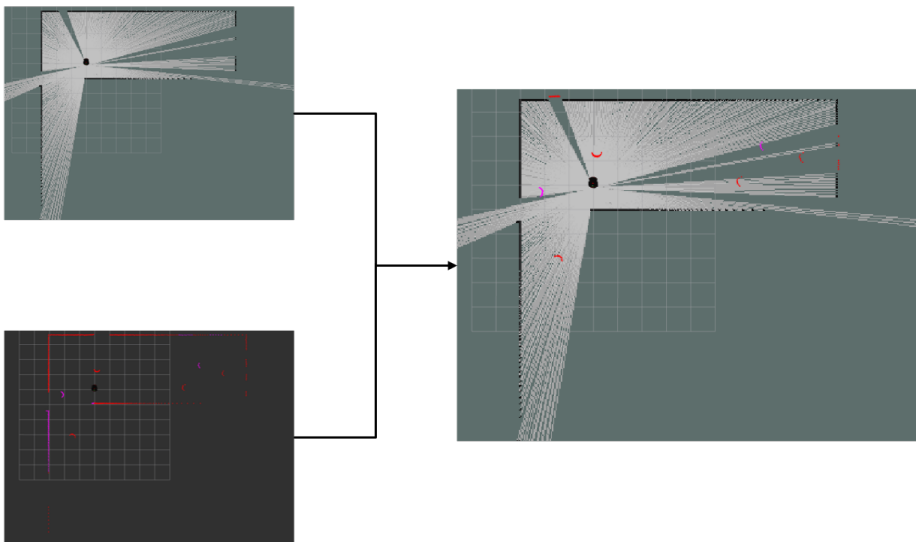


Figure 3.7: This figure shows how static objects in the current occupancy grid map (left upper image) of an exploration are removed from the current laser scan (left lower image). The resulting image (right image) shows the current occupancy grid map and the overlaid adapted laser scan.

3.3.1 Detection and Tracking of Moving People in 2D Laser Scans

To simplify the task of finding scan points in the laser data, which corresponds to moving people, we remove scan points that correspond to static objects, from the raw laser data. We can do this by using the current occupancy grid map we have built so far during exploration. Figure 3.7 shows this idea visually. This simplification assumes that our current occupancy grid map is correct, otherwise we would remove scan points, which could correspond to moving people.

At initialisation this method is not working as we do not have an occupancy grid map built so far. To overcome this issue we record the laser scans over a certain amount of time. During this time, we list the range data corresponding to each scan angle. We then search for the median range values of each angle and check if at least 50 % of the data is in a range around the median value. If this occurs, we assume that the median value corresponds to a static object and use these points in the laser scan data we send to our SLAM framework.

As soon as we have a candidate scan (with removed static scan points), we can cluster the data using an adaptive breakpoint detector [16]. Figure 3.8 shows an example of how the clustering could look. We additionally save the information of whether an object is occluded or not during clustering. This is done by verifying if on both sides of the clusters the following scan point is closer or further away from the robot. The raw laser data is used for this task, as the candidate scan might not have this information anymore after removing the static scan points. The occlusion information is essential to distinguish later if an object can be dynamic. For example when a clustered object is occluded, the centroid of the cluster could move over time, although it is static. This can be caused by objects moving in front of the occluded object or other occlusions, which occur during exploration.

We then compute the centroids of the clusters and use this information for tracking. We use individual Kalman Filters [43], which use a constant acceleration model. By using a Kalman Filter we can extract velocity estimates of the tracks during

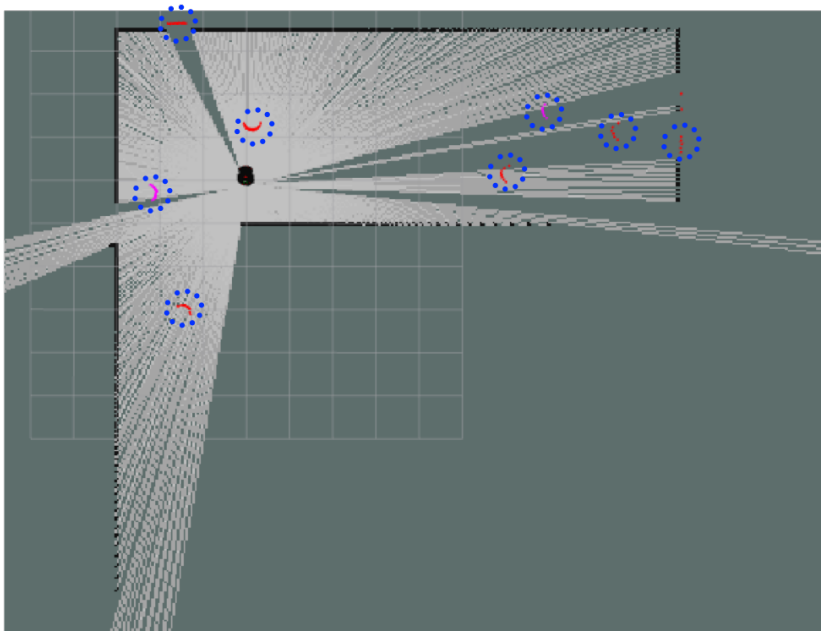


Figure 3.8: This figure shows an example how an adaptive breakpoint detector could cluster laser data, where the encircled laser points correspond to one clustered object.

exploration. This is important to distinguish if a track is assigned as dynamic or not. The data association during the update step in the Kalman Filter is currently solved by searching for the closest measurement around a track. Certainly, this is not the optimal way, but it was good enough for our application. To further improve the association, the Mahalanobis distance [44] could be used instead. If no measurements are available for updating a track, we wait for the next iteration. Tracks are only removed if no measurements are received over a longer time.
To distinguish if a track is dynamic or static we defined a process where the tracks need to pass some checks before they are assigned. Figure 3.9 shows the main process of these checks. In the beginning when a new track is initialised it is assigned as an unknown track. If a track is assigned as static or dynamic, the assignment cannot be changed anymore. The first check is a counter and checks for how long a track is already unknown. If the track is unknown long enough, it is assigned as static.

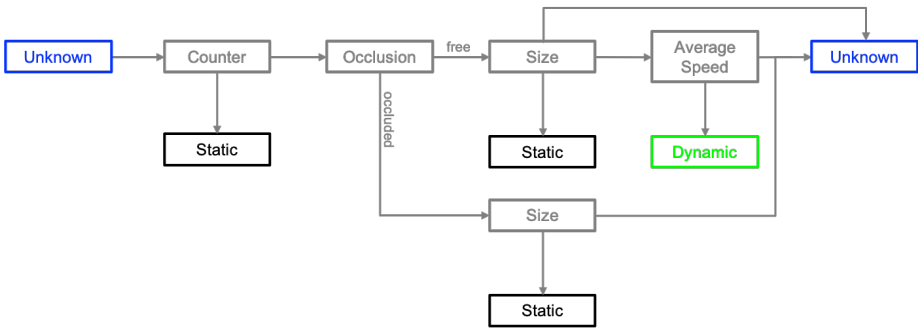


Figure 3.9: This figure shows the main process a track has to pass to be assigned as a dynamic or static track. If it was not able to assign the track it is kept as unknown and repeats this process in the next iteration.

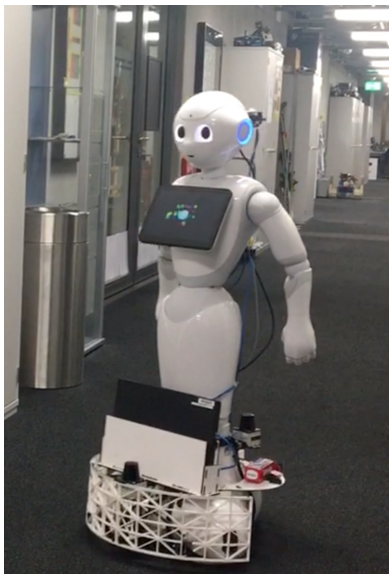


Figure 3.10: The Pepper robot is standing in a hallway at the Autonomous Systems Lab at ETH Zurich. In contrast to the original version of Pepper, this version has additional sensors and mountings attached.

The next check inspects the current cluster size of a track. If the size is bigger than the size of a person it is assigned as static. If only a few scan points are available, the track is kept as unknown. This is done to avoid having false dynamic detections caused by noisy measurements. If a track is not occluded and passed all size checks, the velocity estimates are used to detect if a track is dynamic. But to avoid noisy measurements resulting in high velocity estimate, the velocity is averaged over a certain amount of time before it is compared to a threshold value.

3.3.2 Extensions for Real Data

In this section we will discuss some extensions, which were implemented in order to improve performance with real data. Depending on the robot and sensors that are used in a real environment some laser pre-processing steps are necessary. Also the assumption of people being approximated as cylinders doesn't hold in general.

Pre-processing Laser Scans

As can be seen in Figure 3.10 the Pepper robot has two 2D LIDARs mounted at leg height, one in front and one in the back. As the active SLAM framework built in this thesis assumes only one 2D LIDAR with a 360 ° range, some pre-processing is needed. Because of some mountings obstructing the free observation of the environment, the scans need to be cropped appropriately. As real laser scan data are noisier than simulation data, e.g. caused by reflections or other undesired disturbances, some filtering is needed. To improve the quality of the scans a simple median filter[12] is used. The quality of the laser scans is essential for active SLAM as too much noise in the data can result in sending waypoints, which are not reachable. Finally as there are two 2D LIDARs the scans need to be combined. This is solved by transforming the back scan into the front laser frame.

[12] `http://wiki.ros.org/filters/MedianFilter`, (Accessed on 20.03.2019)

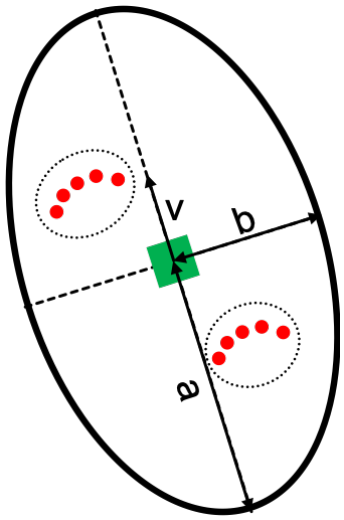


Figure 3.11: The adaptive people ellipse merges leg clusters together. The green block represents the current dynamic track, around which the adaptive people ellipse is used. The red points show laser scan points corresponding to the legs, where the clusters containing all scan points of a leg are encircled.

Adaptive People Ellipse

As mentioned above the approximation of people being cylinders does not really hold if the LIDAR is at leg height. If the LIDAR was at waist height, then the assumption would hold quite well. With the current framework, the detection of legs is still possible, but it is hard to keep track of each leg. Further, the stop-and-go movements of the legs make it impossible for the local planner to include the dynamic obstacles into the planning. To solve this issue, it is necessary to know which clusters correspond to a person. In this thesis, we use an adaptive ellipse, which adapts its form depending on the current estimated velocity and moving direction. Figure 3.11 shows the ellipse around a dynamic track.

The requirement for the usage of this method is having already a track, which was assigned as dynamic. Once this requirement is fulfilled, all clusters, which are in the ellipse, are merged together and the merged centroid is then used to update the track. The formula for the ellipse can be defined as:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \tag{3.11}$$

where

$$a = c\left(1 + \frac{v}{v_{norm}}\right), \quad b = c\left(1 + d\frac{v}{v_{norm}}\right) \tag{3.12}$$

describe the adaptive axes and where v corresponds the norm of the velocity. The parameters c, d and v_{norm} depend on the proportions of a human. We found that $c = 0.35\,\text{m}$, $d = 0.1$ and $v_{norm} = 0.8\,\text{m/s}$ were good approximations for an average-sized person.

3.4 Simulation Environment

For algorithm development and testing purposes a simulation environment has been used. We chose the well-known simulation engine Gazebo[13]. Up to the starting time of this thesis no simulation model of Pepper existed. Because of this and for time reasons a similar mobile robot has been used in the simulation. A Pioneer 3-DX has been used, as there exist a few open source simulation models.[14] The used model has a 2D LIDAR mounted on top. The Pepper robot has in general no LIDAR mounted, but has been extended in parallel in a semester thesis, such that both mobile robots have similar sensor configurations.

3.4.1 Simulated Robot and Environments

As described above a Pioneer 3-DX has been used in simulation. In contrast to the Pepper robot, which is a holonomic robot, the Pioneer is a differential drive robot. In this case, the Pioneer is a nonholonomic robot, which has a castor wheel in the back. A nonholonomic robot is not able to drive freely in any direction and can increase the difficulty of different tasks. But it can be assumed that if an algorithm works for a nonholonomic robot, it is also working for holonomic robots.
For testing purposes multiple worlds have been generated. In Figure 3.13 you can see five worlds, which have been used for testing.

3.4.2 People Simulator

To simulate a crowded environment, a people simulator has been developed. It is based on a simulation used in the work of Pfeiffer et al. [45], which uses a social forces model for pedestrian dynamics [46]. In particular, the interaction between robot and people has been added and for simplicity the same social forces model has been used. For testing purposes, we can define how many people are spawned in the world and test how our framework behaves. People are moving randomly in the world by following a random goal. In Figure 3.12 an example of the people simulator is shown.

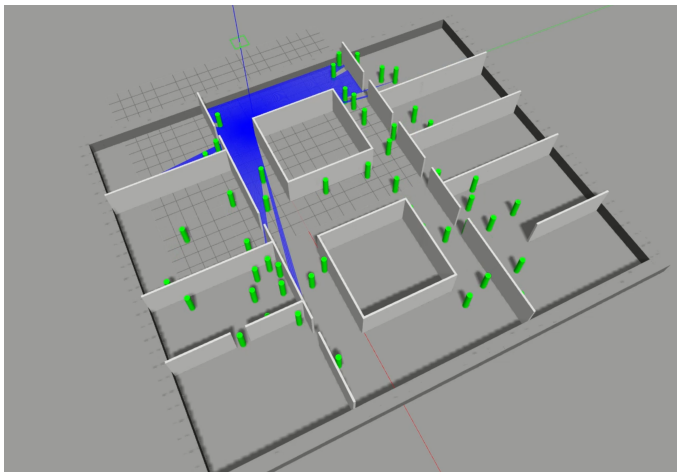
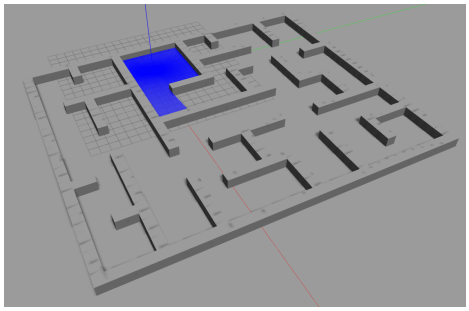
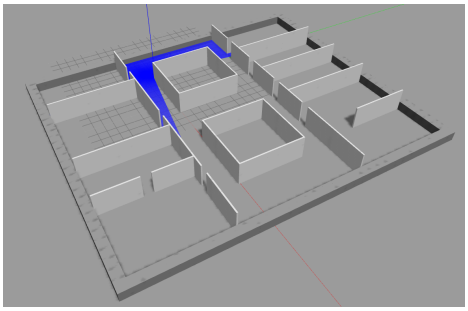


Figure 3.12: The people simulator simulates 50 people (green cylinders) in the Lab world.

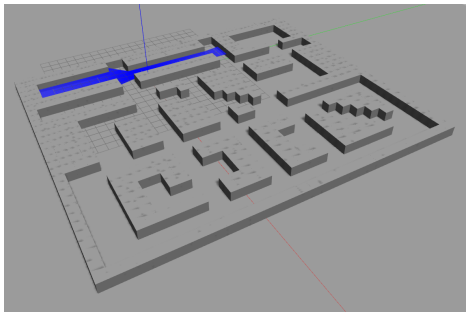
[13] `http://gazebosim.org`, (Accessed on 04.01.2019)
[14] Implementation is based on `https://github.com/JenJenChung/pioneer_description`, (Accessed on 04.01.2019)



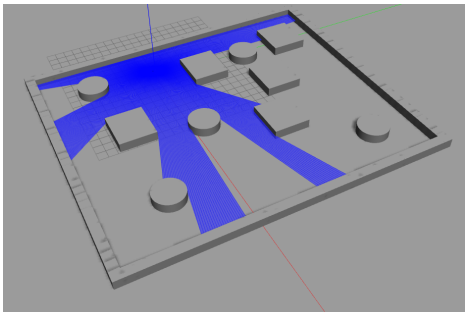
(a) Closed Maze



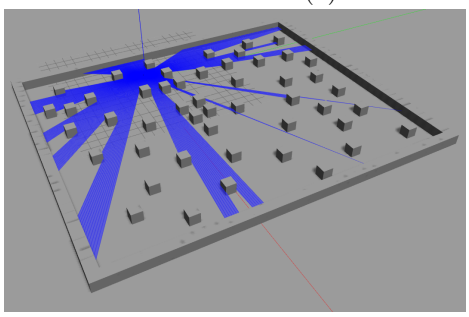
(b) Lab



(c) UTM



(d) Random Boxes and Cylinders



(e) Random Boxes

Figure 3.13: The worlds used for testing and algorithm development are shown in the Gazebo simulator. They have been chosen and built such that most scenarios the robot will see are covered. (a) is a closed maze world where bigger loop closings happens less often than in other worlds. (b) shows a world, which has similarities to the Autonomous Systems Lab at ETH Zurich. (c) shows a suburban street map used for UAV traffic management (UTM). (d) and (e) are random maps built from boxes and cylinders placed at random positions. The blue areas in the worlds demonstrate the LIDAR rays of the Pioneer 3-DX.

Chapter 4

Results

In this section we present test results obtained in our simulation environment and in a real environment. Section 4.1 describes the metrics used for the evaluation of the test results in Section 4.2.1. Section 3.4 describes the simulation environment, where the algorithms have been tested. Section 4.2 presents the test results in the simulation environment for the static and crowded case. Section 4.3 presents test results in a real environment using the Pepper robot and shows static and crowded scenarios.

4.1 Presentation of Results

For the interpretation of test results meaningful metrics are necessary. In SLAM the definition of such metrics is still a hard task and often depends on the purpose of the work. Especially for map metrics there does not exist a general metric that describes the performance of a SLAM algorithm. In Section 4.2 the reader can observe some test results, which use the below mentioned metrics. To prevent getting test-run-specific results instead an overall behaviour interpretation of the system, the tests have been repeated for at least five runs for each strategy and world.

4.1.1 Pose Metrics

The definition of a metric for the pose estimates in a SLAM algorithm is more straightforward than for the map estimates. In the case of a simulation environment, we could define our metric as the error between ground truth position and the estimated position. But there exists a much more meaningful metric, which doesn't need a ground truth position of the robot and is also applicable in a real environment. As we can easily extract the covariance matrix of the current node in our pose graph, we can compute σ, the D-optimality criteria, of the current covariance matrix, which then describes as a scalar the general uncertainty in the current pose estimate. Clearly, this assumes that the covariance matrices are correct, which is highly dependent on the covariance estimation in our laser scan matcher described in Section 3.1.1. We can plot the progression of the uncertainty along the length of the path.

4.1.2 Map Metrics

As map metric, we defined three different metrics, which compare the generated map with the ground truth map. For all three metrics, a lower score indicates a map that is closer to the ground truth map. The first metric is a simple sum of

squared errors of the occupancy probability in the corresponding map cells, which we call the *map_score* [47] metric:

$$map\_score = \sum_{i,j}(P(m_{i,j}) - P(n_{i,j}))^2, \tag{4.1}$$

where $P(m_{i,j})$ and $P(n_{i,j})$ are the occupancy probabilities of the maps $\boldsymbol{m}$ and $\boldsymbol{n}$ at indices i and j. The problem with this score is that it penalises the mapping behaviour. As the mapping algorithm tends to build maps with thicker walls, when the same region is visited more often, the score increases. See Section 3.1.3 for further information on the mapping behaviour. As the mapping behaviour has no influence on the map quality, a better metric is necessary to describe the map quality. To handle errors induced by the mapping behaviour, we defined a signed distance field (SDF) score, which should decrease the impact of wall thickness on the score. The *SDF_score* is defined as:

$$SDF\_score = \sum_{i,j}(s_{i,j} - t_{i,j})^2, \tag{4.2}$$

where $\boldsymbol{s}$ and $\boldsymbol{t}$ correspond to the distance values in the SDF maps. Before we can compute the SDF score, we need to compute the SDF maps of the estimated occupancy grid map and the ground truth map. Each distance value corresponding to a cell describes in our case the distance in number of cells to the closest wall cell. If a cell corresponds to a wall cell, the negative distance to the closest wall cell, which is adjacent to a free cell, is computed. For simplicity, unknown cells are treated as wall cells. To further enforce the score on navigationally important areas, a relaxed SDF score has been defined. The relaxed SDF score uses two relaxations. The first relaxation handles errors caused by wall shifts as induced for example by our mapping behaviour, while the second focuses map regions that are important for navigation. The score is formulated as:

$$rel\_SDF\_score = \sum_{i,j \in IJ_{nav}} \begin{cases} (|s_{i,j} - t_{i,j}| - 1)^2, & \text{if } |s_{i,j} - t_{i,j}| > 1. \\ 0, & \text{otherwise.} \end{cases}, \tag{4.3}$$

where $s_{i,j}$ and $t_{i,j}$ are the distances at indices i and j of the SDF maps. In case of $|s_{i,j} - t_{i,j}| > 1$ we subtract one cell number from the current distance error to relax the score. The set IJ_{nav} corresponds to relaxed map region indices, which focus on navigationally important areas. These areas correspond to regions, which are not unknown in the ground truth map. We consider wall cells, which are not adjacent to a free cell, as unknown cells and are thus not considered in the score.

4.2 Simulation

In this section we describe the performance of our utility approach compared to a simple shortest frontier strategy in the simulation environment. The performance is evaluated against map and pose uncertainty performance. Section 4.2.1 presents the results for the static case, while Section 4.2.2 presents results for crowded environments. For the test evaluation the worlds presented in Section 3.4.1 have been used.

4.2.1 Performance in Static Environments

For the evaluation of map quality and pose uncertainty performance, five test runs for each world and each strategy have been done. In Figure 4.1 resulting maps in the lab world are shown as example. It can be observed that the utility strategy

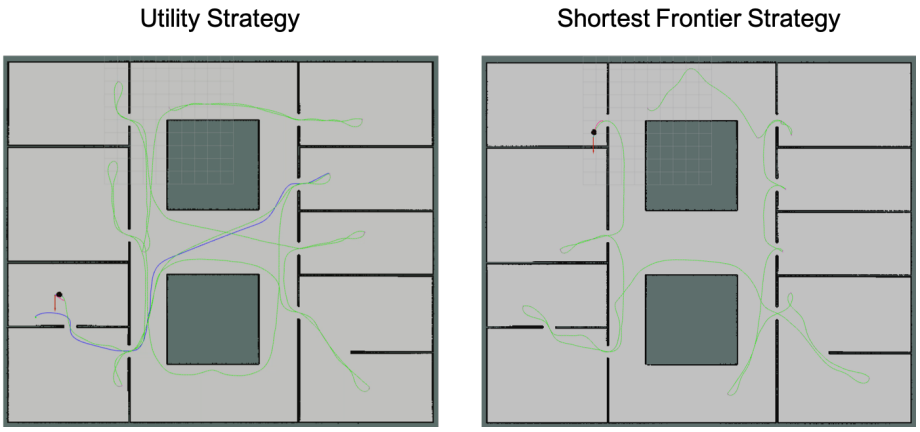


Figure 4.1: An example of resulting maps for the shortest frontier and the utility strategy in the Lab world is presented here. The green lines correspond to the paths executed by the Pioneer in the simulation. The blue path corresponds to a leftover path while computing the utility of this action.

revisits previously observed regions more often. But as can be observed, it is hard to find any map differences in the images. Because of this, the quantitative evaluation was performed using the map metrics defined in Section 4.1.2.

In Table 4.1 the map evaluation for all test runs is shown. We can observe that in general the exploration times for the utility strategy are higher than for the shortest frontier strategy. This is caused by the behaviour of the utility strategy, which always tries to decrease map and pose uncertainty in an optimal way. To decrease both uncertainties in an optimal way, revisiting already seen regions is essential. We can observe that the utility strategy achieves better results in the Lab and UTM world compared to the others. This could be the result of the bigger loop closings, which are possible in the Lab and UTM world. In general, the results do not show a clear tendency for better results achieved by the utility strategy.

In Figure 4.2 the development of σ along the path length for all Lab tests is shown. It can be observed that the σ values stay very low for both strategies. A growth tendency for the shortest frontier strategy can be observed while the utility strategy seems to stay bounded.

We can further see the behaviour of the strategies when observing the paths executed by the Pioneer, where the chosen actions differ clearly. The utility strategy enforces revisiting already seen regions more often to decrease its pose uncertainty. The shortest frontier strategy instead is not able to close bigger loops as it is pursuing the closest goal. Although we would think that the shortest frontier strategy would achieve worse map results caused by the missing loop closing, the map results in Table 4.1 do not clearly represent the expected behaviour. This can be explained by analysing Figure 4.2, where we can observe that independent of the strategy the pose uncertainties stay very low. The standard deviations of the pose uncertainties are much lower than our map resolution. Although in the simulation we use white noise on the laser scans with a standard deviation of 0.01 m, this seems not to have enough impact on the performance. Because of this we had to change to hardware tests using Pepper to show that the utility strategy can achieve better results than a simple shortest frontier strategy. The test results for Pepper are presented in Section 4.3. Further, it is worth mentioning that although the utility strategy revisits already seen regions more often, as can be observed in Figure 4.1, this is mostly caused by the map uncertainty instead of the pose uncertainty. When we

	Closed Maze		
	Shortest Frontier	Utility	Difference (%)
Exploration Time	309.3416	535.3938	73.08
Map Score	3029.928	3361.854	10.95
SDF Score Full	20014.24	29145.02	45.62
SDF Score Relaxed	340.7926	311.0176	-9.57
	Lab		
	Shortest Frontier	Utility	Difference (%)
Exploration Time	182.7874	303.0444	65.79
Map Score	3292.426	3021.506	-8.97
SDF Score Full	37227.86	17580.24	-111.76
SDF Score Relaxed	216.45974	94.7785	-128.38
	Random Boxes and Cylinder		
	Shortest Frontier	Utility	Difference (%)
Exploration Time	220.6572	319.0164	44.58
Map Score	1702.314	1748.006	2.68
SDF Score Full	19769.74	23086.46	16.78
SDF Score Relaxed	190.3798	204.0614	7.19
	Random Boxes		
	Shortest Frontier	Utility	Difference (%)
Exploration Time	298.5364	453.9934	52.07
Map Score	2503.804	2277.326	-9.94
SDF Score Full	59441.98	40365.44	-47.26
SDF Score Relaxed	947.2146	966.6586	2.05
	UTM		
	Shortest Frontier	Utility	Difference (%)
Exploration Time	295.9834	440.1432	48.71
Map Score	3707.704	3524.11	-5.21
SDF Score Full	93210.26	60568.1	-53.89
SDF Score Relaxed	645.6154	402.1316	-60.55

Table 4.1: Evaluation of test results in the Closed Maze, Lab, Random Boxes and Cylinder, Random Boxes and UTM world. The utility and shortest frontier strategy are compared. The right column shows the difference of these strategies in percent, where a negative value signifies the better score for the utility strategy. Exploration time, map score, SDF score (Full) and relaxed SDF score are used for evaluation.

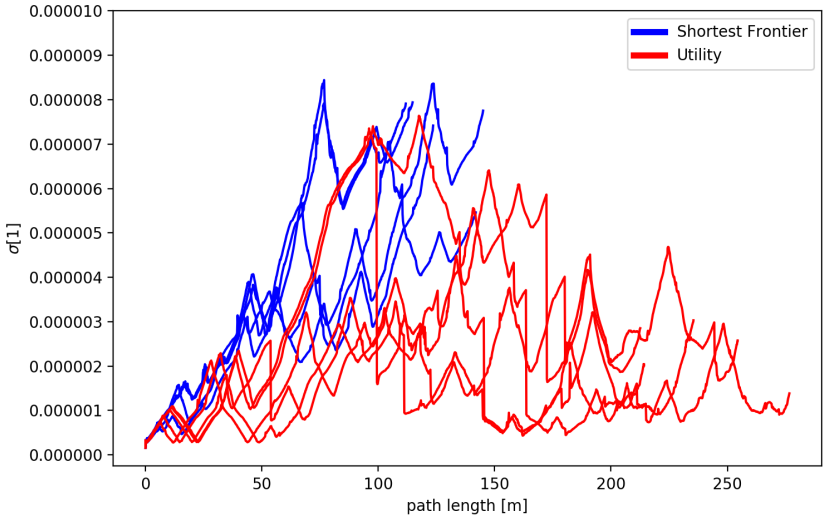


Figure 4.2: The development of σ along the path lengths for all test runs in the Lab world is plotted here.

have such a low pose uncertainty the utility strategy tends to chose actions, which reduce the most map uncertainties, i.e. the most uncertain map cells which will be seen along the action plan. This choice of the action plan can also cause the revisiting behaviour.

4.2.2 Performance in Crowded Environments

If we would use the active SLAM framework for static environments in a crowded environment, the result could look like the maps shown in Figure 4.3. Moving people are assumed to be static and handled during exploration as static objects. This results in severe errors and map shifts. Not handling moving people appropriately can also cause the failure of the exploration, as for example the navigation system could fail because of the wrong occupancy grid map built so far.
If we handle crowded environments appropriately, as presented in Section 3.3.1, the map results can look like the one presented in Figure 4.4. As can be observed, the map result is comparable to the map results in the static case. After evaluating these first test results, we recognised that the performance was almost as good as for the static case. For this reason, we did not repeat the evaluation as done in Section 4.2.1 and switched to hardware tests as described in Section 4.3.

4.3 Tests with Pepper in Real Environment

The tests with Pepper were conducted in the Autonomous Systems Lab at ETH Zurich. Section 4.3.1 presents test results for the static case, while Section 4.3.2 presents test results for the crowded case.

4.3.1 Static Environment

The results for the static case have been recorded without any people moving in the lab (except the one who was conducting the tests). As we use only 2D laser

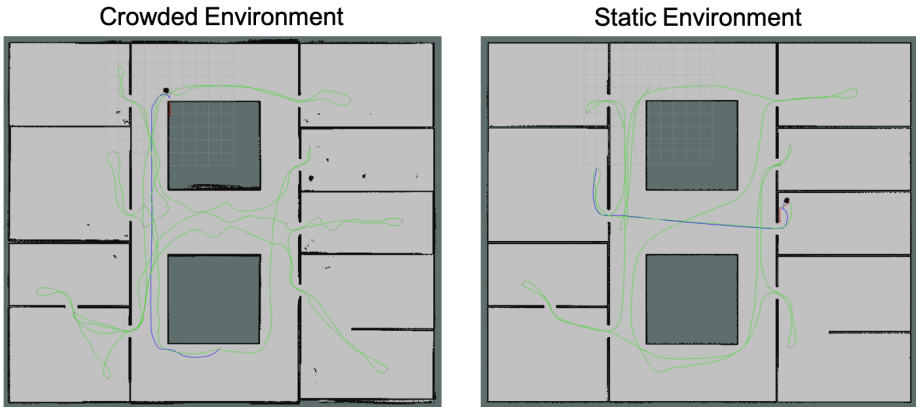


Figure 4.3: The resulting maps in a crowded and static environment of the Lab world are compared when using the active SLAM framework for static environments only.

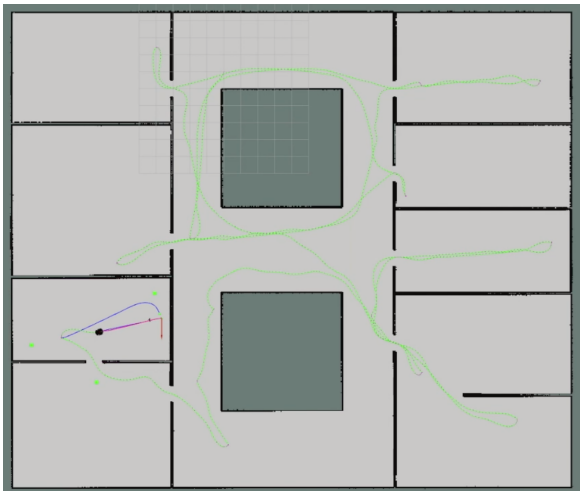


Figure 4.4: The final map in a crowded lab environment is shown. The green blocks correspond to the detected and tracked moving people.

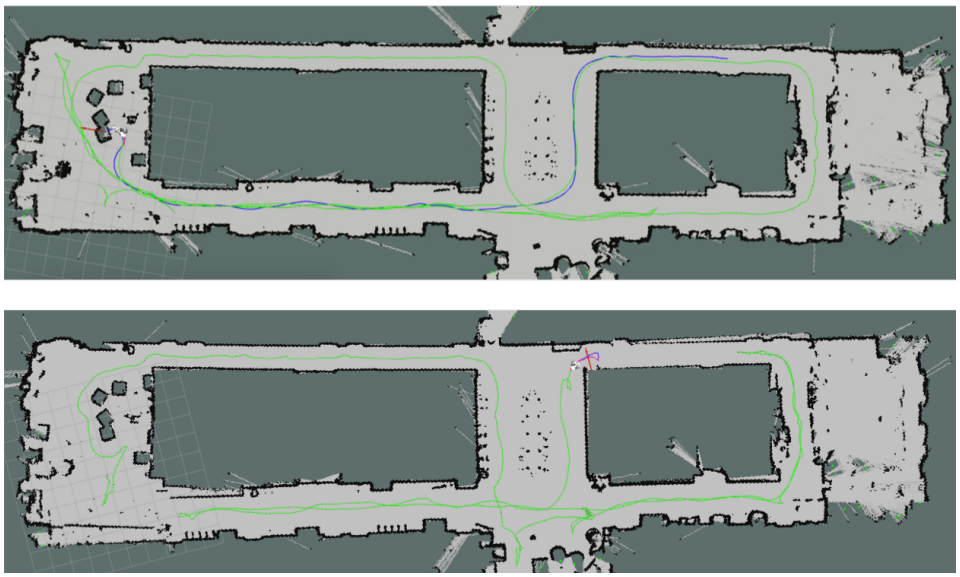


Figure 4.5: The resulting maps in the Autonomous Systems Lab at ETH Zurich are presented for a static environment using the utility strategy (above) and the shortest frontier strategy (below).

scans, some adaptations to the environment needed to be done. All glass doors had to be covered at laser height, as otherwise Pepper would want to explore regions by driving through the door. We also had to make sure that no bigger regions are at laser height if it is not possible to pass through these regions. This can be for example caused by tables or obstacles below the laser height. Figure 4.5 shows a comparison of two resulting maps using the utility and the frontier strategy. You can again clearly see that the utility strategy revisits already seen regions more often and is able to close bigger loops in contrast to the shortest frontier strategy. In the resulting map for the shortest frontier strategy these missing loop closings cause severe map shifts. This shows that the utility strategy can achieve better results than the shortest frontier strategy.
It is clear that more test results would be needed to show the efficacy of the utility function, but because of time reasons these time-consuming tests have not been done. It also seems appropriate to claim that our utility function will achieve better results. One thing that we noticed during testing was that sometimes the utility strategy could produces bad maps too. This is mostly caused because of missing actions to essential loop closing areas. This can be caused by missing frontiers, as the areas have been explored already. This is not a wrong choice of actions per se, but caused by the frontier exploration strategy itself and the fact that only one path to a frontier is considered. To improve this behaviour loop closing areas could be included as possible goals for actions. Alternative paths to the same frontier could further favour the performance.

4.3.2 Crowded Environment

For time reasons, only tests using the utility function have been done in the crowded case. Figure 4.6 shows a screen shot during exploration where three people were walking around Pepper in a narrow scene. The three people are tracked and included into the local planner and as can be observed it finds a path to reach its goal.
Figure 4.7 shows a map result using the utility strategy in a crowded environment.

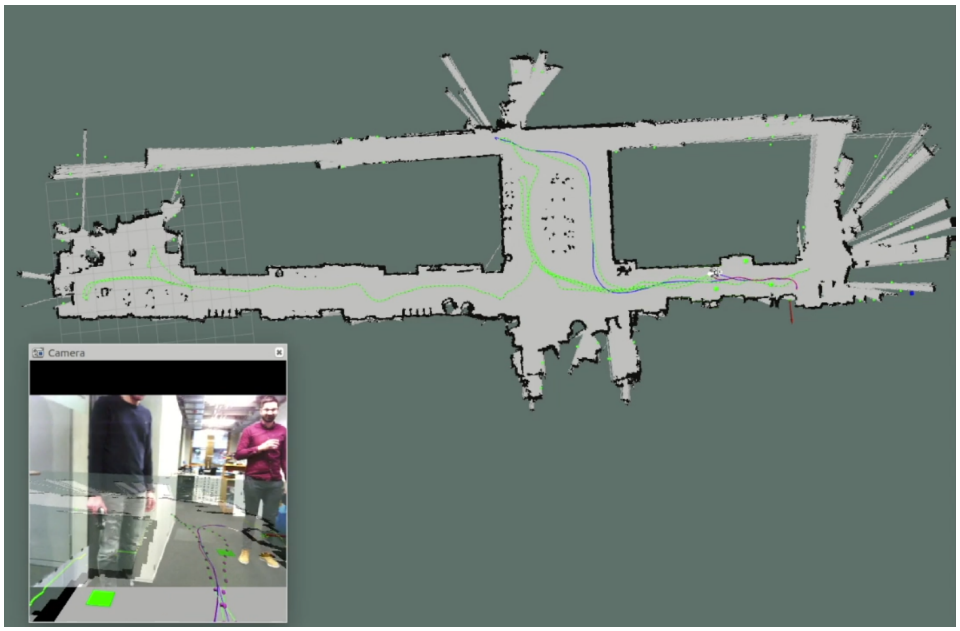


Figure 4.6: Screen shot of an exploration run with Pepper where three people were walking around Pepper. The green blocks correspond to the detected and tracked moving people. The red path corresponds to the local plan computed to the current goal.

The exploration was actually successful, but one can observe that the map is not complete. This has been caused by an unfortunate coincidence. One laser beam has been used while mapping, which passes both frontier regions in the long and narrow hallway. This caused the frontier computation to cluster both frontier regions together. The computed centroid is then close to the mapped laser beam line, as can be observed in the figure. The global planner is in this case not able to find a path to this frontier, as it only plans paths in known areas. This caused the error in the map. As can be observed, the implementation is not perfect and unexpected occurrences can lead to failures or undesired results.
During testing, further weak points stood out. As only laser odometry is used in the SLAM framework, the long and narrow hallway can cause problems. Especially while detecting and tracking moving people, we further remove scan points from our scan data. This means our laser scan matcher has even less information, which could help to get the correct roto-translational parameters in a long, narrow hallway. Laser odometry failures often happen when only laser scan points corresponding to the walls of a hallway and no other object at the beginning or end of the hallway are available. The detection of moving people can sometimes cause some issues, as the averaging of the velocity takes some time. If people walk too fast, the currently unknown leg tracks can be lost and reinitialised, such that there has not been enough time for the averaging of the velocity. Also fast changing moving directions of people can cause a failure in tracking them.

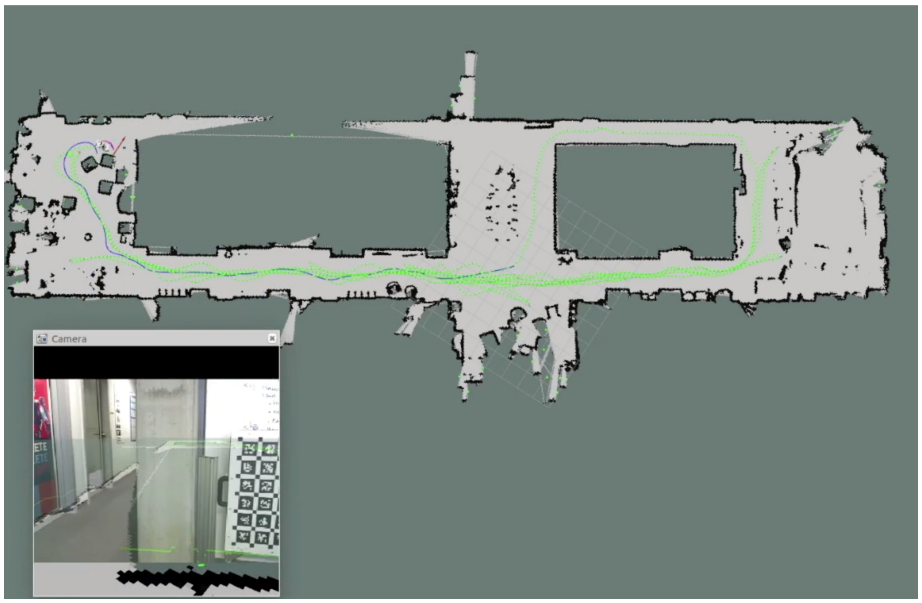


Figure 4.7: Screen shot of a resulting map using the utility strategy in the Autonomous Systems Lab at ETH Zurich. The environment can be assumed as crowded during exploration.

Chapter 5

Conclusion

In this thesis we developed an active SLAM algorithm for static environments, which has been extended to crowded environments by implementing a detection and tracking algorithm for moving people. We could show that the trade off between exploration and exploitation can be automated without the need for manually tuning parameters. Although there is no tuning necessary, the correct relation between map and pose uncertainties in active SLAM is still an open issue. We could partially solve this issue with the definition of our α function, but it only has a scaling relation to the map resolution. The scaling using the map resolution can, in our opinion, be seen as a design choice, as it prevents having pose uncertainties, which are higher than the map resolution during exploration. This is not a requirement of an optimal exploration strategy, since it is possible that an action, which causes you to temporarily have a bigger pose uncertainty, can result in a better map result and localisation in the long run. The goal would be to find a general relation between map and pose uncertainties for active SLAM.
Although detecting and tracking moving people is a hard task, we showed that the integration into active SLAM can clearly improve the performance. The proposed method for detecting and tracking moving people might not be the most novel one, but it fulfilled its purpose in this thesis.

5.1 Outlook

In future, many improvements could be made. As mentioned above and in Section 4.3.2 the detection and tracking of moving people has a lot of improvement potential. Instead of trying to improve the current implementation, a deep learning based approach could be implemented, similar to the work of Beyer et al. [24][25]. This would bring the advantage that people could be also detected when they are not moving. Otherwise the inclusion of further sensors, like RGB cameras, would be necessary to detect people when they are not moving.
To avoid future problems with long hallways, the inclusion of robot odometry or feature scan matching using RGB cameras could be helpful.
Further, the detected moving people information could be used to improve the performance of our algorithm by including this information into SLAM and the utility computation instead of treating them as outliers. A highly crowded environment can cause the current framework to fail, as the laser data may not have enough information after the detected moving people are removed. Including the detected moving people into the utility computation would open a lot of solution possibilities. One idea would be to include the future occlusion caused by moving people into the utility computation.

To avoid missing important loop closings caused by not having enough actions, the action generation could be extended for more variability in the paths. Using loop closing areas as further goal points and computing alternative paths to the same goal could improve the general performance of active SLAM.

Bibliography

[1] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy," *Autonomous Robots*, vol. 42, no. 2, pp. 235–256, 2018.

[2] S.-Y. An, L.-K. Lee, and S.-Y. Oh, "Ceiling vision-based active SLAM framework for dynamic and wide-open environments," *Autonomous Robots*, vol. 40, no. 2, pp. 291–324, Feb 2016.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] A. Pázman, *Foundations of optimum experimental design.* Springer, 1986, vol. 14.

[5] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* IEEE, 2012, pp. 2080–2087.

[6] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms.* Cambridge university press, 2003.

[7] A. Rényi, "On measures of entropy and information," HUNGARIAN ACADEMY OF SCIENCES Budapest Hungary, Tech. Rep., 1961.

[8] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters." in *Robotics: Science and Systems*, vol. 2, 2005, pp. 65–72.

[9] C. Leung, S. Huang, and G. Dissanayake, "Active SLAM using model predictive control and attractor based exploration," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* IEEE, 2006, pp. 5026–5031.

[10] R. Valencia, J. V. Miró, G. Dissanayake, and J. Andrade-Cetto, "Active Pose SLAM," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 1885–1891.

[11] J. Vallvé and J. Andrade-Cetto, "Active pose SLAM with RRT," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015, pp. 2167–2173.

[12] B. Mu, M. Giamou, L. Paull, A.-a. Agha-mohammadi, J. Leonard, and J. How, "Information-based active SLAM via topological feature graphs," in *Decision and Control (CDC), 2016 IEEE 55th Conference on.* IEEE, 2016, pp. 5583–5590.

[13] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review," *arXiv preprint arXiv:1409.7618*, 2014.

[14] J. Burlet, T. D. Vu, and O. Aycard, "Grid-based localization and online mapping with moving object detection and tracking," Ph.D. dissertation, INRIA, 2006.

[15] M. S. Bahraini, M. Bozorg, and A. B. Rad, "SLAM in dynamic environments via ML-RANSAC," *Mechatronics*, vol. 49, pp. 105–118, 2018.

[16] G. A. Borges and M.-J. Aldon, "Line extraction in 2D range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004.

[17] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association," in *Proceedings of Robotics: Science and Systems*, 2007.

[18] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph SLAM: Long-term mapping in low dynamic environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1871–1878.

[19] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[20] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe *et al.*, "Moving object detection with laser scanners," *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.

[21] R. MacLachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006, pp. 301–306.

[22] L. E. Navarro-Serment, C. Mertz, N. Vandapel, and M. Hebert, "LADAR-based pedestrian detection and tracking," 2008.

[23] J. Yin, L. Carlone, S. Rosa, M. L. Anjum, and B. Bona, "Scan matching for graph SLAM in indoor dynamic scenarios," in *The Twenty-Seventh International Flairs Conference*, 2014.

[24] L. Beyer, A. Hermans, and B. Leibe, "DROW: Real-time deep learning-based wheelchair detection in 2-D range data," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 585–592, 2017.

[25] L. Beyer, A. Hermans, T. Linder, K. O. Arras, and B. Leibe, "Deep person detection in 2D range data," *CoRR*, vol. abs/1804.02463, 2018.

[26] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2432–2437.

[27] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[29] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[30] A. Censi, "An ICP variant using a point-to-line metric," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on.* IEEE, 2008, pp. 19–25.

[31] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[32] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Robotics and Automation, 2007 IEEE International Conference on.* IEEE, 2007, pp. 3167–3172.

[33] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[34] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[35] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[36] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* MIT press, 2005.

[37] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on.* IEEE, 1997, pp. 146–151.

[38] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics.* VDE, 2012, pp. 1–6.

[39] ——, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots.* IEEE, 2013, pp. 138–143.

[40] G. Jumarie, *Relative Information: Theories and Applications.* Berlin, Heidelberg: Springer-Verlag, 1990.

[41] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.

[42] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

[43] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[44] M. P. Chandra *et al.*, "On the generalised distance in statistics," in *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, 1936, pp. 49–55.

[45] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[46] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, p. 4282, 1995.

[47] T. Colleens and J. Colleens, "Occupancy grid mapping: An empirical evaluation," in *Control & Automation, 2007. MED'07. Mediterranean Conference on*. IEEE, 2007, pp. 1–6.