# Multi-Robot Navigation and Cooperative Mapping in a Circular Topology

Simon Bultmann, Laëtitia Matignon, Olivier Simonin

# Multi-Robot Navigation and Cooperative Mapping in a Circular Topology

**Research Report - July 2016**

Simon Bultmann[1,2], Laetitia Matignon[2,3], and Olivier Simonin[1,2]

[1] INSA Lyon, Université de Lyon, 20 Avenue Albert Einstein, 69100, Villeurbanne, France
[2] CITI Lab. Inria Chroma team, INSA Lyon 6 avenue des Arts, 69680 Villeurbanne cedex, France
[3] Univ Lyon, Université Lyon 1, LIRIS, CNRS, UMR5205 Villeurbanne, F-69622, France

**Abstract.** Cooperative mapping of an environment by a team of multiple robots is an important problem to advance autonomous robot tasks for example in the field of service robotics or emergency assistance. A precise, global overview of the area the robots are working in, and the ability to navigate this area while avoiding obstacles and collisions between robots is a fundamental requirement for a large number of higher level robot-tasks in those domains. A cooperative mapping, navigation and communication framework supposing unknown initial relative robot positions is developed in this project based on the ROS libraries. It realizes robot displacement, localization and mapping under realistic real-world conditions. Such, the framework provides the underlying functions needed to realize a task of human activity observation in the future. Initially, local maps are individually constructed by the robots using the common *gmapping* SLAM algorithm from the ROS libraries. The robots are evolving on circles around the scene keeping a constant distance towards it or they can change radius, for example to circumvent obstacles. Local maps are continuously tried to align to compute a joint, global representation of the environment. The hypothesis of a common center point shared between the robots greatly facilitates this task, as the translation between local maps is inherently known and only the rotation has to be found. The map-merging is realized by adapting several methods known in literature to our specific topology. The developed framework is verified and evaluated in real-world scenarios using a team of three robots. Commonly available low-cost robot hardware is utilized. Good performances are reached in multiple scenarios, allowing the robots to construct a global overview by merging their limited local views of the scene.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the domain of service robotics or emergency assistance the observation of human dynamic scenes by a team of multiple robots is an important task. Activities of the persons need to be recognized and multiple points of view are necessary for a joint observation of high quality. This student research project proposes to make use of a concentric, circular topology with a common center point for such an application. A fundamental requirement of those high-level autonomous operations is a precise, global overview of the area the robots are working in, and the ability to navigate this area while avoiding obstacles and collisions between robots. In realistic real-world scenarios, no maps are available prior to the beginning of the task and no information on the relative initial positions between the robots is known. This creates a problem of high complexity, however, the concentric topology with a common center point shared between the robots allows for several simplifying hypotheses that will be exploited during this work. Based on this particular topology, a framework for robot navigation, localization, communication and cooperative mapping is developed. It provides the underlying functionalities required for the imagined high-level task. Figure 1.1 gives an example of the experiments realized with three robots in real-world scenarios.

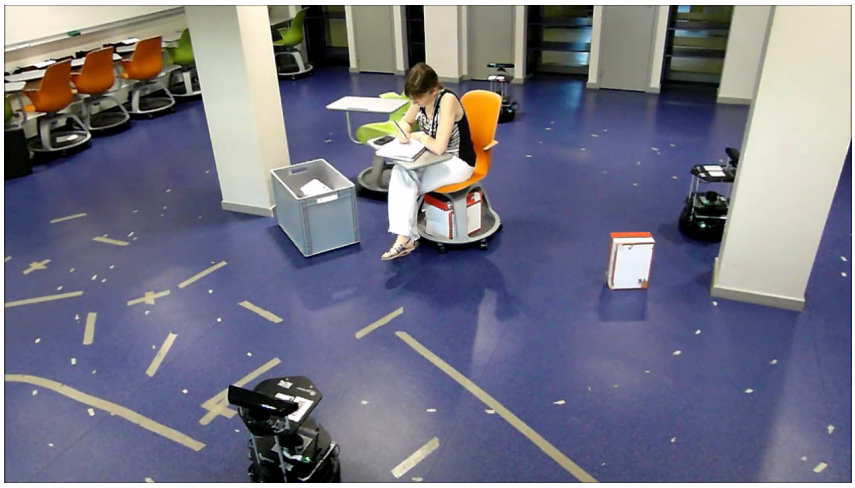


**Figure 1.1:** Picture of the robot task realized during the project: A team of three robots cooperatively explores the environment around a scene containing a human person performing a certain activity.

The framework is developed based on the Robot Operating System (ROS) libraries. It uses the *gmapping* SLAM algorithm for the construction of local maps from laser range-scan data. Functions for circular movement around the scene and radius change, moving towards or away from the scene, are provided as well as obstacle recognition and communication abilities. Several algorithms to merge local maps into a global representation of the environment are adopted from literature and implemented making use of the simplifying hypotheses that result from the circular topology with a common center point shared between the robots. Real-world experiments are realized with a team of three robots using commonly available low-cost hardware.

The report is organized in four chapters. After this introduction, chapter 2 will give an overview of the state of the art of some relevant parts of the vast domain of Simultaneous Localization and Mapping for mobile robots. The focus will be on occupancy grid map representations and particle filter SLAM algorithms as they are used in the practical part of the project. Several approaches on multi-robot SLAM are also reviewed, as a central point of this project was the realization of a cooperative mapping task using multiple robots.
Chapter 3 will detail the work realized during the project. It will describe the developed robot navigation and communication framework as well as the map-merging approaches adapted to the circular topology with a common, shared center point. Results on several real-world scenarios using three low-cost Turtlebot robots are presented.
Lastly, chapter 4 summarizes the achievements of this project and gives an overview of future work.
The annexes A and B resume some topics less related to the concrete work of this project than those presented in the state of the art in chapter 2. Multi-Robot SLAM based on topological and hybrid maps are presented as well as some strategies for path planning in a cooperative robot task.

# Chapter 2

# Simultaneous Localization and Mapping

For a mobile robot to be able to navigate an unknown environment it needs to be able to autonomously learn a map. The mapping problem however includes both estimating the robot's position relative to the map and updating the map with the sensor input and information about the robot's movement, leading to an interconnected problem: For localization, the robot needs a consistent map and to accurately update the map it needs a good estimate of its position in the map. This mutual dependency makes the problem of *simultaneous localization and mapping (SLAM)* complex and requires to search a solution in a high-dimensional space.
Straightforward approaches that localize the robot based on a local, partial map and then update the map based on the maximum-likelihood position tend to produce maps with errors accumulated over time. This is mostly because of the imprecision of the blind reckoning movement information from motor controls and encoders (due to slippage, unevenness of the ground etc.), that are often used as a first positioning hypothesis. To represent the increasing uncertainty one could either blur the current view by the uncertainty, representing a global view of the robot position that becomes ever more uncertain. This leads to sharp observations at the beginning of the mapping process and recent observation that are fuzzy and don't add many useful information. Alternatively, for a robot-based mapping, one could blur the global map by the uncertainty which leads to operations in the past that become more and more uncertain and to recent observations that are sharp.
To construct an accurate map of a large area, one thus needs more advanced approaches to the SLAM problem, often keeping multiple hypotheses of robot position and paths or representing the correlation of errors in the map and positioning errors. Accumulated errors will be especially apparent if a robot re-visits a known position (so called *loop-closure*), a problem that has to be taken into account in SLAM algorithms. Furthermore, despite the complexity of the problem, the maps have to be learned online in real time.
In the following, a brief overview of map representations (section 2.1) and SLAM algorithms (section 2.2) will be given and a popular approach based on so called *Rao-Blackwellized* particle filters will be explained in more detail (section 2.3). Those concepts will then be extended to the case of multiple robots operating together (section 2.4).

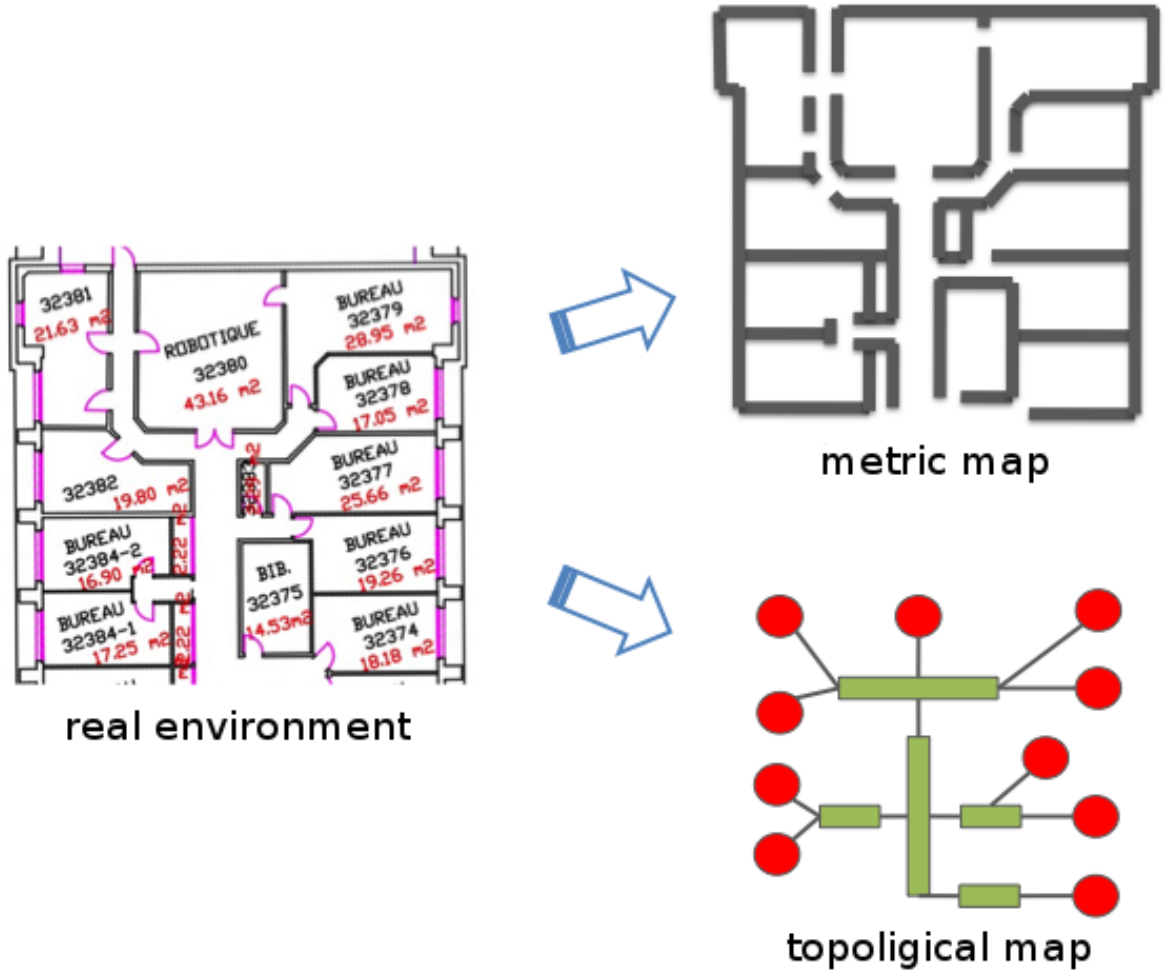



**Figure 2.1:** Example of a map of an office environment. The real environment is represented by its blueprint on the left. The map representations are shown to the right: A metric map on top, showing walls, doorways, corridors and offices by means of occupied or empty grid-cells. A topological map below with a more high-level, graph-like representation of offices and corridors and their connectivity. Figure taken from [15].

## 2.1 Map Representations

The different approaches to the SLAM problem can roughly be characterized by the map representations and estimation techniques used. For the map representations one can differentiate between metric and topological maps [15]:
In a topological map, the environment is represented as a graph-like structure (see fig. 2.1 bottom right). Nodes represent landmarks visible from a certain robot pose, as raw sensor measurements or as a set of extracted features. Edges represent relations between them. Those relations can reach from simple connectivity information to relative positions measurements. This leads to a compact representation but assumes prior knowledge of some structures of the environment (the landmarks as predefined features).

In metric maps, objects are directly represented, often in the form of an occupancy grid (see fig. 2.1 top right). Introduced by Elfes in 1989 [13] this technique divides the space into cells, each with a probability of occupation. The interpretation of the sensor data is done by a stochastic model which relates sensor readings to the true world state. A recursive, Bayesian update of the occupancy grid is calculated at each data acquisition. Often, the robot will operate directly on the probabilities, otherwise, a binary decision (occupied or not) can be taken. This approach is able to represent arbitrary shapes in great detail but will - depending on the resolution of the grid and the size of the map - need a large amount of memory and be computationally expensive.
An extension to a simple occupancy probability are Hidden Markov Models (HMMs) that can be used to assign one of a finite number of states to each grid cell, with an underlying Markov transition model. An example is given in [12] with three possible states: occupied and static (p.ex. for walls and furniture), occupied and dynamic for mobile objects or persons and empty for non-occupied cells. Thus, a semantic, high-level description of the environment, beyond the mere presence or not of an object, can be achieved.

Also, hybrid approaches, combining metric and topological maps may be used: In [8] for example local metric maps of a fixed size are used as vertices of a graph like global map, whose edges will represent the relative position (transformation and assigned uncertainty) of two local maps. This divides a large SLAM problem into several smaller ones. [10] proposes a reinforcement learning algorithm that will (amongst others) decide which representation (grid-based metric map or feature-based topological map) to use during the mapping process. This can be useful for a robot moving indoors and outdoors, the metric representation being generally preferred for indoor maps of rooms and corridors and the topological representation being better suited for outdoor environments.

Such, many different map representations are possible, each one with their advantages and inconveniences. Which one is best suited for a concrete problem will depend on the individual properties of the envisaged application.

## 2.2 Estimation Techniques

Many SLAM algorithms use the Extended Kalman filter (EKF) to solve the SLAM problem. This solution is based on the insight that the covariance matrix maintained by the EKF filter represents the natural correlation between errors in the map and localization errors. Under linear and Gaussian assumptions, the convergence of such an algorithm can be proven [25]. However, the EKF covariance matrices are quadratic in the size of the map (i.e. the number of landmarks $K$) which leads to a quadratic complexity in computational time, as each element of the matrix has to be updated for each new sensor measurement. This is a major obstacle in scaling EKF-SLAM algorithm to larger environments with more than a few hundred features [25]. Also, all landmarks are supposed uniquely identifiable which induces a data association problem

in environments with ambiguous landmarks. Maintaining multiple hypotheses for unknown data association will further increase the computational complexity.

Another approach are algorithms based on *Rao-Blackwellized particle filters (RBPF)* , first introduced by Murphy, Doucet and colleagues [27], [11]. Those take the problem from a Bayesian point of view, as a dynamic Bayes network, and use a particle filter estimation also known as sequential Monte Carlo method [24, p 79ff]. Particle filters are a numerical approximation of arbitrary probability densities by a collection of state vectors (the "particles") and positive importance-weighting factors (see fig. 2.2). They are often en efficient solution to Bayesian estimation problems but a major drawback is that sampling the particles from a high-dimensional state space (as in the SLAM problem) can be inefficient. By the Rao-Blackwellization technique, however, some state variables can be analytically marginalized out, conditioned on the remaining ones, and treated by standard algorithms as the Kalman or HMM filter. Only the remaining state variables will be estimated by the particle filter, such reducing its state space. This

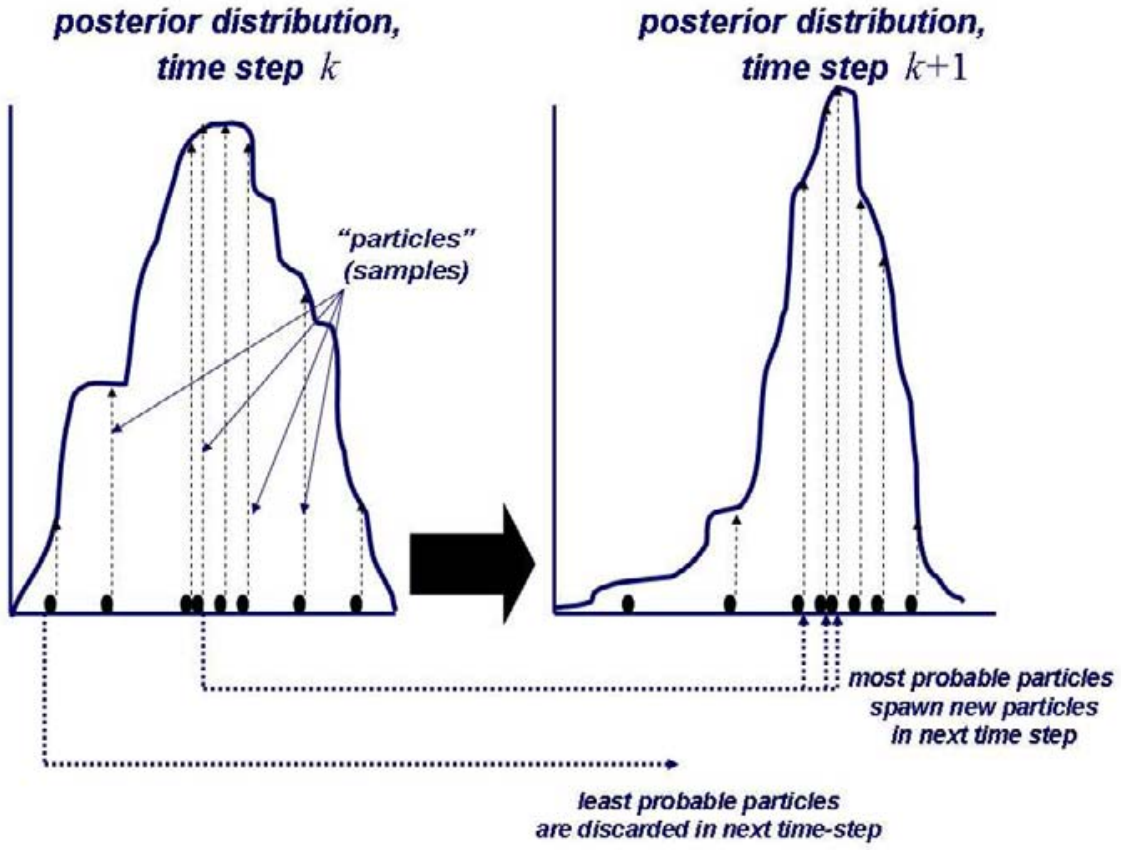



**Figure 2.2:** The basic concept of a particle filter in a one-dimensional state space (x-coordinate): Particles are random samples drawn from the probability distribution. They are more densely located where the probability (y-coordinate) is greatest. The particles are propagated between timesteps in a certain manner to keep this characteristic valid. Figure taken from [24, p. 81].

technique can be applied to the SLAM problem, as it inherently exhibits an important conditional

independence: The knowledge of the robot's path renders the individual landmark measurements independent and such allows for the map construction problem to be marginalized out and computed analytically, conditioned on the robot pose estimate. In a RBPF-SLAM algorithm, a particle filter is therefore used to estimate the posterior over robot paths and each particle possesses its own representation of the environment. Each particle such represents a possible robot trajectory and an associated map.
This approach has proven very efficient to solve the SLAM problem even in large environments and will be further detailed in the following section.

## 2.3 SLAM based on Rao-Blackwellized Particle Filters

### 2.3.1 Rao-Blackwellized Particle Filters

The *Rao-Blackwellization* technique applied to particle filters was introduced by Doucet, Murphy and colleagues [11] [27]. Particle Filters are a sampling-based approximation technique for any type of probability distribution, non-linearity and non-stationarity (see fig. 2.2). The Rao-Blackwellization technique samples only some of the variables of a probability distribution and marginalizes out the remaining ones, using a Kalman filter or other finite dimension optimal filters. The probabilities of the sampled subset are such represented without reference to the variables that have been marginalized-out, reducing the complexity and dimension of the particle filter.
The considered state-space model is that of a dynamic Bayesian network (DBN):
We consider the hidden state variables $\mathbf{s}_t$ and the observations $\mathbf{z}_t$. $\mathbf{s}_t$ is assumed to be a Markov process with the transition model $p(\mathbf{s}_t|\mathbf{s}_{t-1})$ starting from an initial distribution $p(\mathbf{s}_0)$. The observations $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$ are assumed conditionally independent given $\mathbf{s}_t$ of the observation likelihood $p(\mathbf{z}_t|\mathbf{s}_t)$.
The objective is to estimate the joint posterior $p(\mathbf{s}_{1:t}|\mathbf{z}_{1:t})$ calculated by the following recursion:

$$p(\mathbf{s}_{1:t}|\mathbf{z}_{1:t}) = p(\mathbf{s}_{1:t-1}|\mathbf{z}_{1:t-1}) \frac{p(\mathbf{z}_t|\mathbf{s}_t)\, p(\mathbf{s}_t|\mathbf{s}_{t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})}. \tag{2.1}$$

The analytic solution to this recursion contains non-tractable integrals. Therefore a sampling-based numerical approximation is employed. However, sampling in high-dimensional state spaces can be very inefficient. The Rao-Blackwellization approach tries to avoid this problem by using some "tractable substructure" in the model to reduce the dimension of the state space. The idea is to divide the state variables $\mathbf{s}_t$ into two sets $\mathbf{x}_t$ and $\boldsymbol{\theta}_t$, such that:

$$p(\mathbf{s}_t|\mathbf{s}_{t-1}) = p(\boldsymbol{\theta}_t|\mathbf{x}_{t-1:t}, \boldsymbol{\theta}_{t-1})\, p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.2}$$

This makes use of a decomposition of the joint posterior according to the following factorization:

$$\begin{aligned} p\left(\mathbf{s}_{1:t}|\mathbf{z}_{1:t}\right) &= p\left(\mathbf{x}_{1:t}, \boldsymbol{\theta}_{1:t}|\mathbf{z}_{1:t}\right) \\ &= p\left(\boldsymbol{\theta}_{1:t}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}\right) p\left(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}\right). \end{aligned} \tag{2.3}$$

If the posterior distribution of $\boldsymbol{\theta}_{1:t}$ conditioned on $\mathbf{x}_{1:t}$,

$$p\left(\boldsymbol{\theta}_{1:t}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}\right), \tag{2.4}$$

is analytically tractable, then $\boldsymbol{\theta}_{1:t}$ can be marginalized out, and one can focus the particle approximation on $p\left(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}\right)$, which lies in a space of reduced dimension. Since this reduces the complexity of the sampling approximation, one can expect better results.
The implementation of the particle filter is often realized by the *Sampling - Importance - Resampling (SIR)*-algorithm. This involves drawing new particles according to a proposal-distribution that approximates the target distribution as close as possible, calculating importance weights and a resampling method where particles with high importance weights will typically replace those with lower ones. Finally, the marginalized-out variables will be estimated analytically, conditioned on the states of the new particle set. These fore steps are in more detail:

1. *Sampling*: The next generation of particles $\left\{\mathbf{x}_t^{(i)}\right\}$ is obtained from the previous one $\left\{\mathbf{x}_{t-1}^{(i)}\right\}$, according to a proposal distribution $\pi\left(\mathbf{x}_t^{(i)}|\mathbf{z}_{1:t}\right)$. The set contains $N$ particles, indexed with the variable $i$.

2. *Importance Weighting*: An individual importance weight $w^{(i)}$ is assigned to each particle such that:

$$w^{(i)} = \frac{p\left(\mathbf{x}_t^{(i)}|\mathbf{z}_{1:t}\right)}{\pi\left(\mathbf{x}_t^{(i)}|\mathbf{z}_{1:t}\right)}. \tag{2.5}$$

   The weights account for the difference between the proposal and the target distribution. If the two were identical, all weights would be uniform. In practice, the weights will be computed recursively.

3. *Resampling*: The final particles are drawn from the proposed set proportional to their importance weights (see fig. 2.3). This will generally lead to particles with low importance weights being replaced by samples with high weights. It is necessary since the number of particles is finite and proposal and target distribution are different. After resampling, all weights are uniform.

4. *Estimation of the marginalized-out variables*: For each particle $\mathbf{x}^{(i)}$, the corresponding estimate of the marginalized-out variables $\boldsymbol{\theta}_t^{(i)}$ is computed according to $p\left(\boldsymbol{\theta}^{(i)}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}^{(i)}\right)$ conditioned on the particle states $\mathbf{x}_{1:t}^{(i)}$ and on the observations $\mathbf{z}_{1:t}$.
   For the SLAM problem, this step corresponds to the map estimation.

The basic idea of the Rao-Blackwellization approach being explained, some concrete applications of this technique to the SLAM problem will be presented in the following subsection.

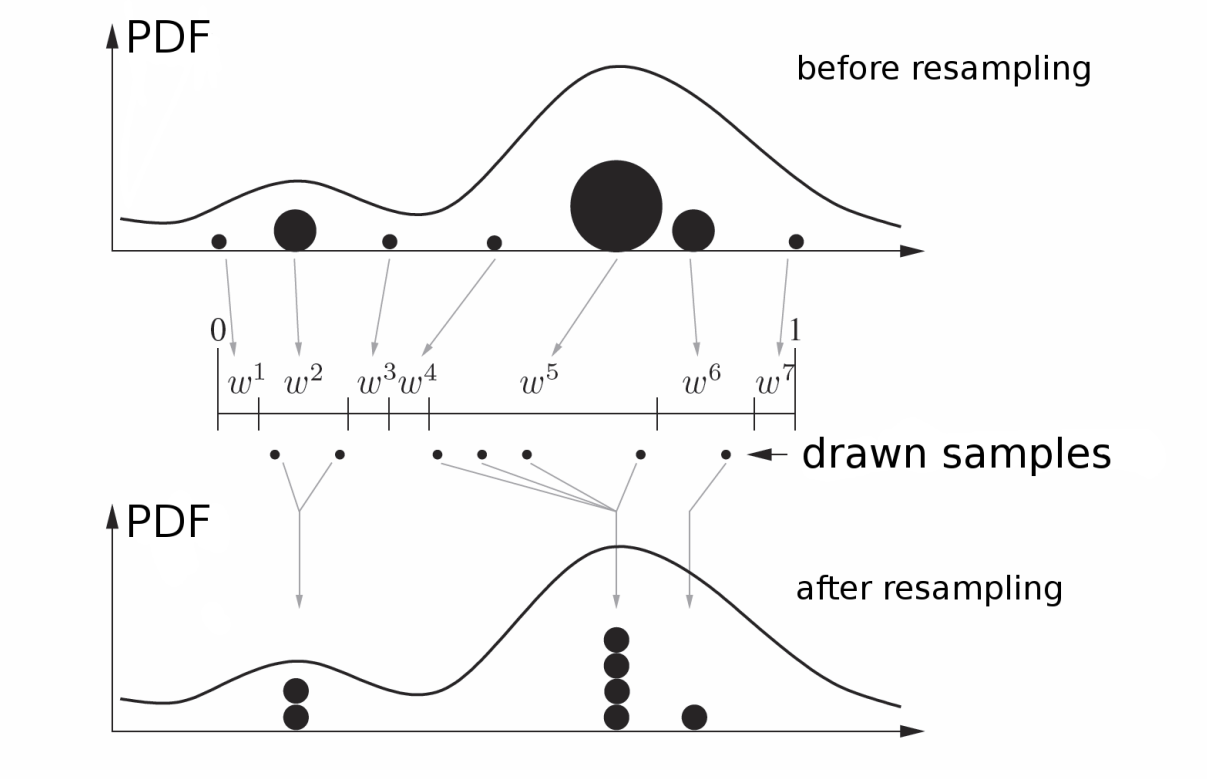



**Figure 2.3:** A simple resampling scheme: On top: particles with diameter proportional to their importance weights, below: particles with equal weights after resampling. Particles with high weights are multiplied, those with low weights are discarded. Figure adapted from [32, p. 183].

### 2.3.2 FastSLAM

One of the first implementation of Rao-Blackwellized particle filters for the SLAM problem is the FastSLAM algorithm proposed by Montemerlo et.al. [26], using a topological, landmark-based map representation.
FastSLAM makes use of the Bayesian approach and the Rao-Blackwellization to factorize the posterior into a product of conditional landmark distributions and a distribution over robot paths. It scales logarithmically with the number of landmarks as opposed to common, extended Kalman filter-based (EKF-based) approaches that require computational time quadratic in the number $K$ of landmarks to process a new sensor observation (see also section 2.2).
For the concrete application to the SLAM problem, the general state model of the previous section 2.3.1 has to be slightly adapted, notably introducing motor commands as a base for the transition model. The sampled state variables are the estimated robot poses, the marginalized-out ones the estimated landmark locations representing the map of the environment. The used state and environment model is (see also fig 2.4):

- *robot poses*: $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$.
- *motor controls and odometry information*: $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_t\}$.

- *observations*: $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$.
- *environment representation*: the positions of $K$ immobile landmarks $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_K\}$.
- *correspondence index*: $n_t \in \{1, ..., K\}$, the index of the landmark perceived at time $t$. It is assumed that only one landmark will be perceived at a time. When the robot senses more than one landmark at a time, the multiple sightings are processed sequentially.
- *motion model*: $p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$.
- *measurement model*: $p(\mathbf{z}_t|\mathbf{x}_t, \boldsymbol{\theta}, n_t)$.

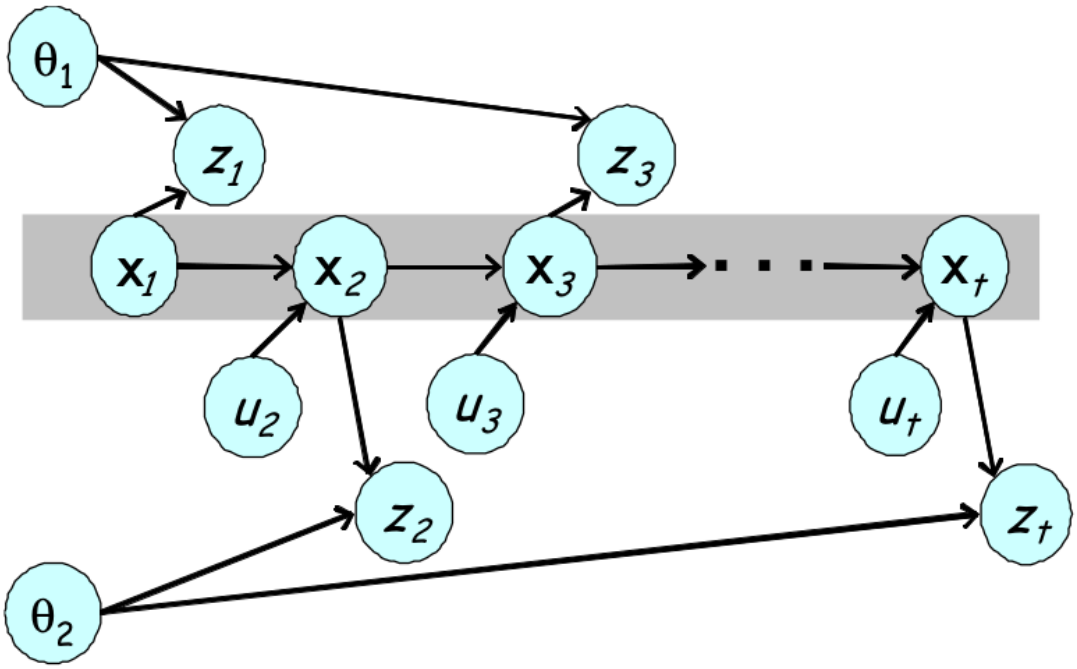



**Figure 2.4:** The SLAM-problem: The robot moves along a path from an initial position $\mathbf{x}_1$ to the current position $\mathbf{x}_t$ by a sequence of motor controls $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_t\}$. While moving, it observes the environment represented by the landmarks $\boldsymbol{\theta}$, the measurements being denoted $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$. The goal of a SLAM algorithm is to estimate the landmark positions and the robot's path from the control information $\mathbf{u}$ and the observations $\mathbf{z}$. Figure taken from [26].

The knowledge of the robot's path $\mathbf{x}_{1:t}$ (shaded grey in fig. 2.4) renders the individual landmark measurements independent, such making possible the factorization used for the Rao-Blackwellization approach. Thereby, the SLAM problem is decomposed into a robot localization problem and a collection of landmark estimation problems that are conditioned on the pose estimate.
A particle Filter (with $N$ particles used for the approximation) is used for the path estimation and each particle disposes of $K$ Kalman filters that estimate the 2-dimensional location of the respective landmark $\boldsymbol{\theta}_k$. A naive implementation would therefore require $O(NK)$ computational time, but an efficient, tree based data structure reduces the running time to $O(N \log K)$.

The SLAM problem consists of determining the locations of all landmarks $\boldsymbol{\theta}$ and robot poses $\mathbf{x_t}$ from measurements $\mathbf{z}_{1:t}$ and motor controls resp. odometry information $\mathbf{u}_{1:t}$. With known correspondences, the posterior can be factorized as [26]:

$$p\left(\mathbf{x}_{1:t}, \boldsymbol{\theta} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, n_{1:t}\right) = p\left(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, n_{1:t}\right) \prod_k p\left(\boldsymbol{\theta}_k | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, n_{1:t}\right). \tag{2.6}$$

The problem is such decomposed into one problem of estimation a posterior over robot paths and $K$ problems of estimating the landmark locations. The path estimator is implemented as a particle filter and the landmark pose estimators are implemented as Kalman filters using a two-dimensional state space (the two coordinates of each landmark).
The set of particles maintained in the particle filter represents the posterior probability density function of the robot paths $p\left(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, n_{1:t}\right)$.
Each particle $\mathbf{s}^{(i)}$ with index $i$ more precisely represents a "guess" of the robot's path $\mathbf{x}_{1:t}^{(i)}$ as well as the mean and covariances of the landmark positions $\boldsymbol{\mu}_{1:K}^{(i)}, \boldsymbol{\Sigma}_{1:K}^{(i)}$, conditioned on this robot pose estimate.
The particle set $\mathbf{S}_t$ is calculated recursively from the previous particle Set $\mathbf{S}_{t-1}$, odometry information $\mathbf{u}_t$ and an observation $\mathbf{z}_t$.
First, in the sampling step, new particles are generated according to a *proposal distribution* that approximates the target distribution. In the case of FastSLAM, alone the motion model $p\left(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1}^{(i)}\right)$ is used, such only taking into account the odometry information and not the current measurement.
As the proposal distribution is different from the target distribution, a resampling scheme is needed. For this purpose, so called *importance weights* $w^{(i)}$ are calculated for each particle according to equation (2.5). With the proposal distribution $\pi$ being the motion model, importance weights can be computed according to the observation model [26]:

$$w_t^{(i)} \propto \int p\left(\mathbf{z}_t | \mathbf{x}_t^{(i)}, \boldsymbol{\theta}_{n_t}^{(i)}, n_t\right) p\left(\boldsymbol{\theta}_{n_t}^{(i)}\right) \, \mathrm{d}\boldsymbol{\theta}_{n_t}. \tag{2.7}$$

Since the landmarks are represented by a Gaussian distribution, the integral can be evaluated in closed form.
In the resampling step, particles with high importance weights will then in general replace those with lower weights to better approximate the target distribution (see also fig. 2.3). However, this can also lead to useful particles being deleted, creating the problem of particle depletion, notably for the closing of loops.
Finally, the position of the observed landmark $\boldsymbol{\theta}_{n_t}$ is estimated by an EKF-update process, assuming Gaussian approximations of the measurement model and the landmark posterior:

$$p\left(\boldsymbol{\theta}_{n_t} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, n_{1:t}\right) \propto p\left(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\theta}_{n_t}, n_t\right) p\left(\boldsymbol{\theta}_{n_t} | \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}, n_{1:t-1}\right). \tag{2.8}$$

The positions of non-observed landmarks are kept unchanged.

The algorithm described this far may require linear time in the number of landmarks $K$: Every time a particle $\mathbf{s}_t^{(i)}$ with an updated pose estimate is generated from the corresponding previous

particle $s_{t-1}^{(i)}$, all $K$ landmark mean and covariance information have to be copied. However, generally only one landmark position is updated at a time (as it is assumed that only one landmark is perceived at a time). A binary tree structure of the landmark parameters attached to a particle allows for a more efficient implementation: Only a single path in the tree has to be modified at a time, all other pointers can be kept from the generating particle (fig. 2.5). This leads to logarithmic complexity in the number of landmarks $K$.

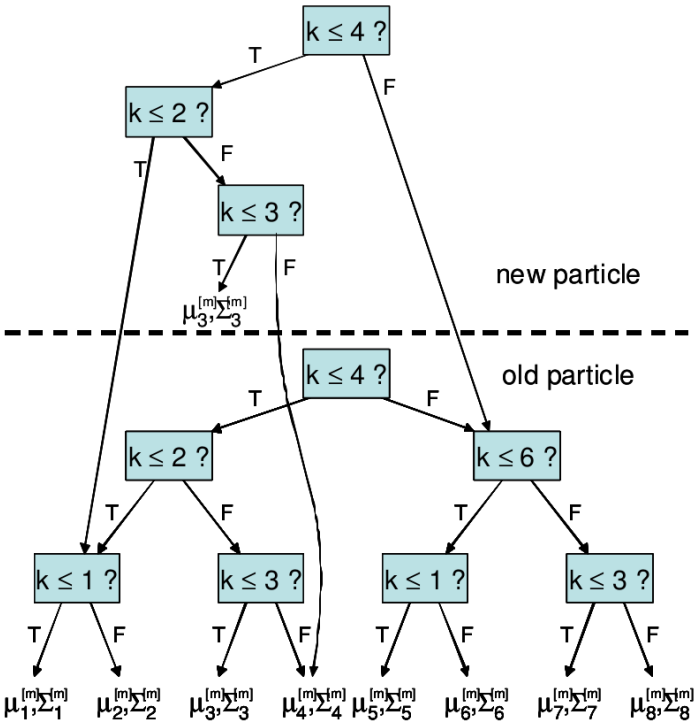



**Figure 2.5:** A tree representing $K = 8$ landmarks within a single particle and its update after generating a new particle where the landmark with index $k = 3$ has been modified: Only a single Gaussian is changed, the new particle such receives only a partial tree containing the path to the modified parameters. All other pointers are copied from the old tree. Figure taken from [26].

Another problem of the presented algorithm is the data-association, i.e. the correspondence between observation and known landmarks. This is generally solved by a maximum likelihood estimator:

$$n_t^{(i)} = \underset{n_t}{\operatorname{argmax}}\, p\left(\mathbf{z}_t | \mathbf{x}_t^{(i)}, n_t\right). \tag{2.9}$$

Each particle calculates its own data association, such making the simultaneous pursuit of multiple hypotheses possible. This decreases the problem of false associations, as those are likely to disappear in the resampling process.

The greatest default of the FastSLAM algorithm, however, is the proposal distribution taking into account only the motion model and not the current observation, which is in general significantly

more precise than the dead-reckoning odometry information. This will lead to a large number of particles with small importance-weights after the weighting with the observation model. Such, a large number particles is needed to correctly represent the target distribution, as many of them will be replaced in the resampling step.

To overcome this default, Montemerlo et. al. introduced an improved version of the algorithm, *FastSLAM 2.0* [25], notably taking into account not only the odometry information but also the most recent sensor measurement for the proposal distribution according to which new particles are drawn:

$$\mathbf{x}_t^{(i)} \sim p\left(\mathbf{x}_t | \mathbf{x}_{1:t-1}^{(i)}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, n_{1:t}\right). \tag{2.10}$$

This modified proposal distribution such approximates the target distribution considerably better than the previous approach, drastically reducing the number of particles needed for an accurate map (often, even a single particle is sufficient). It will be computed using a Gaussian approximation of the observation model. This also leads to a proof of convergence of the filter. Furthermore, a dynamic approach for feature management is proposed: New features are created when the measurement probability is below a threshold, but features can also be deleted, when they are not observed for a longer time. The probability of landmark existence is calculated with a recursion known from occupancy-grid algorithms. Such, particles are able to free themselves of spurious features.

The FastSLAM algorithms, notably the latter version, present an efficient an robust solution to the mapping and localization problem. The algorithm was validated by the authors in real-world and simulated experiments [26] [25], for the improved version FastSLAM 2.0, even a mathematical convergence proof was given [25].
The FastSLAM algorithms use a landmark-based, topological map representation. For the widespread Laser Range finders (LIDAR-sensors) grid based maps are often more appropriate. In the following subsection, some grid-based RBPF-SLAM techniques will be presented.

### 2.3.3 Grid-based RBPF SLAM

To model the SLAM-problem with grid based maps we will keep the same state variables as described in the beginning of the previous section 2.3.2, but replace the set of landmarks $\boldsymbol{\theta}$ with an occupancy grid map $\mathbf{m}$.

A common approach for grid-mapping with laser range finders is incremental scan matching [18]: From a previous pose estimate $\hat{\mathbf{x}}_{t-1}$ and map $\hat{\mathbf{m}}_{t-1}$ as well as a new measurement $\mathbf{z}_t$, a most likely new pose $\hat{\mathbf{x}}_t$ is determined by trading off the consistency of the measurement with the map and the consistency of the new pose with the control action (odometry data) and the previous pose. The map is then extended by the measurement $\mathbf{z}_t$ using the pose estimate $\hat{\mathbf{x}}_t$. This approach can be implemented efficiently on a global map and is accurate on a local scale, but its

disadvantage is its greedy optimization that takes into account only the next time-step and keeps only one hypothesis of the robot's path. This can lead to significant errors during loop closing. A possible solution is to delay the maximization of the likelihood function until a loop closure is detected and then to correct backwards the robot path. However, this approach under-estimates the uncertainty when closing loops [18].

Hähnel et. al. present an algorithm that combines a Rao-Blackwellized Particle filter (RBPF) with scan matching in [18]. They propose an adaptation of the previously presented FastSLAM algorithm to laser scan data and a grid-based map representation (each particle carries its own occupancy grid map $\mathbf{m}$). Scan matching is used to minimize odometric errors. This decreases the number of particles required and reduces the particle depletion problem.
The particle depletion problem arises from the fact that while a robot traverses a large cycle it accumulates a considerable positioning error due mostly to imprecision in odometry data. A RBPF is an efficient method to represent alternative hypotheses on the robot path. However, resampling may lead to particles that upon loop closure would represent a correct path being deleted because they have low importance weights before the closure-point is reached.
This problem is partially solved by the more accurate proposal density due to laser-corrected

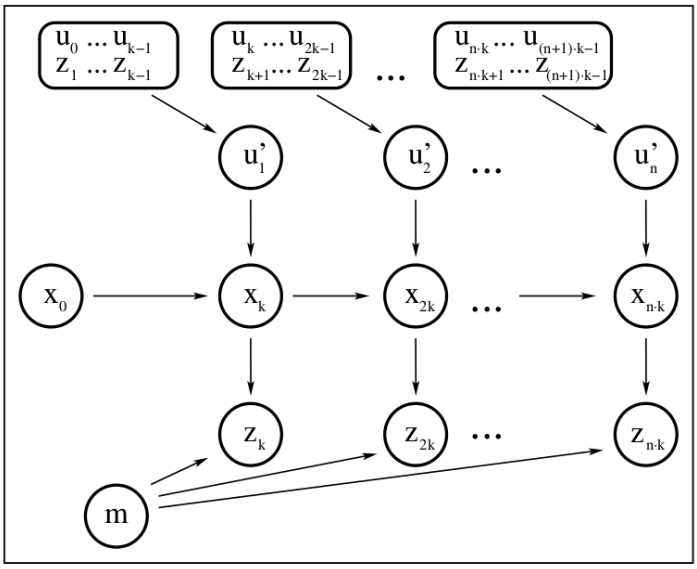



**Figure 2.6:** Schema of the integration of scan-matching into the mapping process: each $k$-th laser scan is used for the mapping process, the remaining $k-1$ scans are used to correct the odometry data $\mathbf{u}$. The laser-corrected odometry measurement $\mathbf{u}'$ is then used for the proposal distribution of new particles. Figure taken from [18].

odometry data. In [18] sequences of laser scans are transformed into improved odometry measurements using scan-matching techniques: The scan matching is calculated using the "beam-endpoint model": the likelihood of a beam is calculated based on the distance between the endpoint of the beam and the closest obstacle to that point in the occupancy grid map. To represent the uncertainty in scan-matching, a parametric model is used of which the parameters are learned on experimental data. This algorithm will give a most likely estimate (with associated

uncertainty) of the new robot pose that can be incorporated in the movement model used in the particle filter. Those improved odometry measurements are then processed in the sampling step. As in the first FastSLAM proposal [26], new particles are drawn only according to the movement model, but implicitly take into account the range sensor observations by processing the laser-corrected odometry data (see fig. 2.6).
Concretely, every $k$ steps, a new corrected odometry measurement $\mathbf{u}'$ is computed out of the $k-1$ previous laser scans $\mathbf{z}$ and the $k$ most recent odometry readings $\mathbf{u}$. the $k$-th laser scan is then used to compute the importance weights of the samples. This ensures that all information is only used once. During the map update step, only the cells visible according to the current position of the corresponding particle are modified.
This algorithm produces accurate maps with around 100 particles.

Grisetti and colleagues proposed an improved algorithm for RBPF-SLAM with laser range measurements and occupancy grid maps. The algorithm presented in [16] and [17] is implemented in the ROS *gmapping* package and will be used in the practical part of this research project.
Different from the previously presented approach [18], a more accurate proposal distribution is computed, taking into account not only the movement but the most recent observation. This is similar to the FastSLAM 2.0 algorithm [25], but adapted to the occupancy grid map representation and laser range measurements. The proposal distribution is computed by evaluating the likelihood around a particle-dependent pose obtained by a scan-matching procedure combined with odometry information (see fig. 2.7). Such, the most recent sensor observation is also considered for creating the next generation of particles. This makes the map more accurate and reduces the estimation error, so that fewer particles are needed.
Furthermore, a selective resampling technique is presented to reduce the risk of particle depletion. It allows to perform the resampling step only when needed.

The algorithm makes use of the Rao-Blackwellization technique by the following factorization, with the notations as in section 2.3.2 but with the environment represented by the metric grid-map $\mathbf{m}$ and not by discrete landmarks:

$$p\left(\mathbf{x}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}\right) = p\left(\mathbf{m} | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}\right) p\left(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}\right). \tag{2.11}$$

This allows to first estimate only the trajectory $p\left(\mathbf{x}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}\right)$ by a particle filter approximation and then to compute the map based on that trajectory. The posterior over maps $p\left(\mathbf{m} | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}\right)$ can be computed analytically since $\mathbf{z}_{1:t}$ and $\mathbf{x}_{1:t}$ are supposed known. This makes an efficient computation possible. In the particle filter, an individual map is associated to each sample. The common sampling-importance-resampling (SIR) approach as explained in section 2.3.1 is used. After sampling a new generation of particles, the calculation of the importance factors, and eventually a resampling step, the corresponding map estimate $p\left(\mathbf{m}^{(i)} | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}^{(i)}\right)$ is calculated for each particle $s^{(i)}$, based on the particles trajectory $\mathbf{x}_{1:t}^{(i)}$ and on the observations $\mathbf{z}_{1:t}$.

The key contribution of [16] and [17] is the improved proposal distribution in the sampling step: Typical particle filters use only the odometry model as a proposal density. This is easy to compute but imprecise. Such, only a fraction of the generated samples will have a high likelihood

under the observation model (and such a high importance factor) and a large number of samples is necessary for a sufficient approximation (see fig. 2.7).
For this reason, the improved proposal distribution takes into account the most recent sensor observation when generating new particles. By this means, one can focus the sampling on the meaningful regions of the observation likelihood.

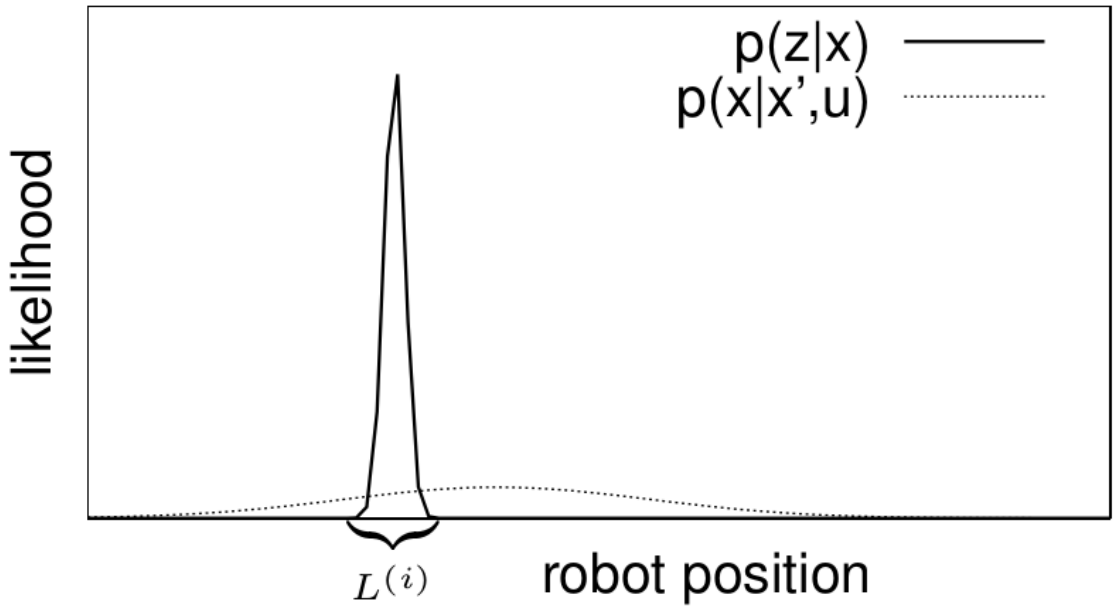



**Figure 2.7:** The two components of the utilized motion model: odometry information (pointed line) and observation likelihood (solid line). Within the meaningful interval $L^{(i)}$ the product is dominated by the observation likelihood. Figure taken from [16].

The used proposal distribution $\pi$ is optimal with respect to the variance of the importance weights [17] and can be written as:

$$p\left(\mathbf{x}_t|\mathbf{m}_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, \mathbf{z}_t, \mathbf{u}_{t-1}\right) = \frac{p\left(\mathbf{z}_t|\mathbf{m}_{t-1}^{(i)}, \mathbf{x}_t\right) p\left(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}\right)}{p\left(\mathbf{z}_t|\mathbf{m}_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}\right)}. \tag{2.12}$$

Grisetti et. al furthermore describe an efficient computation method of the proposal distribution based on a Gaussian approximation. This uses the hypothesis that the scan-likelihood will have a single distinct peak. It will compute a sampled approximation of the optimal proposal given in (2.12): For each particle, a set of $K$ potential poses $\left\{\mathbf{x}_j\right\}$ is created in a region $L^{(i)}$ surrounding the (mostly peaked) maximum of the observation likelihood (see fig. 2.7). This ignores the less meaningful regions of the distribution and such saves computational resources. The meaningful area $L^{(i)}$ is computed by a scan-matching procedure:

$$L^{(i)} = \left\{\mathbf{x}|p\left(\mathbf{z}_t|\mathbf{m}_{t-1}^{(i)}, \mathbf{x}\right) > \epsilon\right\}. \tag{2.13}$$

As the proposal distribution is supposed Gaussian, its parameters are then computed for each particle from the samples $\{\mathbf{x}_j\}$ using the observation and the odometry model:

$$\boldsymbol{\mu}_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} \mathbf{x}_j \cdot p\left(\mathbf{z}_t | \mathbf{m}_{t-1}^{(i)}, \mathbf{x}_j\right) p\left(\mathbf{x}_j | \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}\right), \tag{2.14}$$

$$\boldsymbol{\Sigma}_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^{K} p\left(\mathbf{z}_t | \mathbf{m}_{t-1}^{(i)}, \mathbf{x}_j\right) p\left(\mathbf{x}_j | \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}\right) \left(\mathbf{x}_j - \boldsymbol{\mu}_t^{(i)}\right) \left(\mathbf{x}_j - \boldsymbol{\mu}_t^{(i)}\right)^{\mathrm{T}}, \tag{2.15}$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^{K} p\left(\mathbf{z}_t | \mathbf{m}_{t-1}^{(i)}, \mathbf{x}_j\right) p\left(\mathbf{x}_j | \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}\right). \tag{2.16}$$

The importance weights are computed recursively by multiplying with the normalization factor $\eta^{(i)}$:

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot \eta^{(i)}. \tag{2.17}$$

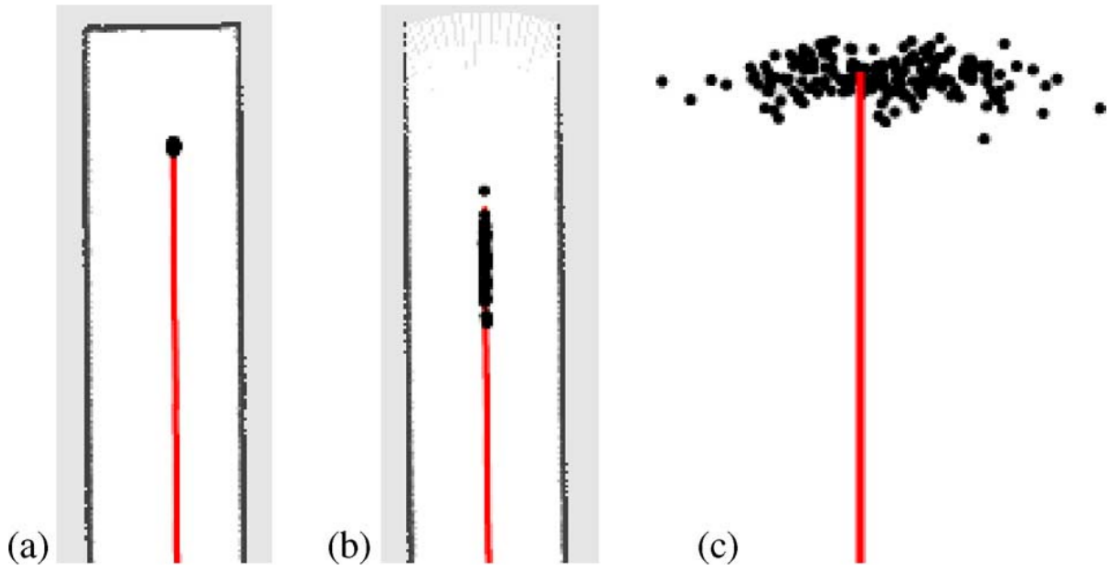



**Figure 2.8:** Some particle distributions typically observed: In a dead and corridor, the uncertainty is small (a), in an open corridor particles are distributed along the central axis of the corridor (b). These distributions have been obtained while taking into account the observation during the sampling step. In (c), the scan-matching failed as there are no distinguishable features. The particles are much more widely spread when only the raw odometry model is used. Figure taken from [16].

The presented proposal distribution focuses the sampling on the important regions (see fig. 2.8). A scan matching algorithm is used to calculate the observation likelihood used to concentrate

the sampling. The scan matching is done by a "beam endpoint model" as in [18], described at the beginning of this subsection. Multi-modal likelihood function can pose a problem, but this is negligible when doing frequent map updates. If scan matching fails, (p.ex. in an empty, featureless space), only raw odometry data is used.
In any case, the odometry model has to be evaluated point-wise at the samples $\mathbf{x}_j$ for equations (2.14) to (2.16). To this end, a Gaussian approximation of the odometry motion model obtained by an extended Kalman filter-like Taylor expansion is used. Such, the full probability density of the movement model is evaluated. This will, in a few situations when only poor features are available for scan matching (p.ex. a long, featureless corridor), better focus the proposal distribution as opposed to a previous version [16] where the movement model was approximated by a constant over the region $L^{(i)}$ of the samples $\mathbf{x}_j$ (see fig. 2.9).

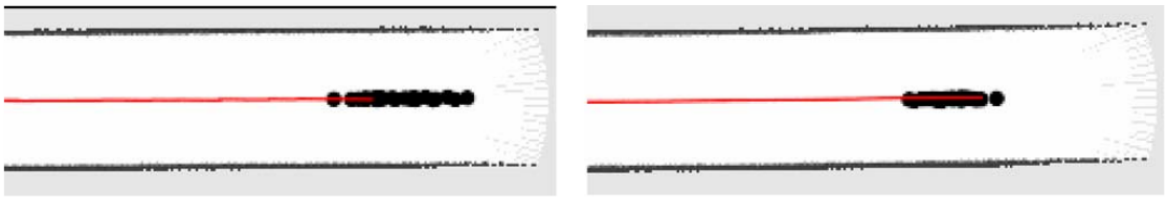


**Figure 2.9:** Influence of considering the odometry model for the proposal distribution. In the left picture, it is approximated as constant, only the laser data is used. In the right picture both sources of information are taken into account. Such, the particles can be drawn in a more accurate manner, as there are not many features for scan-matching in an empty corridor. Figure taken from [17].

A second proposal of [16] and [17] is an adaptive resampling algorithm:
During resampling, particles with low importance weights $w^{(i)}$ are typically replaced by those with higher weights. This is necessary, since target and proposal distribution differ and only a finite number of particles is used, but on the other hand, the resampling step can remove correct samples, leading to particle depletion. For this reason, it is desirable to only perform this step when necessary. To decide when to perform a resempling step, the effective sample size is used.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}^{(i)}\right)^2} \tag{2.18}$$

where $\tilde{w}^{(i)}$ is the normalized weight of particle $s^{(i)}$.
If the samples are close to the target distribution, their weights are nearly equal. If the samples differ from the target distribution, the variance of their weights increases and $N_{eff}$ decreases. A resampling process is started at each time $N_{eff}$ falls below $N/2$ (with $N$ the number of particles used in the filter). This drastically reduces the risk of replacing useful particles, as the resampling is only performed when needed and the overall number of such operations is reduced.

To resume, the overall algorithm can be described by the following steps that will be executed

each time a new measurement tuple of odometry data and laser-observation $(\mathbf{u}_{t-1}, \mathbf{z}_t)$ is available:

1. Initial guess $\mathbf{x'}_t^{(i)}$ of the new robot pose for each particle $s_{t-1}^{(i)}$ from odometry measurements and the previous position of the respective sample.
2. Scan-matching based on the map $\mathbf{m}_{t-1}^{(i)}$ of the particle starting from the initial guess $\mathbf{x'}_t^{(i)}$ to get the most likely new pose $\hat{\mathbf{x}}_t^{(i)}$ and the observation likelihood.
3. Selection of the samples $\{\mathbf{x}_j\}$ around the pose $\hat{\mathbf{x}}_t^{(i)}$ and calculation of mean $\boldsymbol{\mu}_t^{(i)}$ and covariance $\boldsymbol{\Sigma}_t^{(i)}$ of the proposal by equations (2.14) to (2.16).
4. Sampling of the new pose $\mathbf{x}_t^{(i)}$ for each particle $s_t^{(i)}$ from the Gaussian approximation of the proposal distribution $\mathcal{N}\left(\boldsymbol{\mu}_t^{(i)}, \boldsymbol{\Sigma}_t^{(i)}\right)$.
5. Update of the importance weights.
6. Calculation of the map update $\mathbf{m}_t^{(i)}$ for each particle $s_t^{(i)}$ based on the drawn pose $\mathbf{x}_t^{(i)}$ and the observation $\mathbf{z}_t$.
7. Eventual resampling-step depending on $N_{eff}$.

The computational complexity of the algorithm depends mainly on the number of particles $N$. Only resampling is of complexity $O(NM)$ with $M$ a measure of map size, as each particle stores its own grid map, which has to be copied on resampling. More efficient map representations, p. ex. as proposed in [14] based on keeping an ancestry-tree where each particle stores only the grid squares it has updated, can be employed, but as resampling is infrequent, this doesn't pose an important problem and a simple map representation is maintained in the here presented algorithm.

This improved approach to RBPF grid mapping computes accurate maps from 15 to 80 particles according to the authors, depending on the complexity of the environment [17]. This is about an order of magnitude less than in [18]. The algorithm uses an accurate proposal distribution, focusing the particles on the meaningful region of the observation, by taking into account odometry data and range-scan observations in the sampling process of the particle filter. Furthermore, an adaptive resampling strategy avoids unnecessary resampling steps and reduces the risk of particle depletion. This algorithm, as implemented in the ROS *gmapping*-package, will be maintained for the practical part of this research project. Figure 2.10 shows an example of a map of the Freiburg Campus constructed with the presented algorithm.

**Figure 2.10:** Example of a map of the Freiburg University campus constructed with the SLAM-algorithm proposed in [17]. According to the authors, the robot traverses an area of approx. 250m x 250m on a trajectory of a length of 1.75km. 30 particles were used. Figure taken from [17].

## 2.4 Multi-Robot SLAM using Occupancy Grid Maps

When mapping large areas, or for a cooperative robot task, it is often necessary for two or more robots to participate in the mapping process. This is known as multi-robot SLAM or cooperative SLAM and will aim to increase the efficiency and accuracy of the mapping procedure. However, the complexity of the SLAM-problem is significantly increased when several robots cooperate to construct a joint map, especially when the coordinate transform between the initial positions of the robots is not known - as is often the case in practical problems. To approach the problem, several choices have to be made: whether the map representation should be a metric occupancy-grid or a topological graph, based on landmarks and also whether the robots will continuously construct a joint map, if local maps will be merged at discrete points in time or if the joint map will be constructed entirely offline. This implies furthermore the question which amount of data will have to be communicated between the robots: Do they need to share all their measurements, only parts of them and do they need to share the data continuously or at discrete moments?

In the following, two different concepts will be presented, both based on occupancy grid maps, as it is the map representation retained for this research project. Also, we will assume unknown initial relative poses between the robots throughout this section.
A first approach is to construct a common map by sharing range-scan and odometry data between

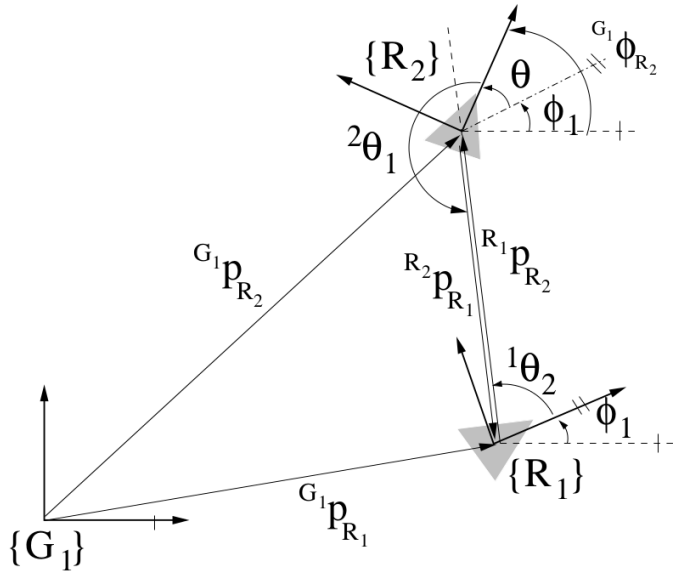



**Figure 2.11:** Schema depicting the relative pose measurements of two robots $R_1$ and $R_2$. Range and bearing measurements are taken and the two robot positions are finally calculated with respect to the global frame $G_1$ of robot 1. Figure taken from [33].

the robots during the exploration process (section 2.4.1). Those approaches are mostly based on relative pose measurements on mutual encounters of the robots (see fig. 2.11). Those measurements will give an initial estimate of the transformation between the robot poses and will often be used as a starting point for map-merging. With an estimate of the relative positions, the shared range-scan and odometry data can then be combined into a mutual global map. After a first encounter, the relative positions are known and the joint map can be augmented continuously with the data of all robots and is available during the rest of the exploration task.

Another approach is to perform local mono-robot SLAM on the individual robots and to merge the occupancy grid maps by finding overlapping regions in the maps (section 2.4.2). Such, no mutual encounters and (often imprecise) relative position measurements of robots are required. The robots still need to visit in parts the same areas of the environment, to create overlap between the individual maps, but they need not be there at the same time. Also, not the entire raw range-scan and odometry measurements need to be shared, but only the already preprocessed local maps, such reducing the mobile communication throughput necessary. The map fusion could take place entirely offline, after completion of the exploration task, as in [3], however, a joint map will not be available during the exploration process but only after its completion. For this reason, the focus of section 2.4.2 will be on online map-merging techniques, continuously sharing and merging local occupancy grid maps between robots.

### 2.4.1 Multi-Robot RBPF-SLAM based on Mutual Encounters of Robots

To begin with, some approaches for multi-robot SLAM relying on mutual encounters and relative position measurements will be outlined. The presented algorithms use the Rao-Blackwellized particle filter introduced in section 2.3.
Carlone et. al. propose an algorithm for multi-robot SLAM with limited communication and unknown initial relative position using a RBPF-filter [6]. Odometry and perceptive data of each team-mate are fused, taking into account the uncertainty of a relative pose measurement. A distributed approach to the multi-robot mapping problem is employed: The robots build their own word representation using local information and the data of other robots. The computation remains local, while the output should be shared.
The robots are each equipped with a laser scanner, a pan-tilt camera and an odometric pose estimation. They can communicate within a given maximal distance. A grid-based, single-robot RBPF-SLAM technique (as in section 2.3.3) is employed before and after each encounter between two robots. On rendezvous an information fusion procedure is started:

- *Data exchange*: Robot $A$ receives the data from the met team-mate acquired from the last meeting to the rendezvous instant. Only preprocessed data (laser stabilized odometry and laser scanner measurements) are transferred.
- *Reference frame transformation*: From the relative pose measurement taken during the rendezvous, the data received is roto-translated in robot $A$'s reference frame. The relative pose measurement is done by the pan-tilt camera and contains the associated uncertainty.
- *Estimation on virtual data*: The transformed data is passed to the sensor buffer and processed as if it were robot $A$'s own measurements ("virtual measurements"). The processing is done in the reverse direction, from the rendezvous location to the team-mate's starting position (respectively the previous rendezvous position).

After completion of the filtering process, the robot resumes its RBPF-SLAM mapping process with the filter particles from before the meeting. After the first encounter, the robots share a very similar but not exactly equal (this is due to the probabilistic processing) representation of their environment, up to a known roto-translation. As the transformation between the reference frames of the two robots is now known, future encounters can be planned to improve map consistency.
Figure 2.12 illustrates an example of the proposed algorithm: Two robots start from positions $S_1$ and $S_2$. They meet at positions $F_1$, resp. $F_2$ where they will exchange odometry and laser observation data. Robot 1 will then process the data received from robot 2 backwards from $F_2$ to $S_2$ to include it in its own map. It will continue the mapping from pose $F_1$ after the merging has completed. Robot 2 will proceed in a similar manner.
This algorithm shares data at discrete moments in time - on mutual encounters of robots - and therefore can deal with limited communication range and bandwidth. The mapping process can be interrupted for a short time during merging.

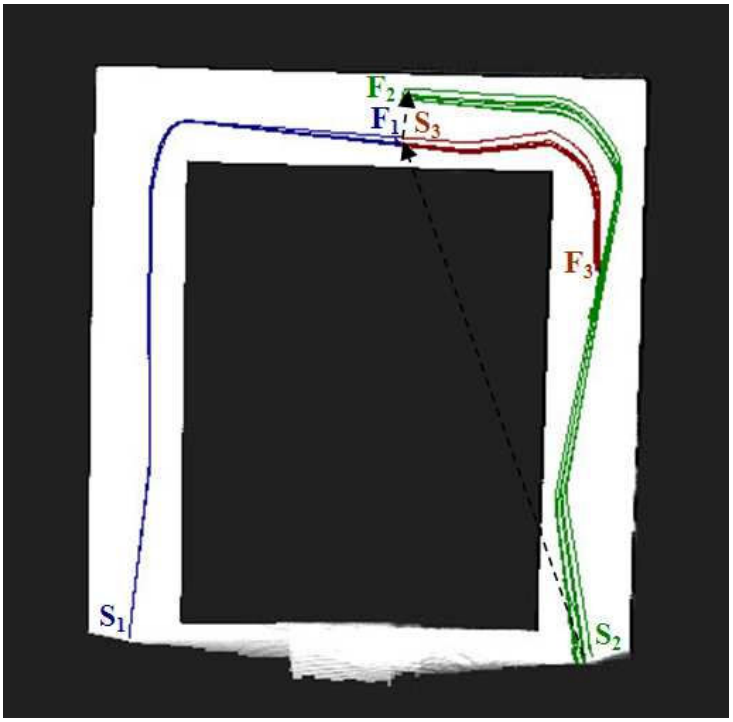



**Figure 2.12:** Illustration of the multi-robot RBPF-mapping procedure. The lines depicted correspond to the robot path proposals kept by the filter particles. Figure taken from [6].

A similar approach is presented by Howard in [19] also using RBPF particle filters for the underlying mapping process, but working in a continuous, online manner: Each robot broadcasts its observations and odometry data and each robot saves those measurements for all robots cooperating in the team. This demands for a stable, high-bandwidth WiFi connection. The initial position being unknown, data from other robots is only saved for later use in a first place. If an encounter between two robots occurs, a relative pose measurement will be taken and the particle filter will be augmented by two new instances: an acausal one, processing the old measurements backwards like in the previously presented approach [6] and a causal one processing the subsequent measurements of the encountered robot. All instances will process the data with the same frequency, the acausal one gradually emptying the previously registered measurements and the causal ones processing the robots own as well as the other robots new data in parallel. Figure 2.13 gives an overview of the procedure: Initially, robot 1 will only register the data from the team-mates, as it doesn't know their relative positions it can't process it. At time $t = a$ robots 1 and 2 meet and take a relative pose measurement. Robot 1 starts two new filter instances, processing robot 2's old data backwards (blue shading) and the newly arriving data forwards in parallel (red shading). A recorded encounter between robot 2 and 3 then permits robot 1 to also incorporate the third team-mate's data. The causal instance for robot 3 will however keep a constant delay to robots 1 and 2 as all measurement queues are processed with the same update frequency.
This algorithm permits a continuous joint map construction. The global map is updated consecutively with the data from all robots whose relative positions are known after a mutual encounter.

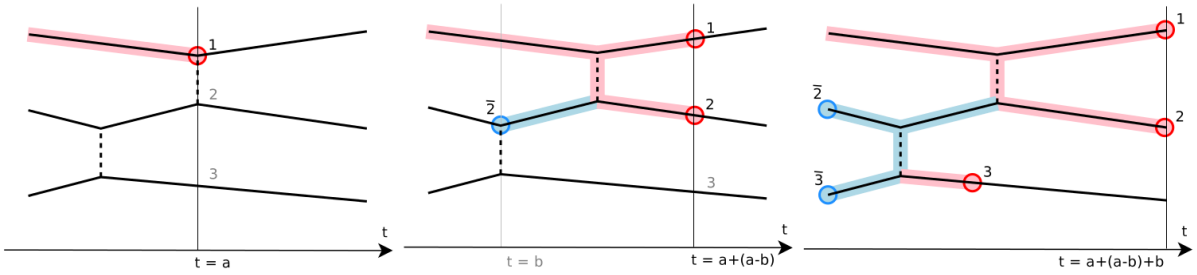



**Figure 2.13:** Encounter diagram for robot 1 of a three-robot experiment: solid lines: data sequences recorded by each robot, pointed lines: mutual observations. Figure taken from [19].

The update is processed continuously and not only at discreet encounters as in the approach from [6]. However, it needs a stable, high-bandwidth WiFi communication to share all measurements of all robots.

### 2.4.2 Multi-Robot SLAM by Merging Occupancy Grid Maps

Secondly, some approaches to multi-robot SLAM by merging local occupancy grid maps into a global one will be presented. These techniques don't rely on mutual encounters or relative position measurements but on finding and aligning overlapping regions between the local maps. Formally, the problem consist of finding a rotation and a translation between two occupancy grid maps $m_1$ and $m_2$ such that the overlap between $m_1$ and the transformed $m_2$, $m_2'$ is maximized and the alignment is coherent. The roto-translation needed to align $m_2$ with $m_1$ is described by the three parameters $\theta$, $t_x$, $t_y$ in the following matrix form:

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.19}$$

Several methods from literature for finding the roto-translation between two local maps will be presented in the following. Figure 2.14 shows an example to illustrate the general idea of occupancy grid map-merging.
Firstly, some direct-optimization methods, searching to optimize an heuristic function defining the quality of the alignment in terms of overlap and coherence, will be presented. In later paragraphs, some other approaches based on extracting and matching Hough-features or features like SIFT, SURF or the Canny edge descriptor known from computer vision, will be examined.

In [4] Birk and Carpin describe occupancy grid merging as an optimization problem using a random search (in their case the Adapted Random Walk algorithm) of the space of possible roto-translations. The algorithm is guided by an heuristic consisting of two parts: A distance measure $\psi(m_1, m_2)$ between the two maps, a common similarity indicator used in image registration, and a custom term defined by the two researchers counting *agreement* and *disagreement* between the

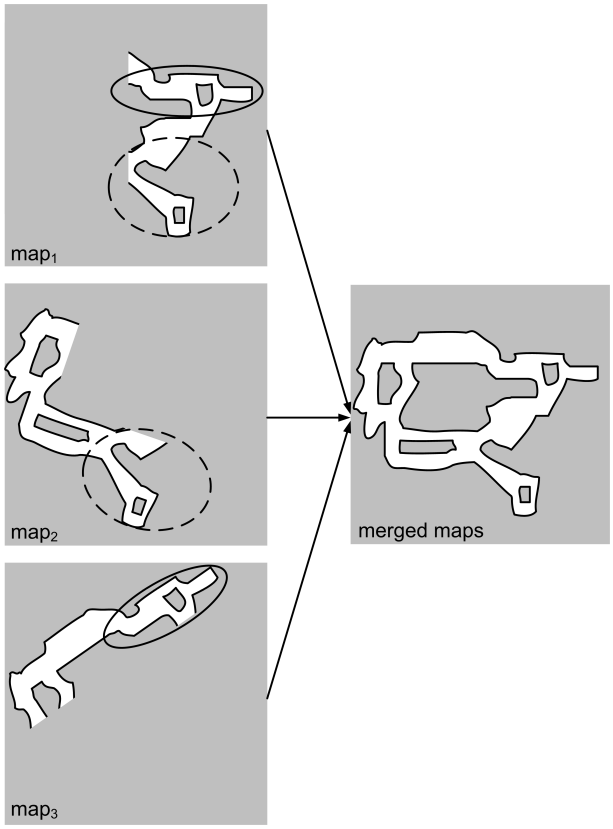



**Figure 2.14:** An example of map-merging: Three local maps are merged into a global one showing the complete environment. Overlap between $m_1$ and $m_2$ is indicated with a dashed line, overlap between $m_1$ and $m_3$ with a solid line. Figure taken from [28].

two maps. Taking two occupancy grid maps $m_1$ and $m_2$ with the possible states *free*, *occupied* and *unknown*, the term $\mathrm{agr}(m_1, m_2)$ is defined as the number of cells where both maps indicate *free* or both maps indicate *occupied*. The term $\mathrm{dis}(m_1, m_2)$ is defined as the number of cells where one map indicates *free* and the other one *occupied*. Cells where either of the maps indicate *unknown* are ignored in the count. The complete dissimilarity formula to be minimized by the search becomes such [4]:

$$\Delta\left(m_1, m_2\right) = \psi\left(m_1, m_2\right) + c_{lock} \cdot \left(\mathrm{dis}\left(m_1, m_2\right) - \mathrm{agr}\left(m_1, m_2\right)\right), \tag{2.20}$$

the second term "locking" into place the alignment of two regions, avoiding the over-fitting usually caused by the distance measure alone.
Also [4] defines a Similarity Index, to check whether a merge was successful or not, that has been widely used in literature since their publication:

$$\omega\left(m_1, m_2\right) = \frac{\mathrm{agr}\left(m_1, m_2\right)}{\mathrm{agr}\left(m_1, m_2\right) + \mathrm{dis}\left(m_1, m_2\right)}. \tag{2.21}$$

The initial work [4] by Birk and Carpin being presented in the context of mobile robotics, a similar approach has been adopted by Li et. al. in the context of autonomous vehicles [22], [23].

The employed heuristic is different, as there are often dynamic objects present in the case of an autonomous vehicle, causing an inherent inconsistency in the maps, to which the measure presented in [4] is very sensitive. Such, the heuristic is adapted to measure the consistency of the maps in terms of the occupancy likelihood of overlapping regions, but is insensitive to false counting of *disagreement* due to dynamic objects present in one map and not in another. Using their new heuristic and a Genetic Evolution search algorithm, Li et. al. successfully adapt the direct-optimization method to map-merging in this new context.
An advantage of those direct optimization approaches is, that the maps can be used directly in their raw occupancy grid form, without extracting any features or other descriptors. A major drawback, however, is the computational complexity due to the three-dimensional parameter space (rotation $\theta$ and two translations $t_x$ and $t_y$) that has to be searched. Such, it is often not very applicable to frequent, online consecutive merging of local maps. In a circular environment with a known center point shared between the robots as presented in chapter 3, the research can however be reduced to only the rotation $\theta$ making the direct optimization method an interesting choice for the research project presented in this report.

Further approaches on occupancy grid merging are based on the Hough- or the related Radon-transform, as those are especially adapted to find rotations in an environment where many perpendicular lines are present, as is often the case in indoor environments (i.e. walls, corridors etc.).
Via the Hough-transform, a parametric representation of the image will be calculated. It exploits the fact that a line in the $x$-$y$ plane can be described in polar coordinates by the parameters $\rho$, the distance of the line to the origin, and $\theta$, the angle between the $x$-axis and the normal of the line.

$$\rho = x\cos\theta + y\sin\theta. \tag{2.22}$$

In the here presented implementations, the discrete Hough-transform is employed. The $\rho$-$\theta$ plane is divided into a grid of accumulators, $n_\rho$ for the $\rho$-dimension and $n_\theta$ for the $\theta$-dimension. The accumulators are all set to 0 initially. For each pixel representing an object (i.e. occupied cells in the occupancy grid maps) and for each of the $n_\theta$ values of the discretized angle space, the formula (2.22) is calculated. The corresponding accumulator $(\rho, \theta)$ is incremented at each step. The result of the discrete Hough transform is stored in a Matrix

$$\mathcal{H}(i,k), \quad 0 < i < n_\rho, \quad 0 < k < n_\theta, \tag{2.23}$$

giving the parametric representation of the image.
As the rotation $\theta$ is one of the parameters of the Hough-space, the relative rotation can be calculated directly by the displacement between two corresponding peaks, representing distinct lines in the map. The relative translation can also be found by the geometrical information represented by the Hough-transform. In [7], the Hough-Spectrum of the maps is analyzed to find the rotation between two local maps. The Hough-Spectrum $\mathcal{HS}$ is defined in [7] as

$$\mathcal{HS}(k) = \sum_{i=0}^{n_\rho - 1} \mathcal{H}(i,k)^2, \quad 0 < k < n_\theta, \tag{2.24}$$

reducing the parametric representation to only the angle space. The peaks of the cross-correlation between two Hough-Spectra of two maps to align will give hypotheses for the rotation between the maps. In contrast, in [30], correspondences between peaks of the Hough-Transforms of the maps to align are directly searched without the reduction to only the angle dimension.
Both approaches are able to track multiple hypotheses and verify their coherence with the Similarity Index introduced in [4]. In [29], the Radon transform (which is related to the Hough transform) is employed to find the rotation between maps and the translation is found by matching edges of the Voronoi-Graph of the maps.
As these approaches allow to directly calculate the parameters of the roto-translation between two maps from the analyzed geometrical features, no computationally expensive exhaustive or probabilistic research of the parameter-space has to be calculated. Such, these methods are generally significantly faster than the direct-optimization algorithms presented before (about 10 seconds for a merge compared to more than 120 seconds for the Adaptive Random Walk approach on the same hardware platform [30]). Therefore they are better adapted for continuous online merging of maps. However, they are dependent on the presence of distinct lines in the maps (i.e. walls, corridors..).

Another approach using more generic features, is to take the map-merging as an image registration problem, known from computer vision, as presented in [5]. After a preprocessing step, an image descriptor is computed, based on the Harris- or Kanade-Lucas-Tomasi feature detectors and feature correspondences between the maps to merge are determined. It is common for a given feature to have multiple candidate correspondences, as occupancy grid maps usually aren't very rich nor very distinct in features, leading to ambiguity. A custom RANSAC algorithm is then employed to determine a coherent subset of correspondences and the associated transformation between the maps. It imposes *uniqueness* - a feature can only have a single correspondence in the other map - and the *rigid transform* constraint - the relative positions of features need to be the same in both maps, i.e. a transformation matrix of the form presented in (2.19) is searched. Through a *Sum of Gaussians* model, several merging hypotheses can be tracked and the most probable one can be determined by stochastic tests.
The performance of the algorithm is mostly evaluated on loop-closure test but it is equally applicable for multi-robot map-merging procedures.

A new and very complete framework for occupancy grid merging was presented by Saeedi et. al. in 2015 [28]. It is based on matching map segments that are not straight lines, but rather corners or curves, possessing image intensity-gradients in more than one direction. After a preprocessing step that removes redundant edges due to noisy measurements and reduces edges to one pixel width to get the exact boundaries of obstacles, overlapping Segments in the map are searched. Map segments are chosen for the alignment only if they contain enough distinct geometric information. A custom descriptor based on distances and differential angles between occupied points in each segment is introduced (see [28] for further details). Histograms of these descriptors are analyzed to select the segments for an alignment attempt and for the chosen segments, those histograms are compared by a cross-correlation function to get the relative rotation and translation. Several refinement and tuning steps are done in post-processing and the

final result is verified by the Similarity Index introduced in [4].
The performances are compared to the Adaptive Random Walk algorithm from [4]. A doubling of performance in computational time is reported and better similarity scores can be achieved. However, still about 95 seconds are needed for a merge on a Core2Duo 2.66GHz laptop [28].

Lastly, a conceptually largely different approach, compared to all previously presented techniques as introduced in [1] will be outlined: Here, the idea is to reuse the particle filter SLAM-framework for the merging of local maps. It is based on the observation that the map-merging problem remains similar to single robot SLAM in a certain manner: In single robot SLAM local laser range-scan measurements have to be integrated in the map of the robot, in multi-robot map-merging, local maps have to be integrated in a global one. Such, the local maps are taken as "measurement" for a *virtual robot*, constructing the common global map reusing particle-filter SLAM with adapted perception and motion model. The image similarity heuristic used in the perception model is similar to [4]. The method is verified by merging maps of ten robots exploring a "maze"-like environment presenting many distinct corners.

To resume, multi-robot SLAM can be realized by online constructing a joint map through sharing raw range and odometry measurements or by merging local maps through the alignment of overlapping sections. The first method allows to continuously build a common map between the robots, but needs relative position estimates that are often imprecise and demands to share a large amount of raw data. For the second approach, no mutual encounters are necessary and only the already processed local maps are shared, however, sufficient overlap between the maps is required. Several different approaches exist, some more suited for offline merging after completion of the explorations, others also applicable for continuous merging during the operation, making available a joint map after the first initial merge. Runtimes on common laptop hardware differ from less than 200 ms for the Hough-Spectrum method [7] or the RANSAC feature correspondence method [5] via around 10 seconds for the Hough-Peak based methods [30] up to more than 100 seconds for the Adaptive Random Walk [4], [30] or map segment matching method [28]. However, most of the methods requiring a lot of computational time could be largely accelerated for consecutive merges. The full processing would only be necessary for the initial fusion.
The right approach has to be chosen depending on the concrete application scenario. Several of the presented methods have been applied to the circular scenes used in this project and evaluated using real-world robot data (see section 3.3).

The chapter on Simultaneous Localization an Mapping presented the basics of this central problem in mobile autonomous robotics, notably a popular mapping algorithm based on Rao-Blackwellized particle filter that will be maintained for the rest of the project as well as some possible map-merging schemes for the integration of multiple robots in the exploration process. The following chapter will now give a problem formulation for the concrete application of this research project: the observation of actions and gestures of a person by a team of multiple robots navigating around the scene in a circular grid centred on it.

# Chapter 3

# Multi-Robot SLAM in a Circular Grid Centred on a Scene

Navigating robots on a circular topology, i.e. on concentric circles with a scene to observe at their center, allows for several simplifying hypotheses that significantly reduce the complexity of certain parts of the multi-robot SLAM problem. The following sections will detail the work carried out during this student's research project, making use of those assumptions to build a simple but reliable navigation framework (section 3.2), and analysing their impact on some of the common map-merging approaches that have been summarized in section 2.4.2 (section 3.3). To begin with, the global context of the work and the assumptions and hypotheses adopted will be detailed in section 3.1.

## 3.1 Multi-Robot Observation of Human Dynamic Scenes

This project has been realized in the context of the coordination of a mobile robot fleet for multi-view analysis of complex scenes, i.e. human activity recognition for example in the domains of service robotics or emergency assistance. As is described in [9], several robots will move around a scene and will observe it from different points of view. The quality of the robots' joint observation is to be optimized to be able to interpret at best the observed scene.
Here, we focus on the aspects of robot navigation and cooperative mapping of the environment in the given context. Those are the underlying functions needed for any such high-level robot task. The human gesture and activity recognition is blended out for the moment but can be added on top of the proposed framework.

Concretely, a circular navigation topology for the robots is proposed (see fig. 3.1, [9]): The robots will move on concentric circles around a scene, examining it by different lines of sight. Obstacles, preventing the robots to navigate across a certain area, and occlusions, preventing them to observe the scene from a certain point of view, are present. The scene can be dynamic, i.e. the person can perform a sequence of activities, however, it will rest approximately in the same place during the task.

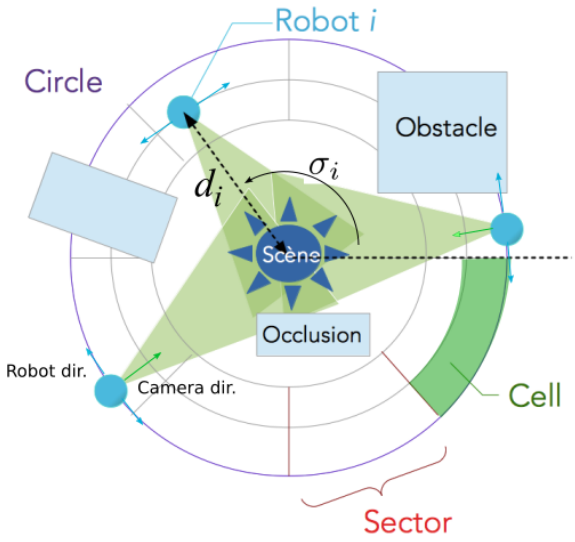



**Figure 3.1:** Model of the environment to navigate and the scene to observe. The robots will move around the scene on concentric circles with their cameras pointing towards the center. Figure taken from [9].

Turtlebot2 robots equipped with an RGB-D Camera (Kinect) for the activity observation and with a low-cost 360° laser range-scanner (RP-Lidar) with a usable range of around 4 meters for mapping and navigation purposes are used for the practical realization of those tasks. Figure 3.2 shows a picture of one of the utilized robots. For the underlying single-robot localization and mapping, a common particle filter SLAM algorithm, as presented in section 2.3.3 and implemented in the ROS *gmapping*-package is used. It allows each robot to construct a local map of the environment. As is usually the case in real-world applications, no information about initial relative positions between the members of the robot-team and no global localization system are available. Those realistic assumptions pose some challenging problems for multi-robot cooperation, as has been detailed in section 2.4.

However, the concentric topology leads to several important advantages and simplifications. It is supposed that the center of the scene, i.e. the position of the person to observe, is known as a common point shared between all the robots. This means that the relative position between the robots is defined up to a rotation $\psi$ around the common center point. This rotation angle can be found by merging their local maps. Only the rotation $\psi$ between the maps has to be searched for, as the translation is known because of the shared center point. This closely limits the scope of possible transformations to a one-dimensional parameter space.
Section 3.3 will detail the analysis and adaptation of several of the map-merging algorithms common in literature, that have been presented in section 2.4.2, on the circular topology with a known center point and detail the results and performances found.

Furthermore, the concentric topology allows for a simple implementation of a navigation framework. Firstly, it is easy to guarantee that the scene to observe will be kept in the Kinects' field of view: If the camera is fixed perpendicular to the robot's movement direction (cf. fig. 3.2), it will

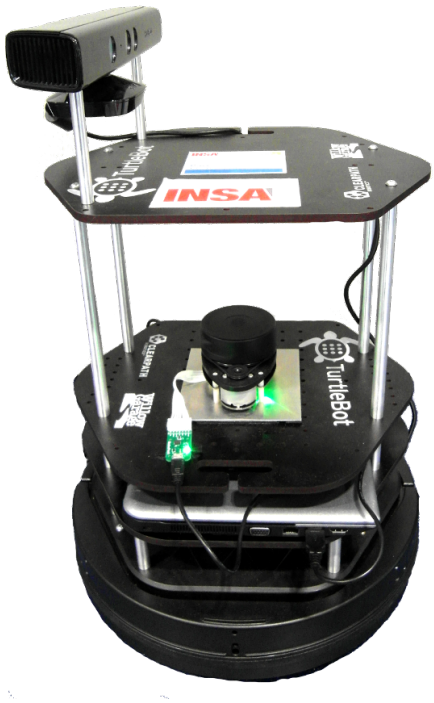



**Figure 3.2:** Picture of one of the Turtlebot2 robots used in this project. On top of the mobile base, they are equipped with a netbook for computation and control, an RP-LIDAR laser range finder and a Kincect RGB-D camera, mounted on several platforms. The picture shows the front of the robot. As explained in the text, the camera is fixed perpendicular to its forward direction.

inherently point towards the center of the scene while the robot is oriented tangentially on the circle perimeter.
Secondly, the navigation can easily be realized by two simple operations: A movement along the circle, keeping constant the distance towards the scene and continuously adjusting the robot's bearing to keep up the tangential orientation. To be able to explore the whole space around the scene, a radius-change operation, moving towards or away from the scene, has to be added. During this operation, the robot will lose sight of the scene. But as the standard operation is the tangential movement along a circle, and the radius change rests the exception, this does not pose a big problem.
Lastly, some difficulties also arise from the circular set-up: As the camera is fixed perpendicular to the robot's movement direction, collision problems can occur: Other robots or obstacles can not be recognized by the Kinect sensor, it is uniquely used for the observation of the scene. The data source for mapping and navigation purposes is the 360° laser range-scanner that will also be used to realize obstacle avoidance operations.
The following section 3.2 will detail the navigation framework for a circular topology implemented during this project.

## 3.2 Robot Navigation and Communication

The proposed circular grid navigation architecture uses two custom coordinate-frames besides the reference given by the SLAM algorithm (see fig 3.3). Each robot has its own set of coordinate frames as shown in figure 3.3. The SLAM-reference $T_{map}$ marks the position and orientation with which the SLAM process has been started. It is different for all robots participating in the task and the relative transformations from one robot's reference to another one's are unknown.

As explained in section 3.1, there is however one common point known to all the robots -

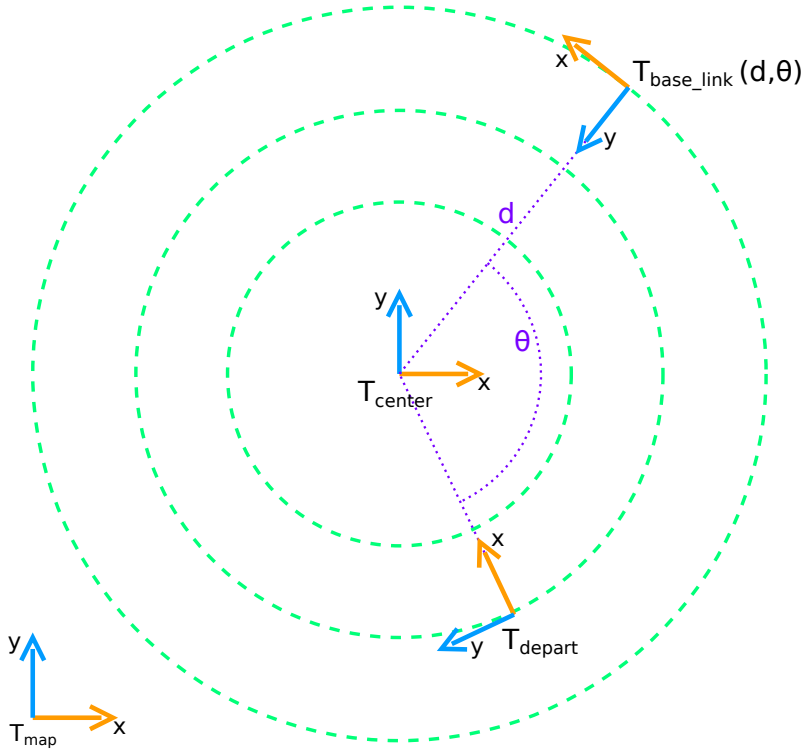
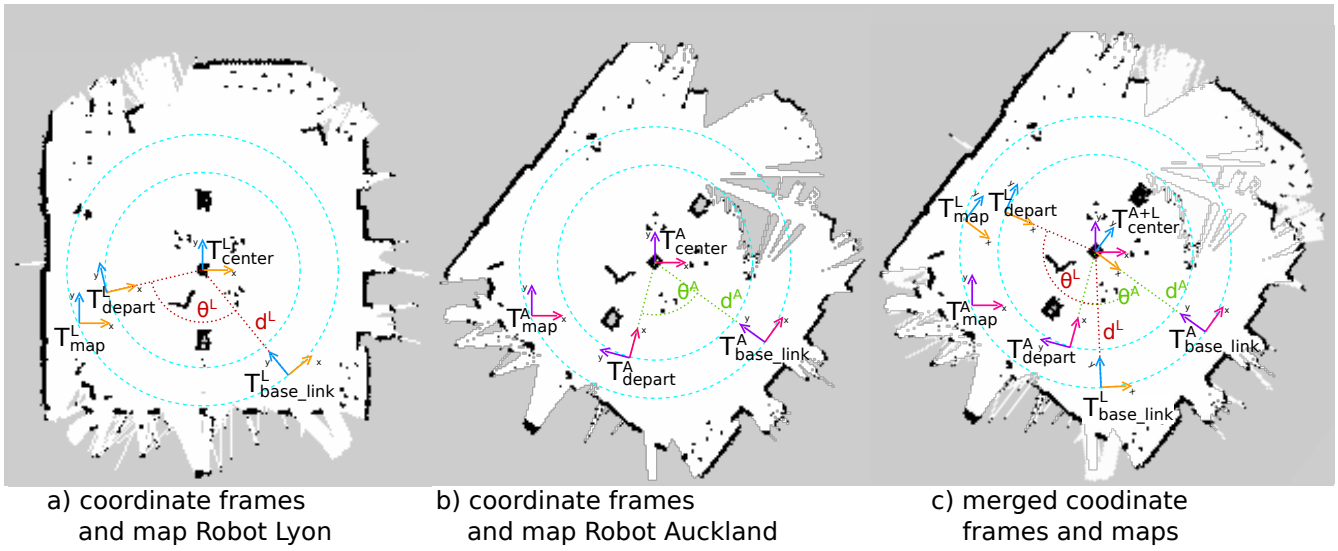



**Figure 3.3:** Coordinate Frames for a single robot used by the navigation algorithm: The robot's dynamic position $T_{base\_link}$ is defined with respect to three static reference points. The reference of the SLAM algorithm $T_{map}$, the center point of the scene $T_{center}$, and the point of depart of the robot $T_{depart}$ marking the zero of the angle $\theta$ that defines the displacement on a circle.

the center of the scene, $T_{center}$. This point has to be given to each robot by the operator for the moment, but in a future version of the framework it is conceivable that the robots will find it by themselves. Thus, the center point marks physically the same position for all robots. Only the rotation of the center coordinate-frame with respect to the physical scene is different and unknown between the robots. It is set identical to the rotation of each robot's reference frame$T_{map}$. Relative rotations between the center points of the robots participating in the task will be found in the map-merging stage (see section 3.3).

Lastly, a point of reference $T_{depart}$ marks the angle $\theta = 0$ for the definition of the robot's position on a circle. Hence, the robot's position $T_{base\_link}$ is defined by a tuple $(d, \theta)$, marking its distance

to the center and its position on the circle (cf. fig 3.3). The angle $\theta$ is measured as the angle between the line connecting the position-reference $T_{depart}$ to the center and the line connecting the robot's position $T_{base\_link}$ to the center. As an angle between two vectors is uniquely defined only between $0^\circ$ and $180^\circ$, the angle sign has to be found another way. To this end, the position-reference $T_{depart}$ is orientated with its x-axis pointing towards the center and the sign of $\theta$ can be determined by the inverse sign of the y-coordinate of the robot's position transformed to the $T_{depart}$ coordinate-fame (i.e. positive angle $\theta$ if the robot is to the right of $T_{depart}$ and negative angle sign if the robot is to its left, see fig. 3.3).
Initially, the position-reference $T_{depart}$ will be different between the robots. Once a map alignment, and such the rotation angle between the robots' center points has been found, the robots will agree on a common position-reference shared between all of them. From this instant on they will be able to work in a common coordinate-frame. Figure 3.4 illustrates the coordinate-frames for two robots, superposed to their respective local maps.




a) coordinate frames and map Robot Lyon

b) coordinate frames and map Robot Auckland

c) merged coodinate frames and maps

**Figure 3.4:** Illustration of the different coordinate frames:
a) and b) : Coordinate frames for two robots (*Lyon* and *Auckland*), superposed on their respective local maps.
c): Merged local maps and coordinate-frames of the two robots. Map and frames of robot Lyon have been rotated around the center by the relative angle found in the map-merging stage. In a last stage (not depicted here) the robots would agree upon a common position-reference $T_{depart}$ to be able to work in a common coordinate-frame (i.e. robot Auckland adopts robot Lyon's reference).

The following subsections will detail the functioning of the navigation and communication framework, beginning with the basic navigation functions based on a control system for the robot's orientation and distance towards the center, continuing with obstacle avoidance functions and communication between robots and concluding with a brief description of the program architecture implemented based on the widespread Robot Operating System (ROS) libraries.

### 3.2.1 Control of Orientation and Distance

Basic navigation functions on a circular grid can be realized by two simple operations: A displacement on a circle, keeping constant the distance to the center, $d$, and a change of radius by moving towards or away from the center keeping constant the angle on the circle, $\theta$. For the following explications, the current robot position and orientation are supposed known, determined and updated by the underlying SLAM algorithm. Initially given relative to the SLAM-reference point $T_{map}$ they can be transformed to the coordinates $(d, \theta)$ as explained in the previous paragraphs.
The robot-base needs a linear and an angular speed as command input. The linear movement can only be along the robot's x-axis (x-axis of coordinate-frame $T_{base\_link}$, cf. fig. 3.3 and 3.5).

For a displacement on a circular trajectory, the robot such needs to continuously correct its orientation to keep it tangential to the perimeter of the circle. As the Kinect 3D camera is fixed perpendicular to the robot direction, it will inherently keep the center in its field of view during this operation.
The linear speed $v$ is set to a constant value. It can be positive or negative resulting in respectively a forward or a backward movement of the robot. The angular speed $\omega$ is controlled by an approach similar to a Proportional-Integral (PI) control. Control of orientation and distance are superposed.
In a first place, the position of the center in the current robot coordinate system $T_{base\_link}$, $(x_{c\_rob}(n), y_{c\_rob}(n))$ is determined at each time-step $n$ (see fig. 3.5). From this coordinates, the angle $\alpha(n)$, needed to orientate the y-axis of the robot toward the center (and such its x-axis tangential to the circle perimeter) is calculated:

$$\alpha(n) = \operatorname{atan2}\left(x_{c\_rob}(n), y_{c\_rob}(n)\right). \tag{3.1}$$

The arctangent is calculated by the atan2-function, as implemented in the C++ standard library, returning the angle in the correct quadrant.

The orientation control part of the angular speed is then determined by a simple proportional factor $P_{or}$:

$$\omega_{or}(n) = P_{or} \cdot (-\alpha(n)). \tag{3.2}$$

For the distance control, a PI-control is used. With $d_{target}$ the target value for the distance between robot and center, the error measured at time-step $n$ is:

$$\epsilon(n) = d_{target} - d(n). \tag{3.3}$$

The distance control part of the angular speed is then determined as

$$\omega_{dist}(n) = P_{dist} \cdot \epsilon(n) + I_{dist} \cdot \sum_{k=0}^{n} \epsilon(i) \tag{3.4}$$

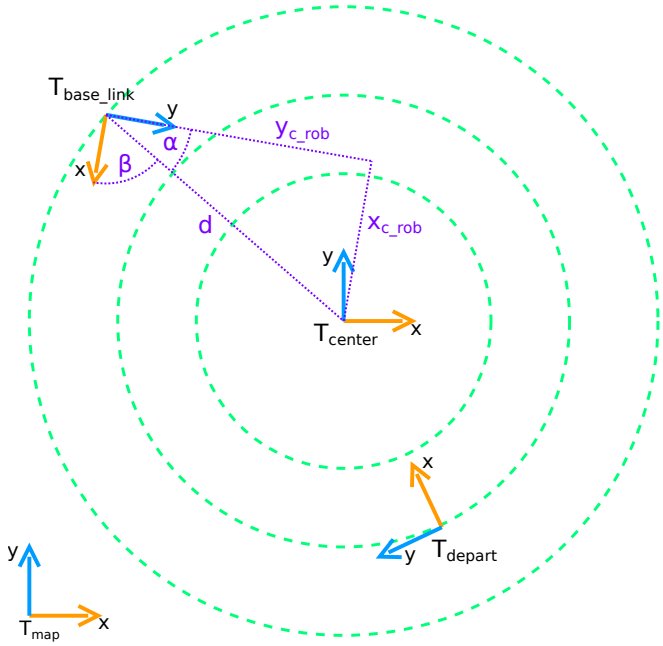



**Figure 3.5:** Determination of the angle $\alpha(n)$ resp. $\beta(n)$, needed to orientate the $y$- resp. $x$-axis of the robot toward the center. The coordinate-frame $T_{base\_link}$ marks the current robot position and orientation.

with the proportional coefficient $P_{dist}$ and the integral coefficient $I_{dist}$.
The final value of the angular speed for a movement along a circle is then calculated as the sum or as the difference of the two parts, depending on the sign of the linear speed $v$:

$$\omega(n) = \omega_{or}(n) + \text{sign}(v) \cdot \omega_{dist}(n). \tag{3.5}$$

In the case of a radius change operation, the robot's x-axis needs to be rotated towards the center. To this end, the angle $\beta$ (cf fig. 3.5) is calculated as

$$\beta(n) = \text{atan2}\left(y_{c\_rob}(n), x_{c\_rob}(n)\right) \tag{3.6}$$

to turn the robot towards the center (for a decrease of radius) or as

$$\beta(n) = \text{atan2}\left(-y_{c\_rob}(n), -x_{c\_rob}(n)\right) \tag{3.7}$$

to turn the robot away from the center (for an augmentation of radius). The angular speed is in this case simply calculated as

$$\omega(n) = P_{radius} \cdot \beta(n). \tag{3.8}$$

For this operation, the linear speed $v$ is kept at zero until the correct robot orientation towards or away from the center has approximately been reached, then it is set to a constant, positive value (forward motion).

The navigation algorithm takes a target robot position $(d_{target}, \theta_{target})$ as input. Depending on the difference to the current position $(d, \theta)$ a movement along a circle, a radius change or a sequential combination of the two operations is executed. Once the target position has been reached with a certain tolerance, the robot is stopped and reoriented with the Kinect 3D camera pointing towards the center. The robot then waits for the input of a new target position.

### 3.2.2 Obstacle Avoidance

As mentioned in the introduction in section 3.1, the robot can encounter obstacles during its movements. Those have to be recognized to avoid accidents. Because the 3D camera is not pointing in the direction of the robot's movement but perpendicular to it, to observe the center of the scene, it is of no great use for this task. Obstacle avoidance is such realized by the laser range-scan data.

Principally two mechanisms to be able to avoid collisions have been implemented.
The first one is based on the occupancy grid map constructed by the underlying SLAM algorithm from the LIDAR and odometry measurements. The current map is continuously transmitted to the navigation algorithm. In a pretreatment step, the map is first transformed to a binary image, keeping only occupied cells as white pixels, free and unknown cells are set to zero (fig. 3.6). Then, an erosion with a small structural element is performed to delete noisy measurements (see for example the top right area of the maps in fig. 3.6). Lastly, the remaining objects are enlarged by a dilation with a circular structural element of a size equal to the dimensions of the robot's base plus an additional security margin. The resulting map is used for the obstacle detection.

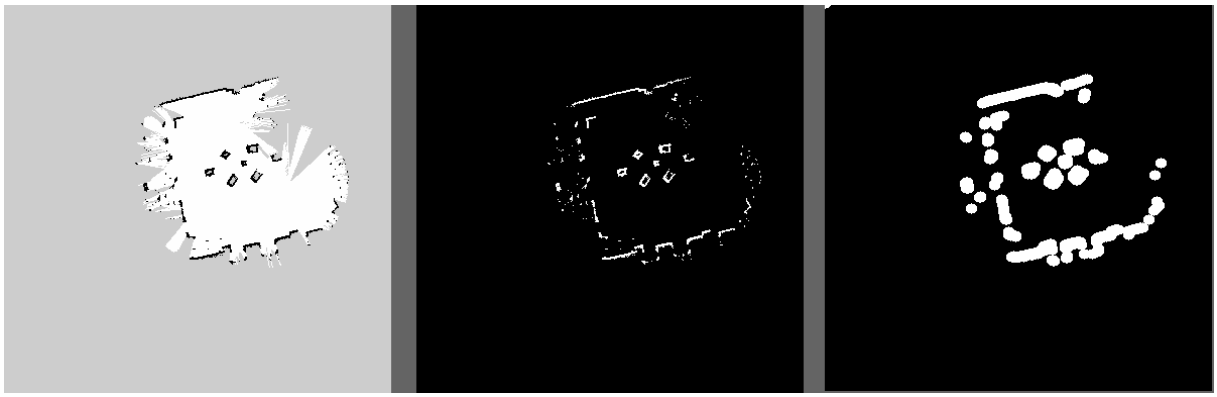


**Figure 3.6:** The pretreatment steps to create the map used for obstacle detection. To the left the original occupancy grid map with occupied cells in black, free cells in white and unknown ones in grey. In the middle the map after binarization, keeping only occupied cells in white. To the right the final map after the erosion and dilation steps.

As all objects have been inflated by the robot's dimensions, it can easily be determined if it is heading into an obstacle or not. From the robot's position in the pretreated map (marking the center of its base) and from its bearing it need only be verified that the next cell the robot is

moving into is not a white one. Otherwise it risks to hit an obstacle and has to be stopped. This mechanism reliably detects static objects that are present in the occupancy grid map as obstacles. It remains however the problem of dynamic objects that usually won't be present in the map, or will only be seen as noisy measurements and such not been taken into account. This is especially the case for the other robots of the team. Additionally, the construction of the Turtlebots' bodies further complicates the problem. They will be visible for the laser scanner only by four small shafts (see fig. 3.2 and fig. 3.7). To this end, a second obstacle detection function has been implemented.

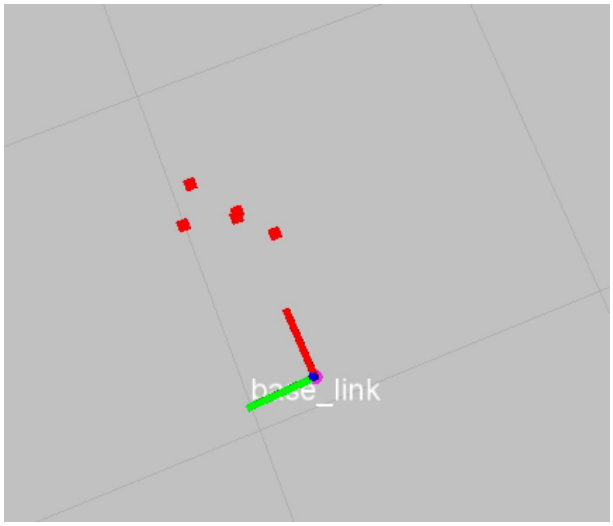



**Figure 3.7:** The robot at position *base_link* observes another robot in front of itself with its laser scan. It is visible only by a very small number of points. The center and three of the four shafts of the body of the other robot can be distinguished (cf. fig. 3.2).

The second mechanism for obstacle detection works directly on the laser range-scan data. It is especially adapted to recognize other robots in the measurements. Because of the construction of the Turtlebot's bodies, they leave a characteristic signature of four fine shafts in the LIDAR scans (see fig. 3.2 and fig. 3.7). This signature is searched in a certain angle sector around the movement direction of the robot. If it is detected close to the robot's position, the latter is stopped to avoid a collision. For simplification and as it was sufficient in practice, the signature was reduced only to the number of close range measurements in the angle sector, their geometric relations were ignored.

Finally, as a last measure, the Turtlebot base possesses several bumpers in forward, left and right directions. If one of those bumpers is pressed, the robot will move backwards for a small distance and then stop. This measure should be avoided by the previous two mechanisms but is added as a security function.

In the current development stage, in case of obstacle detection by one of the three mechanisms described above, the robot stops, reorients itself towards the center and signals the detection of an obstacle to the superior navigation level (i.e. the planning algorithm that also sends the navigation goals $(d_{target}, \theta_{target})$). The circumnavigation of the object has then to be handled at

this level. However, this is often possible by very simple operations, as decreasing or increasing the distance towards the center and then continuing the movement along the circle.

### 3.2.3 Communication between Robots

A key question of a multi-robot system is the communication framework. To be able to cooperate, the robots need to share a certain amount of data. However, in practical cases, communication bandwidth is limited and the network may be unreliable. Such, the amount of data broadcast needs to be carefully chosen. Also, the robots should not be too dependent on the communication, they should be able to continue their task as a mono-robot system if communication fails. For this reasons, a distributed approach has been chosen: In a team of $N$ robots, one robot will transmit its data to all other $N-1$ robots and also receive data from all of them.

A central aim of this project was to realize a cooperative mapping task, i.e. the construction of a common global map between the robots. To this end, we choose to share the local maps of the robots between them in discrete intervals in time. Each robot will then try to align the maps he received from the other robots with its own, as explained in detail in section 3.3. The choice of transmitting local maps reduces the communication bandwidth required as compared to transmitting the laser range-scan data. It also allows for map-merging without the need for encounters between the robots or relative position measurements, as has been discussed in section 2.4.
The occupancy grid maps present by far the greatest amount of data to transmit between the robots. On top of them, several small data frames will be communicated, too. Together with its map, each robot transmits the position of the center in it. Furthermore, it transmits its position relative to its own center point ($T_{base\_link}$ in the $T_{center}$ coordinate frame), as well as its point of reference on the circle $T_{depart}$ (cf. fig. 3.3). As has been discussed before, the center coordinate frame $T_{center}$ is assumed to represent the same physical point for all the robots. However, its rotation is different between the robots at first. Once the rotation has been found by aligning local maps (see section 3.3), also the relative rotations between the center coordinate frames of the robots participating in the task will be known. Such, with the information shared between the robots as described above, each robot will know the positions of its team-mates after a first successful map-merging.

Concretely, a distributed communication architecture has been realized during this project for three robots via a WiFi network using the TCP-IP protocol. Each Robot gets assigned a static IP-address and communicates using three ports, one for transmission towards the other robots and two for reception, as is outlined in fig 3.8.

As our cooperative robot task is locally focussed around a scene it is supposed that they will always stay in the area covered by the utilized WiFi network. There is no explicit handling for connection losses or reconnection. If a robot loses connection, this will not block the other robots. As they won't be able to open a connection towards it, they will just work on using the last known information from this robot.

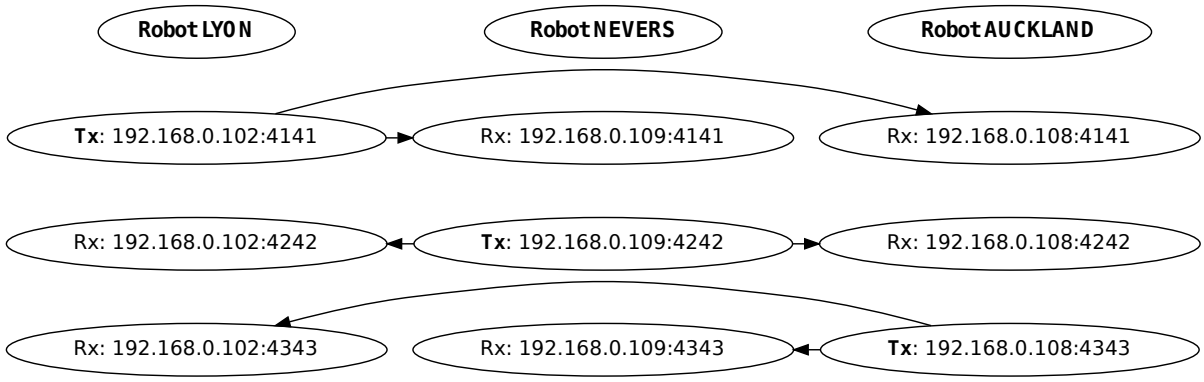



**Figure 3.8:** Outline of the communication architecture for three robots named *Lyon*, *Nevers* and *Auckland*. Each robot will transmit data to the two other robots using one tcp-ip port. The reception is handled with one port for each robot to receive from, to be able to distinguish between them.

### 3.2.4 ROS Architecture

The navigation and communication functions as described in the previous sections have been implemented as a framework based on the widely known ROS libraries. The general architecture is shown in figure 3.9. The five custom nodes, that have been developed during the project, are highlighted in colour. Ellipses mark nodes and rectangles the communication topics used to share data between them. So as not to further complicate the overview, only a small part of the underlying ROS nodes and topics are shown. Notably, the *tf*-topics used to share the robot positions, coordinate frames and transformations are not depicted, they will be detailed in a later paragraph.
All those nodes are run on each robot respectively and each robot uses its own local ROS-Master. Such, the robots don't communicate via the standard ROS mechanisms but only via the custom communication node.

The framework is based upon some standard ROS libraries. The *slam_gmapping*-node realizes the simultaneous localization and mapping task, dynamically constructing a map of the environment which is used by the other nodes. The laser range-scan measurements are handled by the *rplidarNode* and shared on the *scan*-topic.
The *cirular_nav*-node realizes the navigation and obstacle avoidance functions as described in sections 3.2.1 and 3.2.2. It will receive navigation goals from the superior planning node *send_goal* and communicate the status of the operation back towards it. It sends the appropriate commands for angular and linear velocity via the *cmd_vel_mux*-node towards the nodes controlling the robot base, not entirely shown in the graph. For the obstacle detection functions as described in section 3.2.2 the node receives the map of the environment, the laser-scan data and the events in case one of the bumpers has been pressed.
The *send_goal*-node realizes rudimentary path planning functions and circumvention of obstacles by sending appropriate waypoints $(d_{target}, \theta_{target})$ to the navigation node. It has not been a central point of this project and only has very basic functionalities.

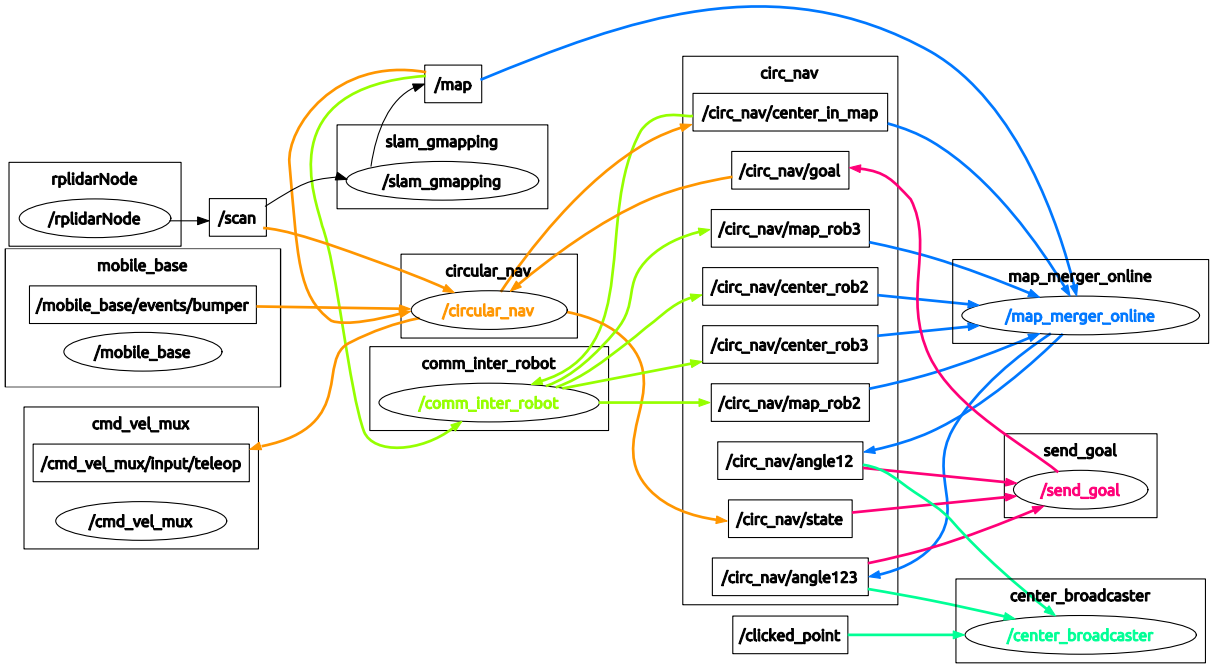



**Figure 3.9:** General architecture of the implemented ROS framework showing the most important nodes and communication topics. Ellipses mark nodes and rectangles the communication topics used to share data between them. The custom nodes are highlighted in colour. Only a part of the underlying nodes from the ROS libraries is shown. All these nodes are running on each of the robots respectively.

The *comm_inter_robot*-node realizes the communication functions as described in section 3.2.3. It will retrieve the current map and center point of the robot and send it on to the team-mates. In return it will receive their maps and centres and will publish them on the appropriate topics. The same is done for the other robots' positions $T^{rob2}_{base\_link}$ and $T^{rob3}_{base\_link}$ as well as their position-references $T^{rob2}_{depart}$ and $T^{rob3}_{depart}$. As this is realized by the *tf* coordinate transformation library, this is not shown here but explained in a later paragraph.
The *map_merger_online*-node retrieves the three maps and respective center points of all three robots and tries to find an alignment. The methods used for map-merging will be explained in detail in section 3.3. The output will be two rotation angles: the angle between robot 2 and the robot itself *angle12* and the angle of the third robot towards the fusion of the first two maps *angle123*.
Lastly, the node *center_broadcaster* handles the coordinate frames needed for the circular navigation. At the beginning of the task, it receives the center point of the scene by the operator via the *clicked_point*-topic. It then broadcasts all the necessary coordinate frames via the ROS *tf*-library. Those are the center frame $T_{center}$ and the position reference frame $T_{depart}$ of the robot itself (see figure 3.3) as well as the center coordinate frames $T^{rob2}_{center}$ and $T^{rob3}_{center}$ of the other robots, rotated by the correct angles, once a map alignment has been found.
The functioning of the different coordinate frame transformations will be detailed in the following paragraph.

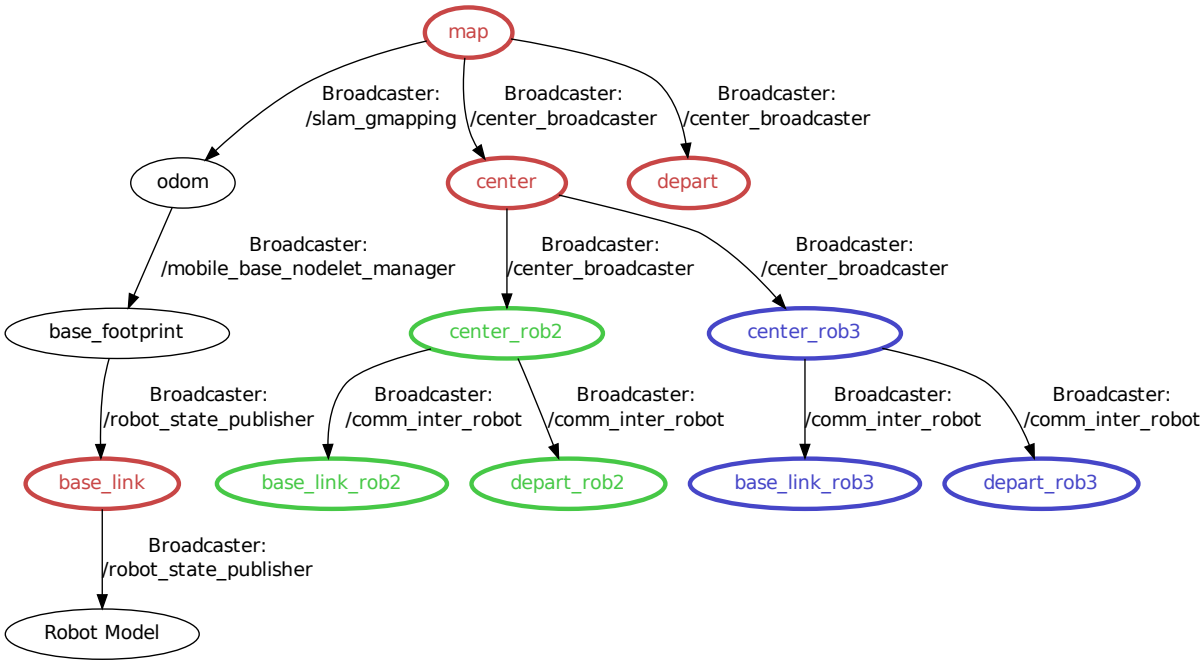



**Figure 3.10:** Schema of the tree of transformations between the coordinate frames used, handled by the ROS *tf*-package. The coordinate frames highlighted in red are those also depicted in figure 3.3. The frames highlighted in blue resp. green are those marking the positions of other robots. The nodes that broadcast the respective transforms are marked on the edges.

Figure 3.10 shows the coordinate frames used and the relations between them. The ROS *tf*-library orders the transformations in a tree structure. As was the case for the nodes described in the paragraphs above, this transformation tree exists in a similar version for each robot. The nodes marked in red are those also present in figure 3.3. The *map*-coordinate frame is the reference of the SLAM-algorithm and at the root of the transformation tree. The center point of the scene $T_{center}$ and the position reference on the circle $T_{depart}$ (called *depart*, as it is initially set at the position of the robot where the circular navigation algorithm is started) are broadcast by the *center_broadcaster* node. The current robot pose is given by the coordinate frame *base_link*. It is calculated and updated by the *gmapping* SLAM algorithm and some intermediary nodes. Based on this frame, a 3D-Model of the robot, designing individually each part of its body (Platforms, shafts, sensors, etc.) is also broadcast, here summarized in the single node *Robot Model*.
The nodes marked in green and blue are the coordinate frames marking the information from the other two team-mates. The centre coordinate frame of the other robots $T_{center}^{rob2}$ and $T_{center}^{rob3}$ are given with respect to the center of the current robot $T_{center}$ without any translation (as they mark the same position) but rotated by the angles found in the map alignment process. Relative to the centres of the respective team-mate, their positions (*base_link_robX*) and points of reference on the circle (*depart_robX*) are diffused by the communication node as they have been received via the WiFi network.
Such, with the completed transformation tree, relative positions between robots are known once a map alignment has been found. After a successful map alignment, the robots will also agree

upon a common position-reference $T_{depart}$. From then on, all three robots will measure their position on the circle with reference to the same point. A robot position $(d, \theta)$ (cf. fig. 3.3) will then denote the same point for all robots participating in the task. Such, a common, global reference frame is created.

To summarize the general course of the cooperative robot task, each robot individually maps the environment in a first place. From the beginning on, it continuously transmits its local map and pose information relative to its own center point to its team-mates and receives the corresponding information from each of them. All robots continuously try to find a correct alignment between the local maps. Once a first successful match has been found (i.e. the relative rotation between the robots' maps), a common global reference frame is created, the relative positions between the team-mates are known and they operate from then on in a common coordinate frame $(d, \theta)$, marking their position on the circular topology around the scene. The merged map gives a global overview of the scene, one robot doesn't need to visit the entire environment but can also operate based on the information collected by the team-mates.

Figure 3.11 gives an example for this procedure: It shows one robot's view of the environment

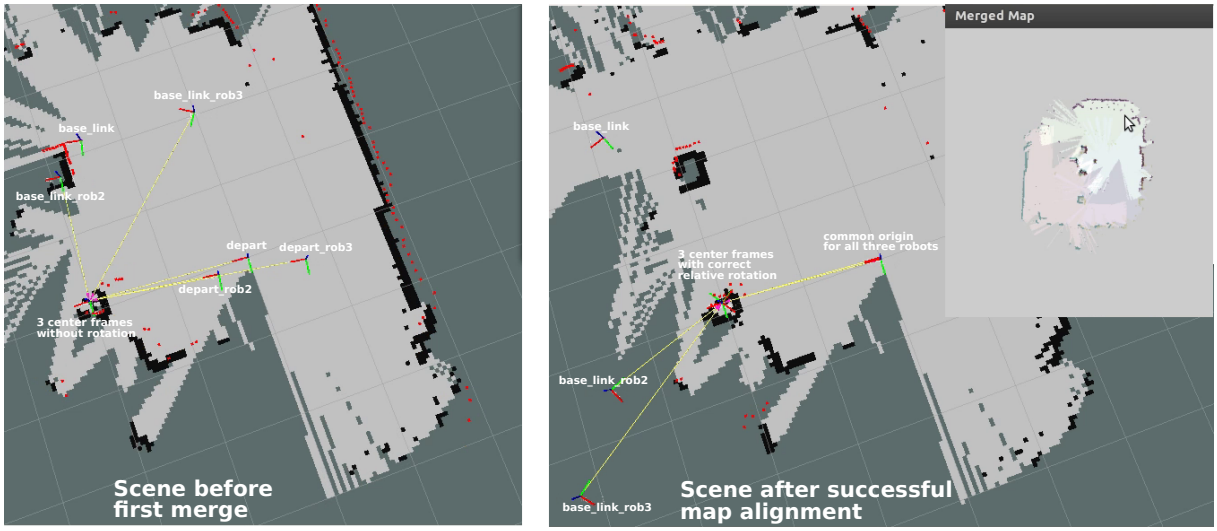



**Figure 3.11:** Example of a scene with the corresponding coordinate frames (red, green and blue axes). Left: before a first map-merge. The positions (*base_link*) and position-references (*depart*) of the other robots (*rob2* and *rob3*) aren't correctly indicated as the rotation between the centres is not yet known (set to zero par default).
Right: after a successful map alignment. The relative rotations of the centres are now known. Other robots' positions are indicated correctly. The team-mates decided upon a common position-reference. The merged map is shown in the top-right corner.

with the important coordinate frames at two moments in time: before and after a first map alignment. Before a first match, the rotation between the robots' centres is not known and set to zero par default. Such, other robot's positions and position-references are not indicated correctly. After a first match, the correct rotation between the centres is known, and such, the relative positions between all robots can be determined. Also, a common position-reference has been

adopted by all team-mates. The merged map, with each robot's local map on one colour channel, is shown in the top right corner of the figure. Almost the whole scene is present in it even though the individual robots have not covered more than a quarter of a circle.

The presented framework has a distributed architecture, all nodes and algorithms are running on all robots respectively. Such, the success of the task does not depend on maintaining connectivity with a master-robot, or another central operator, all team-mates have equal functionality and can replace each other in case of failure.
After a first map-fusion, a global planning algorithm, optimizing the joint quality of the observation of the human activity in the center of the scene, as described in [9], can be set up. The aim of this project was to develop the underlying functionalities of mapping, map-merging, navigation and communication for such an approach.
The following section will now detail the central point of the here proposed cooperative robot task, the alignment of local occupancy grid maps to a global one in a circular topology with a shared center point.

## 3.3 Merging of Occupancy Grid Maps with Common Center

One of the key objectives of this project was to realise a cooperative mapping task between multiple robots. To this end, the choice was made to merge local occupancy grid maps of the members of the robot team. As has been developed in section 2.4, this avoids the need for mutual encounters and relative pose measurements between robots. It needs only be assured that there is enough overlap between the maps. Also, only the already processed local maps need to be shared between the robots, significantly reducing the communication bandwidth necessary compared to a joint mapping approach sharing raw laser range-scan and odometry measurements.
Concerning the methods for occupancy grid map-merging, the state of the art in literature has been resumed in section 2.4.2. Several of the techniques presented have been implemented in the context of this project and adapted to the circular, concentric topology the robots are operating on.

In fact, the circular topology, in particular the knowledge of a common center point shared between the robots participating in the task, allows for several simplifying hypotheses.
While the methods presented in section 2.4.2 in a general context search for a rotation and translation to align the local maps, here only the relative rotation has to be found. As each robot knows the position of the center of the scene in his local map and the other robots will transmit their respective local maps with the position of the center marked in it, the relative translation is inherently known. It can be achieved just by aligning the center points of the local maps. Such, after prior superposition of the center points, the sought-after transformation matrix to align two

local maps (2.19) can be reduced to a pure rotation matrix with only the single parameter $\theta$.

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Furthermore, because of the robots all evolving on circles around the same center, the region where it is most probable to find overlap between the local maps is easily determined. It is the area around the observed scene at the center. Here, even with little displacement of the robots, from the beginning of the task on, overlap can usually be found. Figure 3.12 shows the region of interest typically used to find the map alignment. It contains the objects around the center

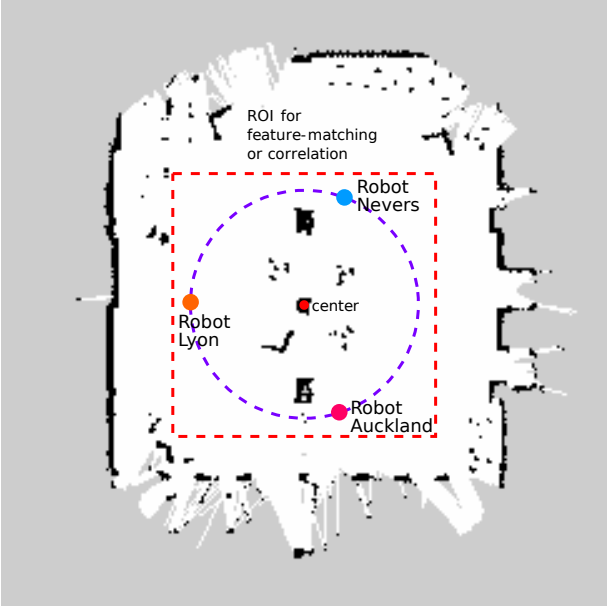



**Figure 3.12:** Map of the room used for experiments with the region of interest used for map-merging marked around the center. It contains all the objects placed around the scene but not the outer walls of the room. The starting positions of the three robots are marked as coloured points, the circle they where evolving on for this experiment is also given.

of the scene but not the outer walls of the room. For the objects around the center, overlap is usually present after only a short movement of the robots. However, for the walls it is frequent that one robot will for example see the right wall and another one the left wall of the room at the beginning of the task. Such, the walls present in the local maps should not be aligned to one another although they might look quite similar from the local views of the robots. To this end, they are blended out by the definition of the region of interest. Additionally, limiting the merging process to the region where overlap is the most probable significantly reduces the computation time of the algorithms presented in the following, as it increases linearly with the number of pixels to treat. The region of interest, as shown in figure 3.12, was defined as a square around the center with a side length of around 5.7 m.

In the following sections, two methods for occupancy grid map-merging, based on the approaches presented in section 2.4.2 and adapted to the circular topology with a common center point are presented and compared.
The map-merging methods pairwisely align respectively two local maps. In the case of this project, the experiments have been realized with three robots. Such, two maps are merged in a first place, and the third one is then aligned to the superposition of the first two. A fusion is accepted as valid only if the two stages succeeded. Often, the success of the operation depends on the order, in which the local maps are aligned. Basically, each robot has three possible merging orders, noting itself as *rob1* and the other two robots as *rob2* and *rob3*:

$$\begin{aligned} &(\text{rob1} \leftarrow \text{rob2}) \leftarrow \text{rob3} \\ &(\text{rob1} \leftarrow \text{rob3}) \leftarrow \text{rob2} \\ &\text{rob1} \leftarrow (\text{rob2} \leftarrow \text{rob3}) \end{aligned} \tag{3.10}$$

with robX $\leftarrow$ robY being the merging operation aligning the map of robot Y to the map of robot X.
In theory, the merging of two maps should be a commutative operation, i.e. robX $\leftarrow$ robY should find the same relative rotation as robY $\leftarrow$ robX (up to the angle sign). However, practical experiments showed, that because of imprecisions due to the discrete nature of the occupancy grid maps and due to the discretization of the space of possible rotations, this is not always the case. This multiplies the number of possible merging orders by 4, resulting in theoretically 12 different sequences to test. Nevertheless, experiments also showed that commutativity holds in the majority of cases, such it was decided to only test the three merging orders as presented in (3.10) per robot, to avoid unnecessary calculations.
The general approach of the map-merging framework developed during this project is thereby to calculate several merging hypotheses based uniquely on the region of interest in the center of the scene as presented in figure 3.12 and then to verify theses hypotheses on the entire maps. Such, merging hypotheses are only accepted if they are also globally consistent (in particular w.r.t. the walls of the room or other outlying objects).
The following sections will detail the two developed map-merging methods, beginning with a direct optimization approach similar to [4] and [22],[23], followed by an approach based on the Hough-Spectrum as presented in [7].

### 3.3.1 Direct Optimization Method

The direct optimization method finds the rotation $\theta$ between two local maps $m_1$ and $m_2$ by optimizing a similarity measure $S(m_1, m_2)$ between the maps. As has been laid out before, the optimization is done only on a region of interest around the center of the scene (cf. fig 3.12). We will note $roi_i$ the region of interest of map $m_i$. The research has to be executed only on a one-dimensional parameter space, as the only unknown is the rotation $\theta$, the relative translation being known because of the shared center point. As the parameter space to be searched is very

limited, it was decided to just implement a brute force, exhaustive research of all possible rotation angles $\theta \in [0^\circ, 360^\circ)$ in a given resolution. The ROI $roi_2$ is rotated by each of the possible angles $\theta$ and at each step, the similarity measure $S\left(roi_1, roi_2^*\right)$ between the rotated region of interest $roi_2^*$ and the area $roi_1$ is calculated. The rotation angle $\hat{\theta}$ that results in the maximal similarity measure is selected as the alignment hypothesis. Due the the exhaustive nature of the research, the gradients of the similarity measure $S$ are not important to guide the search algorithm, the global maximum can be found right next to a local minimum. Also, given the precision of the utilized laser-scanners and such the precision of the occupancy grid maps, as well as their discrete nature (one cell represents a 5cm x 5cm area in physical space), the meaningful angle resolution is not very high and the exhaustive research remains computationally tractable, also for frequent re-calculations of the alignment during the operation.
The optimal angle is calculated in two steps: first with a resolution of 1 degree on the entire angle range of 360°, then with a resolution of 0.1 degree on a region of +-1 degrees around the optimal result of the first step. This results in 360 + 20 = 380 rotations and similarity measures to calculate on a region of interest for the alignment of two maps. The great majority of processing time ( 95%) is spent on the first optimization the refinement needs only about 5% of the operations.

In a first place, the normed cross correlation between the two regions was taken as similarity measure $S$. It was calculated on binary images of the maps, keeping only occupied cells in white and setting the remaining ones to zero (cf. middle image of figure 3.6). Such, only overlapping occupied cells add to the similarity measure, free space and unknown cell are ignored. Merging hypotheses were calculated in several orders as presented in (3.10). A threshold value on the correlation coefficient was set to decide whether a merge was accepted as valid or not.

A simple experimental set-up was created to evaluate the map-merging methods (see fig. 3.13). Three robots were placed around a scene and then manually displaced around their starting point on a circle. Maps were saved at discrete points in time, noting the displacement of the robots on the circle. The merging was then calculated offline.

Already the simple cross correlation on binary images was able to find alignments between maps and to decide upon their validity or not. However, comparatively many overlapping cells are needed. The first valid alignments for two different data sets are shown in figure 3.14 for this method. For the first scene, a successful merge could be achieved after a displacement of the robots of +-28 degree around their starting point, for the second scene +-40 degree were necessary. Each robot is displayed on its own colour channel in the RGB-image. Only the occupied cells are taken into account for this method. The robots need to move a certain length on the circle perimeter to sufficiently build their local maps. Before, no successful merge will be possible. After a sufficient overlap for a first successful match has been reached, there can still occur errors for certain orders of the merging that need to be sorted out by the verification criteria. Those are mostly errors of 180° due to the symmetry of the scene because of the two pillars present in the room. Them being part of the building, this could however not be changed during the experiments.

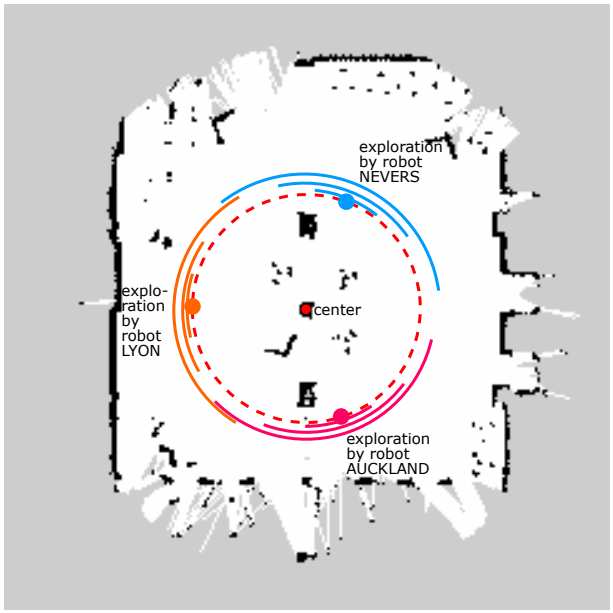



**Figure 3.13:** Setup of the experiments to evaluate the map-merging: Three robots were placed with a distance of about 120° on the same circle around the scene. They are then displaced incrementally around their starting point (marked by the coloured arcs). The radius of the robots rests the same, the coloured segments are displaced only for clarity of appearance of the figure.

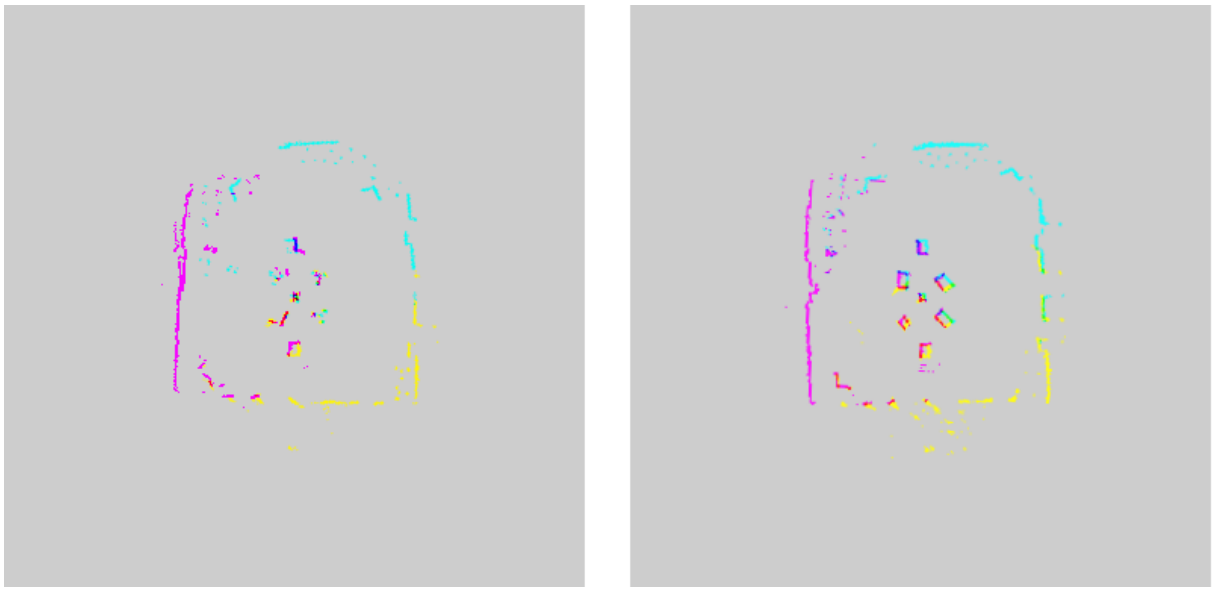


**Figure 3.14:** The first valid merges of the cross-correlation method, tested on two data sets. The experiment space contains two pillars, that are present in all data sets. In the first scene (to the left) three chairs, of which only the legs are visible to the laser scanner and one overturned chair are present. In the second scene (to the right) four rectangular boxes are present. For the first scene, a successful merge could be achieved after a displacement of the robots of +-28 degree around their starting point, for the second scene +-40 degree were necessary.

In order to improve the merging results, a different similarity measure was searched. The method retained for the rest of the project is a modified version of the Similarity Index (2.21) introduced by Birk and Carpin in [4]. This Similarity Index is calculated making use of all three states of the occupancy grid map. Birk and Carpin define their Similarity Index by counting *agreement* and *disagreement* between two maps. Taking two occupancy grid maps $m_1$ and $m_2$ with the possible states *free*, *occupied* and *unknown*, the term $\mathrm{agr}(m_1, m_2)$ is defined as the number of cells where both maps indicate *free* or both maps indicate *occupied*. The term $\mathrm{dis}(m_1, m_2)$ is defined as the number of cells where one map indicates *free* and the other one *occupied*. Cells where either of the maps indicate *unknown* are ignored in the count.
For our context, we define differently the *agreement* between the maps. The term $\tilde{\mathrm{agr}}(m_1, m_2)$ is defined as the number of cells where both maps indicate *occupied*. Cells where both maps indicate *free* are ignored. This modification was necessary, as there are always significantly more free cells overlapping than occupied ones in our case. Such, counting overlapping free cells hides the effect of a small number of occupied cells overlapping or not. Yet it is the occupied cells that define contours of objects and such it is them that need to be brought to alignment. This modification permits to use the Similarity Index directly as the similarity measure $S$ to optimize during the exhaustive research. It is defined as

$$\tilde{\omega}\left(m_1, m_2\right) = \frac{\tilde{\mathrm{agr}}\left(m_1, m_2\right)}{\tilde{\mathrm{agr}}\left(m_1, m_2\right) + \mathrm{dis}\left(m_1, m_2\right)}. \tag{3.11}$$

To find and verify an alignment between three maps, merging hypotheses are found for the three merging orders presented in (3.10) by optimizing the Similarity Index $\tilde{\omega}$ on the regions of interest of the respective combinations. First, two maps are aligned, and the third one is then aligned on the superposition of the two previously aligned ones. To superpose two maps, the choice was made to give priority to occupied cells. If either of the two maps indicates a cell as occupied, it will be occupied in the superposition. A cell will be indicated as free only when both maps indicate free, otherwise it will be unknown.
After the merging hypotheses have been calculated, they need to be verified. To this end, the Similarity Index $\tilde{\omega}$ is recalculated for each hypothesis, this time for the entire map. An alignment is accepted, if $\tilde{\omega}$ is greater than a threshold, indicating a globally coherent alignment. In particular, $\tilde{\omega}$ calculated on the whole maps will be distinguishably low if a wall in one map falls into free space of another one, leading to a high *disagreement* value. To verify that the alignment is coherent for all three maps, $\tilde{\omega}$ needs to be calculated for three combinations:

$$\begin{aligned}
\nu_{12} &= \tilde{\omega}\left(m_1, m_2^*\right) \\
\nu_{13} &= \tilde{\omega}\left(m_1, m_3^*\right) \\
\nu_{23} &= \tilde{\omega}\left(m_2^*, m_3^*\right)
\end{aligned} \tag{3.12}$$

with $m_2^*$ and $m_3^*$ the maps of the other robots aligned to $m_1$ as proposed by the respective merging hypothesis. If all three values are above the threshold $\nu_{min}$ the merging is accepted. If several merging orders lead to a successful alignment, the hypothesis with the highest sum of the three verification indices is selected.

The method based on the modified Similarity Index was tested on the same data set as the correlation-based method. The first successful merge found is shown in figure 3.15 for the same two data sets used for the evaluation of the correlation-based algorithm. For the first scene, a successful merge could be achieved after a displacement of the robots of +-17 degree around their starting point, for the second scene +-11 degree were necessary. This is significantly sooner than with the correlation-based approach Such, the number of overlapping cells needed for a first successful merge could be greatly reduced. As for the first approach, errors mainly of 180° mis-rotation between two maps can occur. Those need to be sorted out by the verification criteria.

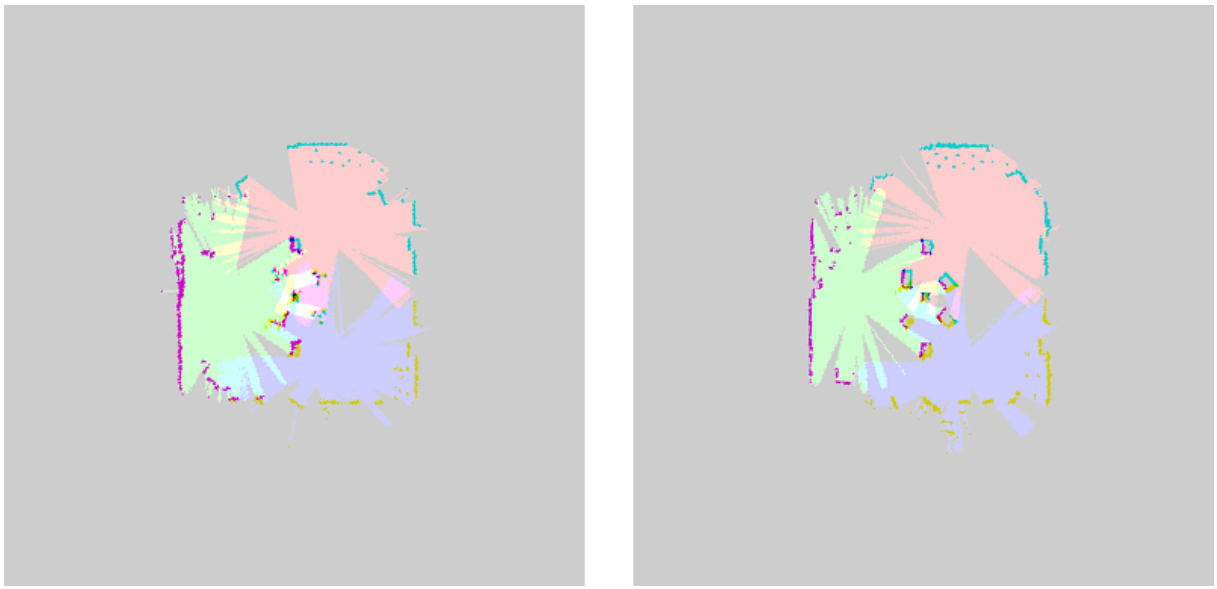


**Figure 3.15:** The first valid merges of the Similarity Index method on the same data-set used in figure 3.14. For the first scene, a successful merge could be achieved after a displacement of the robots of +-17 degree around their starting point, for the second scene +-11 degree were necessary.
Each robot is shown on a colour channel. Occupied cells have darker colours, free ones lighter colours. magenta: occupied robot Lyon, light-green: free robot Lyon. dark-green: occupied robot Nevers, light-red: free robot Nevers. yellow: occupied robot Auckland, blue: free robot Auckland. grey: unknown to all robots. Other colours result from superposition of the previous values.

The results presented until now have been calculated offline on saved maps during manual movement of the robots. After this first verification, the direct optimization method using the modified Similarity Index has been implemented for online map-merging working together with the navigation and communication framework presented in section 3.2. The map alignments are now searched in parallel on all three robots while they are moving around the scene. The movement pattern was changed compared to the offline experiments: The robots don't move to both sides of their starting points (cf fig. 3.13) but only in the counter-clockwise direction. Each time a new map arrives from the gmapping SLAM algorithm of the robot itself or from one of the other robots over the network, knew merging hypotheses are searched using the three

possible orders (3.10). The most probable, valid one is then selected.

Several results of the online merging experiments are shown in figures 3.16 and 3.17.
3.16 shows the first successful merges for the three robots Lyon, Nevers and Auckland participating in the task for two different experiments. Several observations can be made in comparison to the offline experiments:

During the online experiments, a longer robot movement and such more overlapping pixels are needed for a first successful match. However, as maps are continuously merged, it is difficult to determine the exact angle of displacement. The maps can only be compared visually by the size of the different coloured regions, marking the local maps of each robot used for this merge.
The increase in required overlap is mostly due to imprecisions in the center coordinate. The center point is currently given to the robots by the operator via a graphical interface. Its precision is therefore limited, it might not exactly mark the same physical point in the maps of each robot. As the initial assumption was exact shared knowledge of a common center points for all robots, the translation part of the map alignment is completely ignored in the current implementation of the framework. However, an exact alignment by searching only a rotation is not possible with an imprecisely given center and the number of pixels that can be brought to overlap will be sufficiently reduced. Such, larger local maps are necessary to find enough overlap to successfully compute a map alignment.
Furthermore, because of those imprecisions, that were more significant during the online experiments than the before conducted offline ones, the verification conditions had to be modified. Besides the threshold on the similarity indices between the rotated maps 3.12, a condition on the absolute number of agreeing or disagreeing pixels has been added. In fact, the sum of agreeing and disagreeing pixels gives the number of overlapping pixels that are not unknown or free in both maps, and such the number of pixels on which the verification is based. If this number is too small, a hypothesis has to be rejected in in any case, because the thresholds on the similarity values are no longer significant if they are based on too few pixels. This modification favours the false rejection of of good matches (false negatives) over the erroneous acceptance of false alignments (false positives). The criteria were tuned to accept almost no false positives on the saved datasets, however a significant number of false negatives may occur, especially if the center point is given with imprecision. However, this seems the best acceptable choice in practice, as a falsely accepted alignment might seriously disturb the robot's navigation as well as its localization of the other robots. A false reject has less serious consequences, especially when a first good alignment has already been found. This can just be maintained, until the next good alignment is accepted, several rejections between two correctly validated alignments have no consequences.
Such, with the modified verification criteria, the map-merging algorithm could be adapted to provide reliable alignment hypotheses and verification during multiple online scenarios.

Another observation of figure 3.16 is, that the first accepted merges don't happen after the same lengths of movement on a circle (and such not at the same point in time) for all the robots. The sizes of the local maps used for the first merge are sometimes different between the robots, as

can be distinguished by the different sizes of the coloured regions.
There are two explanations for this observation: Firstly, it might be due to different merging-orders and the fact that the operation is not always commutative, as has been observed during the offline experiments. Secondly, and this probably is the more significant factor, it might be due to delays in the communication network, as the occupancy grid maps actually aren't sent at a very high frequency (once every two seconds) so as not to overcharge the WiFi network. The robots don't communicate successful alignments between them for the moment, but are calculating the merging hypotheses independently from one another. However, as becomes apparent in figure 3.17, even at the current stage, all robots converge towards an almost identical global map with enough displacement. A sharing of merging hypotheses could be added in a future version of the framework, to enhance robot cooperation and to ensure that they are all working with the same global map representation.

Figure 3.17 shows the joint maps of the three robots at the end of the task the first merges of which are shown in the bottom line of figure 3.16 after one tour around the scene. A globally coherent alignment, also with respect to the walls and objects farther away from the scene has been reached for all three robots. However, the walls often don't overlap exactly. This is due, as discussed before, to imprecisions in the center coordinate. Such, not all pixels can be brought to overlap by only searching a rotation between maps.

Globally, it could be verified during multiple experimental setups that the proposed online map-merging framework, together with the circular navigation framework, provides reliable performances for the joint mapping task. A first, correct merging hypothesis can usually be achieved with the robots having moved less than a quarter of a circle (cf fig 3.16) giving them a global overview of the scene only a small time (usually 20 - 30s) after the beginning of the task. False alignments, that consist mostly of errors of $180^\circ$ between two maps due to the symmetry of the room where the experiments were conducted, are reliably rejected by the verification step. Even if there is a number of rejections of correct merges, this is not a serious problem after a first alignment has been found, that will be maintained in this case.
The computational performances of the algorithm on the limited netbook-hardware of the robots are suited for a frequent recalculation of the merging hypotheses upon the arrival of new, more complete local maps from the team-mates or from the SLAM-algorithm of the robot itself. Calculation of a merging hypothesis takes around 120ms. A complete merging cycle, calculating respectively two alignments in three different orders takes such around 720ms.

### 3.3.2 Hough-Spectrum based Method

The second approach to occupancy grid map-merging analysed in this project is based on the Hough transform. The algorithm presented in [7] that has been resumed in section 2.4.2 has been reimplemented, retaining only the part determining the rotation between two maps. Figure 3.18 illustrates the different steps of the method. As for the direct optimization method, alignment hypotheses are computed from a region of interest around the center of the observed

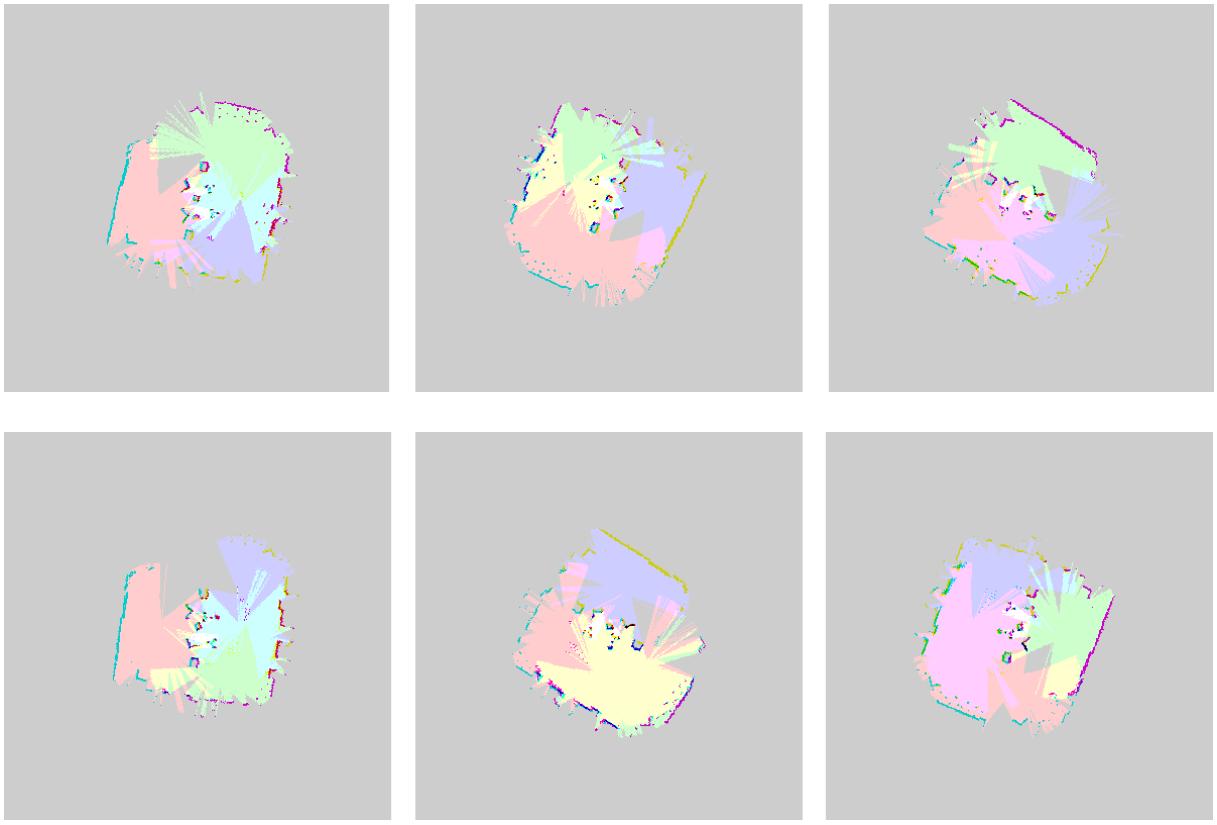


**Figure 3.16:** First successful merges during two of the online map-merging experiments for the three robots Lyon, Nevers and Auckland (from left to right). Colour schemes are the same as in fig 3.15, however the color-channels are interchanged between the robots. light-green, light-red and blue indicate free space seen by resp. one robot. yellow, rose and light-blue indicate free space seen by resp. two robots. White indicates free space seen by all three robots.

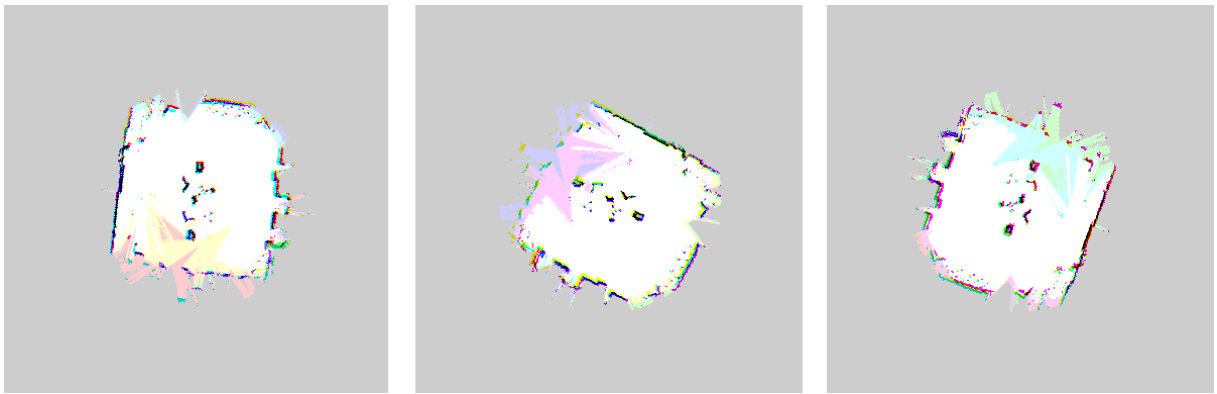


**Figure 3.17:** Final merged maps after each robot moved almost a complete circle around the scene for the three robots Lyon, Nevers and Auckland (from left to right). Colour schemes are the same as in fig 3.15, however the color-channels are interchanged between the respective robot's point of view. Here, much more overlap is present. White areas are free for all three robots, black cells are occupied for all robots.

scene, as illustrated in figure 3.12. In a first step, the Hough transforms of the two binary input images, representing occupied cells as white pixels are computed. They are then reduced to their respective Hough-Spectrum representation as introduced in [7] (see equation 2.24). This representation retains only the angular dimension of the Hough transforms. Peaks in the Hough-Spectrum represent probable lines with the corresponding angle of their normal towards the $x$-axis of the plane. To find the relative rotation between two maps, the cross-correlation between the two spectra is computed. As the Hough-Spectrum represents the orientation of probable lines in the input image, the spectrum, as well as the cross correlation, are $180^\circ$ redundant. Such, we will find at least two symmetric peaks, at a distance of $180^\circ$ in the correlation between the spectra (cf. bottom line of fig. 3.18). In many, less evident cases, more than two local peaks will be present.
All local peaks of the cross correlation of the two Hough-Spectra will be retained as possible relative rotation angles. To decide upon the angle to retain, the modified verification index $\tilde{\omega}$, as defined in equation 3.11, will be computed between the entire maps, the second map rotated by each of the possible angles. The angle that results in the maximal similarity $\tilde{\omega}$ will be retained as the rotation hypothesis to align the two maps.
As for the method presented in section 3.3.1, the goal of the overall framework was to align maps of three robots. Identically to the previous method, several merging orders for three maps will be tested, each sequence resulting in a hypothesis of possible alignment between three maps. To check a coherent alignment between all three maps, a verification step is added, identically to the procedure presented in section 3.3.1 for the online algorithm. The modified Similarity Index is computed for the entire maps, rotated by the possible relative angles. A combination of thresholds on the similarity measures and the number of overlapping, not unknown pixels are used for verification of a globally coherent alignment. Theses criteria are exactly the same between the direct optimization and the Hough-Spectrum alignment.

The Hough-Spectrum method has been evaluated on the same two offline datasets as the correlation- or Similarity Index based implementations of the direct optimization method (figures 3.14 and 3.15). Especially for the first data set, containing only few rectangular objects, a long movement of the robots of +-63 degree around their starting point is necessary before a first merge can be found. For the second data set, +-40 degree are necessary, this is the same as for the correlation based method, however, for both data sets, the robot movement necessary before finding a first valid match is considerably larger than for the Similarity Index based direct optimization method. This observation was confirmed by tests on further datasets, reusing the maps saved during the online experiments. The Hough-Spectrum method has only been tested offline, on saved maps of experiments and not online during the experiments themselves. An online implementation can however simply be derived from the existing one.
The Hough-Spectrum method is computationally faster than the Similarity Index optimization: For a data set resulting in the computation of 768 alignments between to maps, the Hough-Spectrum method needed around 10s to finish while the Similarity Index method took around 30s on the same computer. However, the Hough-Spectrum method calculates the rotation angle only with a precision of 1 degree. The optimization of the Similarity Index is calculated also

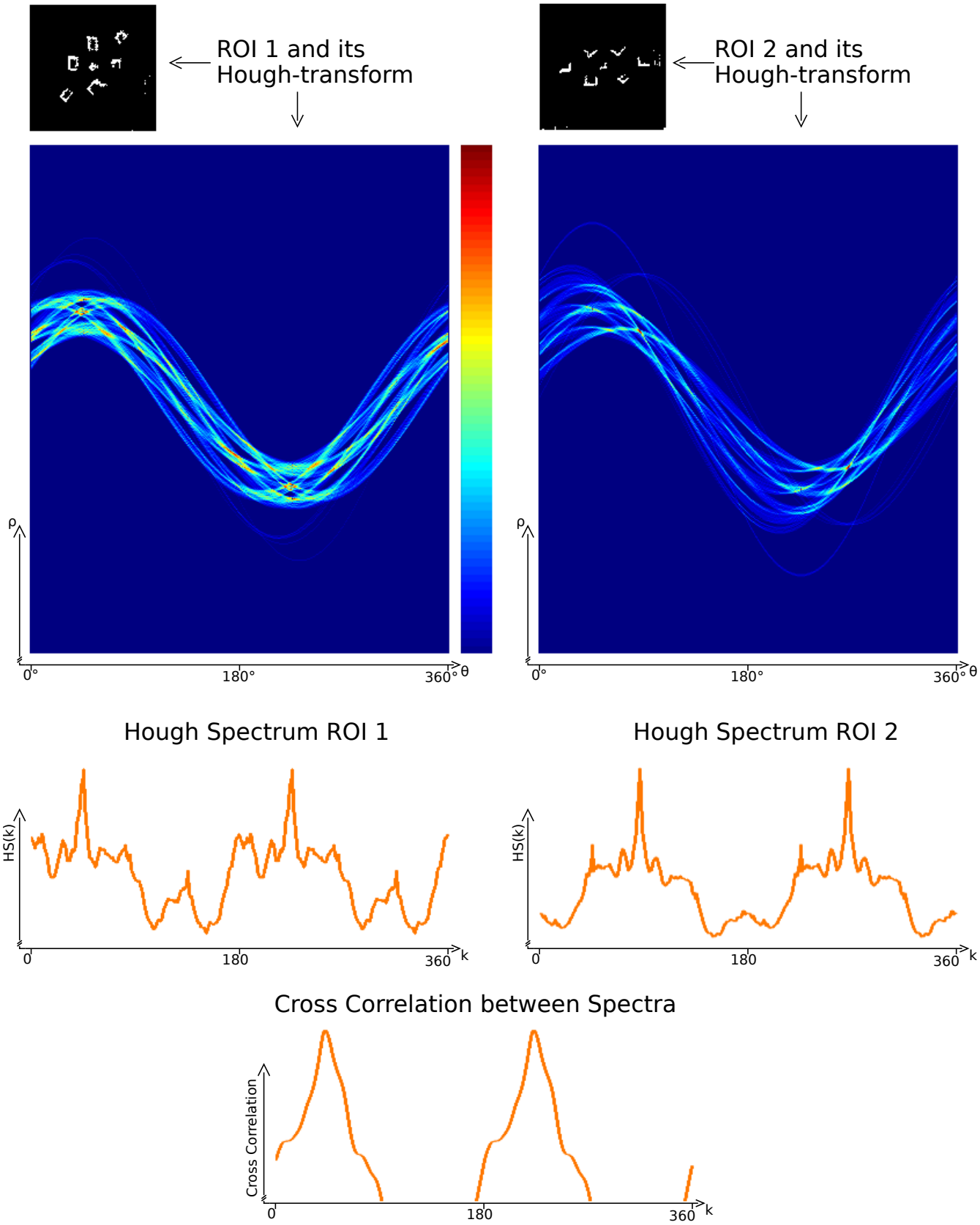



**Figure 3.18:** Illustration of the Hough-Spectrum based map alignment. On the top the two regions of interest to align. Below, their respective discrete hough transforms. The values of the accumulators for the angle and distance dimensions $\theta$ and $\rho$ are indicated by the color-map shown between the two figures. Blue represents low values and red higher ones. In the third line, the respective Hough Spectra, as defined in equation (2.24) are shown. At the bottom line, the cross correlation between the two spectra is given. A $180^{\circ}$ symmetry is apparent.

with a precision of 1 degree in a first place, but then a refinement step is added, recalculating the optimization with a resolution of 0.1 degree in a region of +-1 degree around the previously found max. The resulting resolution is such of 0.1 degree and therefore finer than that of the Hough-Spectrum based approach. However, as laid out in section 3.3.1, the great majority ( 95%) of the processing time of the optimization algorithm is spent on the first step with a resolution of 1 degree. Such, the above example still gives a good indication of a performance comparison between the direct optimization and the Hough-Spectrum algorithms. The latter is about 3 times faster.

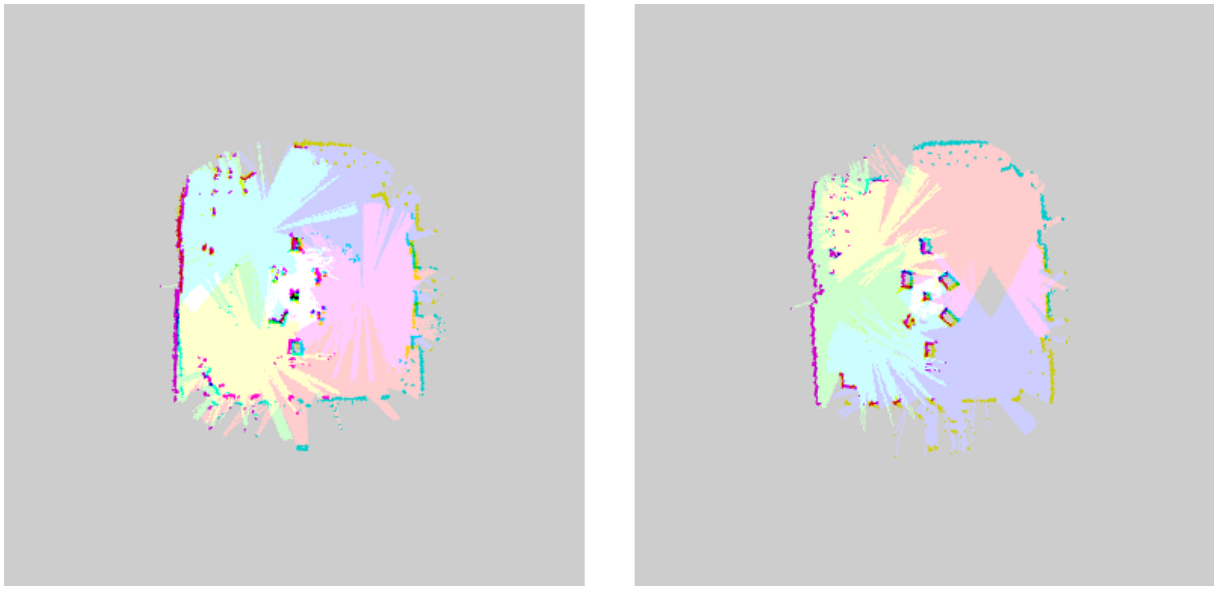


**Figure 3.19:** The first valid merges of the Hough-Spectrum method on the same data-set used in figures 3.14 and 3.15. For the first scene, a successful merge could be achieved after a displacement of the robots of +-63 degree around their starting point, for the second scene +-40 degree were necessary.
The color scheme is the same as in figure 3.15.

The following section will further compare the two presented map aligning methods with a focus on their dependence on the quantity of objects present around the scene.

### 3.3.3 Influence of the Number of Objects and Symmetry

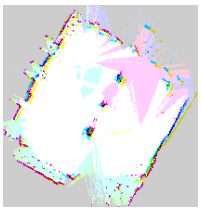
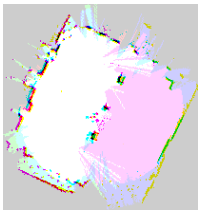
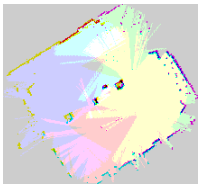
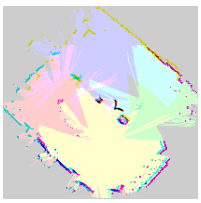
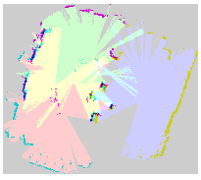
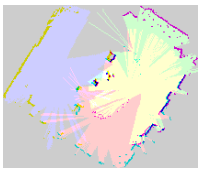
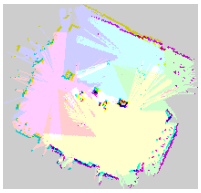
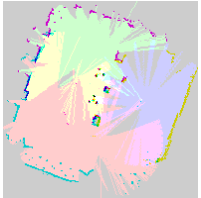
The previously described experiments have all been conducted with a rather dense scene: four or five objects were present around the center. They were approximately equally distributed to each side of the scene.

To examine the dependence of the map-merging on the number of objects present in the scene and on symmetries in the arrangement of the objects, several experiments have been realized. The results are summarized in table 3.1. The table shows the results of a number of online experiments, with the map-merging based on the optimization of the Similarity Index running

continuously in parallel to the circular navigation and communication framework. The navigation task was for each robot to make a complete circle of the scene. Three robots participated in each task.
Table 3.1 gives a description of the respective scene set-up and an image of the first correct match found during the experiment in its first two columns. The images show the first match only for one of the three robots participating in the task, however, the first matches were similar between all three robots. The images give a visual impression of the objects present as well as an indication of the amount of displacement necessary for the robots to find enough overlapping objects to align their local maps. How far the robots had to move can be distinguished by the size of the different coloured regions and especially the amount of overlapping free space. The colour scheme is the same as in figure 3.16. Light-green, light-red and blue indicate free space seen by resp. one robot. Yellow, rose and light-blue indicate free space seen by resp. two robots. White indicates free space seen by all three robots. Occupied cells are represented by darker colours.
Different numbers of objects were present in the scene during the experiments. Two pillars that are part of the experiment space are always present. At first, no other object was added, resulting in a perfectly symmetric scene. Next, different single objects were examined: A chair of which only three legs and a center point are visible to the laser scanner. A small box, resulting in a tiny but continuous object, as opposed to the chair that only yields some discrete points in the map. And lastly an overturned chair resulting in a rather large, continuous object. Then, two objects in a more or less symmetric arrangement were analysed. Lastly, a scene with three objects was studied. For each experiment, table 3.1 states the number of matches found and accepted as valid by all three robots participating in the task. This number was similar between the respective robots if not exactly the same. Furthermore, the number of false alignments erroneously accepted (i.e. the number of false positives) is stated. As for the accepted matches, the sum of false positives of all three robots is indicated. The individual numbers again were similar between the robots. The quotient of these two numbers indicates the percentage of false validation. Lastly, the time between the robots starting to move and the first correct match found is shown in the table. The average between the three robots is given. There was no great difference between the individual times. Unfortunately, because of an error during the logging of the experiments, the time to a first match can not be indicated for two scenes. However, the images of the first match allow for a visual comparison.

The first two scenes containing no additional object resp. one chair resulting in a small number of non-connected points, pose problems for the map-merging. The symmetry inherently present because of the two pillars leads to frequent false alignments. The few points the chair leaves in the laser scan are not sufficient to compensate for the symmetry of the pillars. For both scenes, the rate of false validation is about 50%. Also, a fist valid match is found only after a large displacement of the robots (around 150s after the beginning of the movement). For the first scene, only one robot finds a correct alignment at all, the other two only find false ones. For the second scene, all three robots eventually find a good match. A comparison of the images of the first alignments shows that the added chair allows for a match after slightly less movement of the robots (smaller white region indicating overlapping free space of three robots). The time to a

**Table 3.1:** Results of online merging experiments with a varying number of objects present in the scene. The number of accepted matches and false positives are given as the sum between all three robots participating in the task. Time to first match is the average between the robots.

| Scene | First Match | Accepted Matches | thereof False Positives | % False Validation | Time to First Match |
|---|---|---|---|---|---|
| **1** - no object |  | 152 | 84 | 55.3 | 127s |
| **2** - one chair |  | 314 | 173 | 55.1 | 155s |
| **3** - small box |  | 208 | 4 | 1.9 | n.A. |
| **4** - overturned chair |  | 258 | 0 | 0.0 | n.A. |
| **5** - chair and small box |  | 182 | 0 | 0.0 | 12s |
| **6** - chair and small box - symmetric |  | 388 | 58 | 14.9 | 20s |
| **7** - two chairs |  | 431 | 129 | 29.9 | 46s |
| **8** - two chairs and small box |  | 338 | 3 | 0.9 | 31s |

first match is somewhat larger but this may be due also to other experimental conditions that could not all be taken into account. Compared to the other scenes, these first two pose a large problem as there is an insufficient number of objects present.
In the third scene, a small, continuous object (i.e. a small cardboard box) suffices to compensate the 180° ambiguity of the pillars present in the room. The error rate is drastically reduced to only 2%. A first merge can be found sufficiently sooner than for the first two scenes. Even though the time is not available for this experiment, this can be concluded from a visual comparison of the first matches. In the fourth scene, a larger, continuous object was present (i.e. a overturned chair). The error rate is reduced to 0. As can be distinguished by a visual comparison, the first match is reached slightly sooner than with only the small box present. A single, asymmetrically placed object, being present as a set of spatially continuous, occupied cells in the maps, such suffices to provide enough overlap for a map alignment and to compensate the ambiguities that are due to symmetries in the scene.
Next, some experiments were conducted with two objects. A chair of which only the legs were visible and the cardboard box were present in the scene. Firstly, an asymmetric arrangement with both objects to the same side of the center was analysed (scene 5). The error rate remains zero and a first match can be found with even less robot movement than during the previous two experiments. The yellow, light-blue and rose regions indicating overlapping free space between two robots are significantly smaller than for the first matches of scenes 3 and 4. Here, the time to a first match could also be measured, it was around 12s in average for the three robots. Then, a 180° symmetric arrangement of the two objects was examined (scene 6). Here, it is significantly more difficult for the robot to decide between a false and a good match, as a 180° error may also lead to overlapping pixels. (although in this case they don't belong to the same object). The error rate of erroneous validation of false alignments raises to around 15 %. Also, it takes somewhat longer to find a first correct match, 20s in average between the robots.
Scene 7 illustrates the difficulty that pose objects that are only visible as a small number of discrete, not connected points to the laser scanner (i.e. legs of chairs). Even an asymmetric arrangement of two chairs leads to an error rate of around 30%. A first match can be found after around 46s. Such, more robot movement is needed compared to the scenes 5 and 6 to correctly align the maps, as can be verified by a visual comparison of the first matches found. These difficulties can be largely compensated by adding the small box, resulting in a continuous object in the maps. For scene 8, the error rate goes back to approx. 1% and the time for a first match decreases to around 30s.

In a second place, the Similarity Index direct-optimization method and the Hough-Spectrum based method were compared. To ensure that the comparison takes place on the same data set, the local maps saved during the previously described experiments were re-analysed by offline versions of the two merging algorithms. A relevant subset of the saved maps was chosen, as the data set contained many redundant items because the local maps were saved with a relatively high frequency during robot movement. From all the saved maps, predominantly incomplete local maps from the beginning of the task were kept, to analyse the map alignment in its more difficult cases. Also, not all of the saved maps have been tried to align during the online experiments, as

there were considerable delays present due to the communication between the robots. Thus, the offline merges don't result from the same local maps that have been merged during the online task and it is difficult to compare online and offline version of the direct-optimization method. However, the Similarity Index and the Hough-Spectrum methods can be compared among one another on the offline data set.

**Table 3.2:** Offline Merging results comparing Hough- and direct optimization method. A subset of the local maps saved during the experiments analysed in table 3.1 was processed for these statistic.

| Scene | Hypotheses computed | Algorithm | Accepted Matches | thereof False Positives | % False Positives | % True Positives |
|---|---|---|---|---|---|---|
| **1** - no object | 312 | Sim. Idx. | 19 | 19 | 6.1 | 0.0 |
| | | Hough | 2 | 0 | 0.0 | 0.6 |
| **2** - one chair | 372 | Sim. Idx. | 93 | 48 | 12.9 | 12.1 |
| | | Hough | 105 | 28 | 7.5 | 20.7 |
| **3** - small box | 456 | Sim. Idx. | 131 | 46 | 10.1 | 18.6 |
| | | Hough | 141 | 32 | 7.0 | 23.9 |
| **4** - overturned chair | 384 | Sim. Idx. | 93 | 0 | 0.0 | 24.2 |
| | | Hough | 68 | 0 | 0.0 | 17.7 |
| **5** - chair and small box | 528 | Sim. Idx. | 120 | 0 | 0.0 | 22.7 |
| | | Hough | 71 | 9 | 1.7 | 11.74 |
| **6** - chair and small box - symmetric | 600 | Sim. Idx. | 168 | 60 | 10.0 | 18.0 |
| | | Hough | 162 | 31 | 5.2 | 21.8 |
| **7** - two chairs | 504 | Sim. Idx. | 120 | 43 | 8.5 | 15.3 |
| | | Hough | 93 | 37 | 7.3 | 11.1 |
| **8** - two chairs and small box | 696 | Sim. Idx. | 115 | 8 | 1.1 | 15.4 |
| | | Hough | 43 | 7 | 1.0 | 5.17 |

Table 3.2 shows the results from the comparison of the Similarity Index and Hough-Spectrum based map-merging approaches on the same offline dataset. The scenes analysed are the same as in table 3.1, however, as explained above, the subset of local maps analysed is not equal to the combinations tried during the online task.
For the offline datasets, the total number of merging hypotheses computed can be indicated. The rates of false positives and true positives are computed with respect to this number. The global tendencies rest the same as for the online analysis: Scenes 1, 2 and 7 with no continuous objects and scene 6 with a symmetric arrangement are problematic. They result in a rate of false positives around 10% and in a rate of true positives between 0 and 20%. Both algorithms work well for scenes with continuous objects (4, 5, 8). The rate of false positives stays between 0 and 1%. The rate of true positives never is significantly higher than 20%. This is due to the

verification criteria, which, as was explained before, were tuned to accept very few false positives. The trade-off is a high number of false negatives - not accepted good matches - those, however, are a lot less problematic than falsely accepted incorrect matches. An exception is scene 3 that actually seems a lot more difficult in the offline analysis (table 3.2) than in the online one (3.1). This is mostly due to the choice of the subset of saved maps analysed in table 3.2. Even if no exact conclusion can be drawn, it shows, that the small box as only continuous, asymmetrically placed object is at the limit of sufficient overlapping points for a successful map alignment.

To compare the Similarity Index and the Hough-Spectrum method, the latter seems to work better for the more difficult scenes 1, 2, 3 and 6. It results in a lower rate of false positives and a higher rate of correctly accepted alignments. For the "easier" scenes with larger continuous objects 4, 5 and 8 and almost no false positives, the Similarity Index method performs better, resulting in a higher percentage of true positives. Especially for scene 5 the rate almost doubles compared to the Hough method. For Scene 7, the results are ambiguous: The Similarity Index method results in a higher number of true positives but also in more false positives compared to the Hough-Spectrum method. The increase in true positives, however, is higher than the increase in false positives.
It is difficult to draw a general conclusion. This may be due to the choice of the subsets of maps analysed or to other factors that influenced the experiments and have not been taken into account. Further analysis, especially also using an online version of the Hough-Spectrum method, needs to be done here.

As a conclusion, the direct optimization algorithm based on the adapted Similarity Index (3.11) provides very reliable results during real-world experiments, running online during a robot task, if there is at least one continuous, asymmetrically placed object present in the scene. Symmetries and objects that result only in a small number of unconnected points in the occupancy grid maps (i.e. chair legs) pose problems. However, the problems of the chair legs solely persists if there are exclusively objects of this kind present. Adding even a small continuous object like a cardboard box resolves the difficulties. Further research has to be conducted to compare the direct optimization of the Similarity Index with the Hough-Spectrum method, also during online experiments.

# Chapter 4

# Conclusion and Future Work

During this student research project, a framework for robot navigation and communication on a circular, concentric topology has been developed. A key-point of the project was the realization of a cooperative robot task, a team of multiple robots jointly constructing a map of the environment.
The framework was developed using the common Robot Operating System (ROS) libraries and experimentally verified on low-cost robot hardware. The starting point of the project was the notion of a circular navigation topology, the robots evolving on concentric circles around a scene with a human person at its center, performing an activity that is analysed by the robots. Realistic assumptions concerning the initial robot position were taken: it is not known to the robots at the beginning of the task.
The displacement of the robots was realized by two simple operations: A movement along a circle perimeter with constant radius and a movement towards or away from the scene to change the radius of the circle. These movements were realized by a simple feedback control approach using the localization information of an underlying SLAM algorithm. During their circular displacement, the robots continuously construct a local map or the environment from laser range scan data, using a standard particle filter SLAM algorithm, the *gmapping*-package from the ROS libraries. Obstacle recognition and avoidance functions were added to the framework, based on the local maps and the laser scan data. To realize the cooperative task, a communication framework was developed, sharing local maps and positioning information between the robots via a WiFi network.
The concentric topology assuming a common center point shared between the robots, allows for an important simplification in the cooperative mapping task. The joint map is constructed by aligning local maps to a global one. To this end, the local maps and the respective center point are shared between the robots. An alignment between local maps is then found by searching overlapping regions in them. As the center point is known in each map, they can simply be superposed, the translation part of the map alignment is already inherently given. Contrary to common map-merging approaches ([4], [7], [30], [28]), only the rotation between the local maps has to be found. This reduces the space of possible transformations to one dimension - the rotation angle $\theta$ between two local maps. Such, the computational complexity is greatly reduces with respect to the general case of a three transformation parameters (two translations $t_x, t_y$ and the rotation $\theta$). Two map-merging approaches from literature have been adapted to the special topology and implemented on top of the navigation and communication framework

during this project. The first one is based upon the direct optimization of a similarity measure [4], the second one is based on matching of image features extracted by the Hough Transform [7]. Results for both methods are given by evaluating multiple data sets containing different types and numbers of objects present around the scene observed by the robots. The direct optimization method has been integrated into the developed framework, allowing to demonstrate a complete cooperative mapping task using three robots in a real-world environments. Three robots evolve around a scene containing multiple objects with a person to observe in its center. Local maps are continuously transmitted between the team-mates and alignment hypotheses are computed in real time during the experiments. Successful alignments are separated from erroneous ones by a verification step. Such, robots can have a global overview of the scene after only a little individual displacement and can furthermore know their relative position on which no information was given initially. The experimental evaluation shows, that the framework works reliably if their is at leas one continuous, asymmetrically placed object present in the scene.

Further research should be done to compare in more detail the map-merging approaches. The Hough-Spectrum base approach has only yet been implemented offline, on saved data sets. An online version should be tested against the direct optimization approach. Also, the verification of the global coherence of an alignment hypothesis should be reviewed. Currently, a combination of multiple criteria is used, as no unique meaningful parameter could be found. A possible approach could be to introduce a measure of distance between distinct structures (walls, obstacles, etc.). At the moment, only direct overlap of occupied areas is counted. This could be generalized using a distance measure. Objects in the maps that are close to each other but not completely overlapping could better be taken into account. Actually, as only the rotation between local maps is searched, the center point of the scene, around which the maps are rotated needs to be known with precision. At the current stage of the project it is given by the operator via a graphical interface but in a later stage it should be found by the robots autonomously. In case of imprecisions in the center coordinate, a pure rotation can not perfectly align the maps. Such, it should be considered to add a limited research of relative translation constricted to a small region around the center point. Also, the influence of the number of robots participating in the task should be analysed, as so far the experiments have all been realized with three robots.
On top of the developed framework, a gesture recognition layer and a high-level planning algorithm will be added in future. Such, the cooperative observation of a human dynamic scene by a team of multiple robots as imagined as the long-term goal of the project can be realized in practice on affordable low-cost robot hardware.

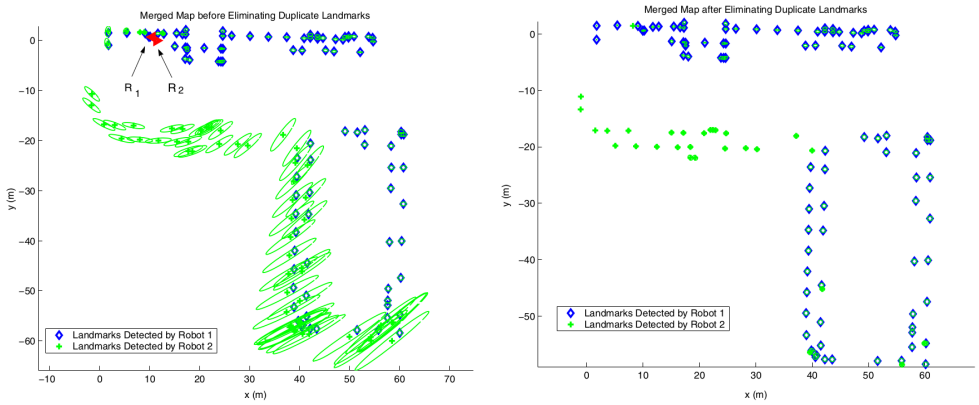
# Appendix A

# Other Architectures for Multi-Robot SLAM

## A.1 Multi-Robot SLAM based on Topological Maps

[33] presents a multi-robot SLAM algorithm for map-alignment with unknown initial relative poses on feature-based maps, using EKF-SLAM to estimate robot and landmark positions for the individual mapping processes.
Relative pose measurements upon rendezvous are taken to compute the coordinate transform between two maps (fig. 2.11). When there is overlap between the maps, landmarks that appear in both of them are used to reduce the uncertainty of the map alignment. The relative distance and bearing measurement upon rendezvous of two robots gives an initial estimate for the alignment. To this end, the robots have a very distinct cylinder mounted on top, which will be detected by an omnidirectional camera. An error and uncertainty representation of this initial transformation is computed. Especially the uncertainty in the rotation increases significantly with growing distance to the rendezvous-location. To determine duplicate landmarks, a nearest neighbour algorithm using the Mahalanobis distance is employed. As a merging decision is irrevocable, false matches will lead to inconsistency. Yet no false matches will occur when all landmarks' position errors are smaller than the distance between any two landmarks. However, this doesn't hold for large maps, as the rotational error will amplify with growing distance to the rendezvous-position. To overcome this, a solution is to match landmarks in the right order: The map is consecutively updated starting from the rendezvous-position where there is only small uncertainty in the relative transform. This will gradually reduce the error of further away landmarks and such make coherent map merges possible (see fig. A.1). After a mutual encounter, the relative positions are known and the robots will continue to augment the joint map in a cooperative SLAM process.

Another proposal, avoiding to share the whole map during the online mapping process by exchanging only the relevant portions of maps between robots, is that of *condensed measurements* presented in [21]: Here also a graph based map representation is used: nodes are poses and landmarks, the edges contain odometry measurements that constrain the connected poses.
Each robot computes its own map and refines it by virtual measurements obtained from the other robots. These condensed measurements are a reduced version of the other robots' graphs




**(a)** **(b)**

**Figure A.1:** Uncertainty of landmark positions (ellipses) before (a) and after (b) the merging of duplicate landmarks. Figures taken from [33].

containing only the information relevant to the receiving robot. Therefore, each robot's graph gets only minimally augmented. It is supposed that robot $A$ knows which nodes of robot $B$'s graph he is seeing and such it should include the information on the relative positions of these nodes from robot $B$. So as not to transfer the whole graph of robot $B$, but only the relevant part, one of those nodes is chosen as a central node and the relative position of the other observed nodes to this one is computed from the information in Robot B's graph (see fig. A.2). Only this condensed information is then transferred to robot $A$.

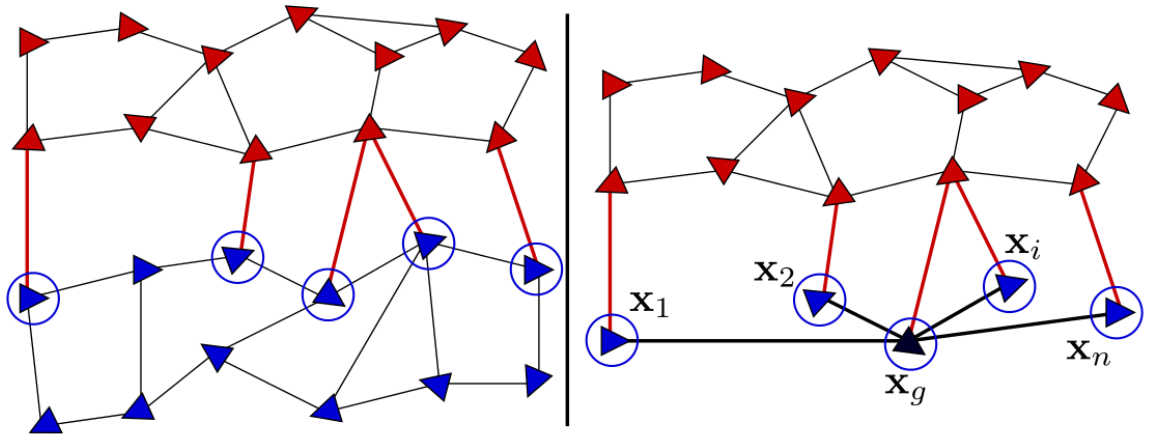



**Figure A.2:** Illustration of the condensed measurements: Graph of robot $A$ (red) and robot $B$ (blue). Robot $A$ sees the encircled nodes of robot $B$'s graph and has relative measurements (edges). Only those nodes will then be communicated, as relative positions of the nodes $x_i$ with respect to the central node $x_g$. Figure taken from [21].

A RANSAC-based approach is applied to localize a robot in another robot's graph. Each robot transmits his current laser scan and the node from which this scan was taken to the robots in communication range, as well as the positions of the last $N$ nodes. If the robots buffer the $N$

laser-scans they will be able to reconstruct other robots' local maps. Those are then used to localize the other robots in the current robot's map. The goal is to find a set of edges between the nodes of the two local maps such that they are maximally consistent. The solution of a correlative scan-matching gives a pool of candidate edges, the selection of the edges to keep is done by a RANSAC algorithm. To manage the graphs, a list of matched nodes and a condensed graph containing the edges between the matched nodes are transmitted to the corresponding robot.
A post processing step is added to merge the entire maps after the exploration.

Another algorithm for landmark based maps is *Decoupled-SLAM* [31]. Each robot builds a local map with EKF-SLAM representing the path of the robot relative to the initial location of the local map and all the local beacons in field of view. The global map contains the positions of all known beacons and the initial positions in all local maps with their relative transformations. The main idea is to consider local maps as measurements of the global map: The fusing of local maps is done by transforming the local map state estimate into relative location information among the beacons and the robot's initial pose and then fusing the local map into the global one. To this means, map overlap and corresponding beacons have to be found. The main difficulties are deformation of measurements (observations of different robots cannot be exactly the same) and beacons with no correspondence. An algorithm from a medical feature-matching scenario is employed to find alignment hypotheses through an optimization network. The beacon correspondences are verified by a joint compatibility test.

## A.2 Hybrid Map Representations for Multi-Robot SLAM

Finally, some approaches to multi-robot SLAM that combine topological and metric map representations will be presented. The idea presented in [8] is to build a graph-like topological global map where the vertices represent local metric maps of fixed size, constructed by RBPF-SLAM. The edges give the relative positions of local maps including transformation matrices and uncertainty. To fusion maps from multiple robots or to incorporate a new metric map in the global graph one needs only to add an edge that connects two local maps. This divides a large SLAM problem into several smaller ones. The local SLAM processing runs while the robot remains within a local-map square. If it leaves the square an edge is added and a knew local mapping process is started. After the exploration is completed, an optimization process is added to retrieve a corrected global map.

Another interesting hybrid approach is presented in [10]. Grid based and feature maps are used in parallel in an environment containing an indoors and an outdoors part. A policy will be trained by Reinforcement Learning to decide which map representation is best to use. Furthermore, the point in time when to merge a map between several robots is decided by the learned policy as opposed to merging it immediately upon rendezvous. The decision is based on the current status of the mapping particle filter and the current overlap in sensor data. Upon rendezvous, an initial estimation of the relative pose is computed by observation (range and bearing measurements).

As a representation of the confidence in the transformation matrix between the poses of two robots the mutual observation likelihood is then used: the likelihood of the other robot's sensor readings in the current robot's map. This the part of the decision algorithm based on the sensor data. The particle filter is confident in a map representation when this representation's particle has a high weight compared to all others, i.e. there is a high variance in the particle weights ($N_{eff} < N/2$ criterion (see equation (2.18))). Combining these heuristics (the $N_{eff}$ criterion and the confidence in the pose estimate for topological and metric maps) a reinforcement learning-based decision policy is calculated. The possible actions are: don't merge, merge feature-based, merge grid-based or merge based only on the initial pose estimate. The results show that not merging was chosen the most. Feature based merging was the most favoured merging technique. In general, merging was executed when at least one robot has $N_{eff} < N/2$ and one of the likelihood values is greater than 0.333. Here, the robots only learned when to merge maps and which representation to use, but the reinforcement learning technique could be extended to particle filter parameters or different weighting factors for the different sensors representing the confidence or the usefulness of their observations.

In general, multi-robot SLAM algorithms have high potential for cooperative robot tasks and to efficiently map large environments. The problem of map merging, however, remains complex. Almost all approaches are based on a relative pose measurement as an initial guess for the merging process, which requires the robots to physically meet to be in the same line of view. This initial estimate often being rather imprecise, overlapping map regions or landmark correspondences are used to improve the joint map. This calls for robots partially visiting the same areas, to be able to find overlap in the map and to be able to meet, but also for them to visit different parts of the environment to make use of the multiple team-mates to efficiently finish the exploration task. The following section will resume some approaches to coordinate and plan trajectories for an efficient mapping process.

# Appendix B

# Exploration and Path Planning Strategies

Efficient exploration and path planning strategies are a vital part of a mapping process. To build a complete map of an environment, a robot needs to plan which areas to visit next and, even more importantly, in the case of a multi-robot team the robots need to divide the different regions between each other in a distributed manner. According to [20] path planning strategies can be categorized by two main characteristics - whether they represent single-robot or coordinated multi-robot strategies and whether they integrate the uncertainty of the robot's localization and the uncertainty of the map in the planning.

The simplest approaches are neither coordinated nor integrated: The goal is to acquire as much information as possible in a short time. The uncertainty of the localization does not affect the path planning. The goal is simply to identify and explore unmapped regions. To this end, often the notion of *frontiers*, or *frontier cells* for occupancy grid maps, is used: This term refers to the boarders between explored and unexplored areas. There exist cost-based approaches only searching the nearest unexplored region optimal in terms of the cost to arrive. As a cost function, often a distance map to the frontier cells computed by a wavefront-propagation algorithm is used [3]. A bit more complex are cost-utility approaches also considering the utility of visiting a cell in terms of the expectation of the information to be gained. To this end, the benefit between cost and utility is maximized. The path planning will be concretely realized mostly by reactive algorithms based on potential fields and leading to the discrete behaviours of obstacle avoidance, attraction to goals (i.e. frontier cells), and eventually repulsion from previously visited cells to ensure an exhaustive exploration. Those techniques being simple and relatively easy to implement they are, however, not well suited for large and complex environments as for example local minima can exist in the potential fields.

Coordinated strategies are adapted for using multiple robots that construct a global map in a centralized manner or distributed among the robots. If the joint map is supposed known to all robots, frontiers can be assigned to robots in a coordinated manner using a cost-utility approach: A "market model" is employed to distribute targets among the robots: targets are negotiated by a system of auction and bids. In [3] a similar strategy is used, determining the rank of a robot towards a frontier. A robot determines how many robots are closer to the frontier than itself. Only the closest robot will move towards a boarder, separating the robots in different directions which leads to a well balanced distribution along the building explored in this project.
A different approach is to divide the environment in many disjoint regions and assign a different

region to each robot. Here, inner and outer frontiers (in the robot's region or not) need to be considered separately. Also, structural knowledge of corridors, rooms, etc. can be used. Furthermore, the reactive technique from the previous paragraph can be augmented with a week coordination by an *avoid other robots* behaviour.
If there is only a limited communication channel one needs to limit the distance that robots can separate, to this means different roles (explore or regroup) can be assigned to the robots, or one could keep a robot as a static reference point. Another option is the deployment of RFID tags. If the initial positions of the robots are unknown, they can explicitely plan to meet to confirm a map alignment hypothesis and to fuse individual maps. For a more detailed description of map fusion techniques, see section 2.4.

Integrated but non-coordinated strategies consider the uncertainty of the robot's localization and the uncertainty of the map in the planning. They such favour high quality maps. There exist explicit actions to close loops or to improve imprecisions in the map, for example by returning to past positions with high uncertainty to improve this measurement and thereby the uncertainty of the whole map.
Applied to the cost-utility approach, utility now represents possible information gain and localizability of the target cell, localizability meaning the predicted covariance of the position to be reached. As target points are now considered frontiers, past poses and points that close a loop.
The verification of loop closures is another important point: when a robot thinks to have reached a past place, it verifies the hypothesis p. ex. by navigating to the next known distinct place and checking the coherence with the loop closure hypothesis.
A simple implementation technique would be to continuously return to past poses or to the initial pose. More complex approaches, like path planning using reinforcement learning, can also be used. The reward function then measures the posterior uncertainty of the map. The path planning is no longer a *greedy*-algorithm considering only the information gain in the next step but plans with a longer horizon. [2] propose a nonmyoptic control policy for the robots with a horizon of $T = 12$ steps, such that the uncertainty in the map is minimized. To also effectuate a continuous exploration of the environment, dummy landmarks that promise information gain are introduced at the frontiers. The possible actions of the policy are: *None*, *Explore*, *Improve Map* and *Improve Localization*, with respectively increasing priority. This approach allows for an efficient and accurate mapping where the entropy of landmark positions decreases over time while new landmarks are continuously discovered.

Lastly, integrated and coordinated strategies take care of the coordination between multiple robots as well as the uncertainty in the SLAM process. They consider the full path until reaching a target point for the utility function. A possible implementation is to extend the reactive approach with *improve imprecise landmarks* and *go to unexplored zones* behaviours. For better coordination a high level planner in form of a decision tree is added.

To resume, there exist many possible path planning strategies and which one best to use depends on the application. Integrated mechanisms generally improve the map quality. Techniques including utility explore large areas in an early stage but discard small parts until the end, thus

increasing total exploration time. Therefore cost-based approaches are better suited to completely explore the whole area in a short time. Robot teams generally improve the map quality and the exploration time. The multi-robot coordination should be done by a global optimization p. ex. a market model.

# Bibliography


[1] Adluru, N., Latecki, L. J., Sobel, M., and Lakaemper, R. Merging Maps of Multiple Robots. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008.

[2] Atanasov, N., Ny, J. Le, Daniilidis, K., and Pappas, G. J. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4775–4782, May 2015.

[3] Bautin, Antoine, Lucidarme, Philippe, Guyonneau, Rémy, Simonin, Olivier, Lagrange, Sebastien, Delanoue, Nicolas, and Charpillet, François. Cart-O-matic project : autonomous and collaborative multi-robot localization, exploration and mapping. In *5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (a IROS 2013 workshop)*, pages 210–215, Tokyo, Japan, November 2013.

[4] Birk, A. and Carpin, S. Merging Occupancy Grid Maps from Multiple Robots. *Proceedings of the IEEE*, 94(7):1384–1397, July 2006.

[5] Blanco, Jose-Luis, González-Jiménez, Javier, and Fernández-Madrigal, Juan-Antonio. A Robust, Multi-Hypothesis Approach to Matching Occupancy Grid Maps. *Robotica*, 31:687–701, 8 2013.

[6] Carlone, L., Ng, M. Kaouk, Du, J., Bona, B., and Indri, M. Rao-Blackwellized Particle Filters multi robot SLAM with unknown initial correspondences and limited communication. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 243–249, May 2010.

[7] Carpin, S. Merging maps via Hough transform. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1878–1883, Sept 2008.

[8] Chang, H. Jacky, Lee, C. S. George, Hu, Y. Charlie, and Lu, Yung-Hsiang. Multi-robot SLAM with topological/metric maps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1467–1472, Oct 2007.

[9] Cohen, Jonathan, Matignon, Laetitia, and Simonin, Olivier. Concentric and Incremental Multi-Robot Mapping to Observe Complex Scenes. In *On-line decision-making in multi-robot coordination (DEMUR'15), IROS 2015 Workshop on*, Hamburg, Germany, Oct 2015.

[10] Dinnissen, P., Givigi, S. N., and Schwartz, H. M. Map merging of Multi-Robot SLAM using Reinforcement Learning. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 53–60, Oct 2012.

[11] Doucet, Arnaud, Freitas, Nando de, Murphy, Kevin P., and Russell, Stuart J. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[12] Dubois, Amandine and Charpillet, François. Tracking Mobile Objects with Several Kinects using HMMs and Component Labelling. In *Workshop Assistance and Service Robotics in a human environment, International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, October 2012.

[13] Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[14] Eliazar, Austin and Parr, Ronald. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping without Predetermined Landmarks. In *in Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03*, pages 1135–1142. Morgan Kaufmann, 2003.

[15] Filiat, David. *Robotique Mobile*. École Nationale Supérieure de Techniques Avancées ParisTech, 2013.

[16] Grisetti, G., Stachniss, C., and Burgard, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437, April 2005.

[17] Grisetti, G., Stachniss, C., and Burgard, W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007.

[18] Hahnel, D., Burgard, W., Fox, D., and Thrun, S. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 206–211 vol.1, Oct 2003.

[19] Howard, Andrew. Multi-Robot Simultaneous Localization and Mapping using Particle Filters. *Int. J. Rob. Res.*, 25(12):1243–1256, December 2006.

[20] Juliá, Miguel, Gil, Arturo, and Reinoso, Oscar. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[21] Lazaro, M. T., Paz, L. M., Pinies, P., Castellanos, J. A., and Grisetti, G. Multi-robot SLAM using condensed measurements. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1069–1076, Nov 2013.

[22] Li, H. and Nashashibi, F. A New Method for Occupancy Grid Maps Merging: Application to Multi-Vehicle Cooperative Local Mapping and Moving Object Detection in Outdoor Environment. In *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*, pages 632–637, Dec 2012.

[23] Li, H., Tsukada, M., Nashashibi, F., and Parent, M. Multivehicle Cooperative Local Mapping: A Methodology Based on Occupancy Grid Map Merging. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2089–2100, Oct 2014.

[24] Mahler, R. P. S. *Statistical multisource-multitarget information fusion*. Artech House, Boston, Mass., 2007.

[25] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[26] Montemerlo, Michael, Thrun, Sebastian, Koller, Daphne, and Wegbreit, Ben. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Eighteenth National Conference on Artificial Intelligence*, pages 593–598, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[27] Murphy, Kevin P. Bayesian Map Learning in Dynamic Environments. In *NIPS*, pages 1015–1021, 1999.

[28] Saeedi, Sajad, Paull, Liam, Trentini, Michael, and Li, Howard. Occupancy Grid Map Merging for Multiple-robot Simultaneous Localization and Mapping. *International Journal of Robotics and Automation*, 30(2):149–157, 2015.

[29] Saeedi, Sajad, Paull, Liam, Trentini, Michael, Seto, Mae, and Li, Howard. Efficient Map Merging using a Probabilistic Generalized Voronoi Diagram. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4419–4424, 2012.

[30] Saeedi, Sajad, Paull, Liam, Trentini, Michael, Seto, Mae, and Li, Howard. Map Merging for Multiple Robots using Hough Peak Matching. *Robotics and Autonomous Systems*, 62(10):1408–1424, 2014.

[31] Wang, Z., Huang, S., and Dissanayake, G. Multi-robot simultaneous localization and mapping using D-SLAM framework. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 317–322, Dec 2007.

[32] Wendel, J. *Integrierte Navigationssysteme : Sensordatenfusion, GPS und inertiale Navigation*. Oldenbourg, M"unchen, 2., "uberarb. aufl. edition, 2011.

[33] Zhou, X. S. and Roumeliotis, S. I. Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1785–1792, Oct 2006.