STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

STEFAN DÖRR

# Cloud-based Cooperative Long-Term SLAM for Mobile Robots in Industrial Applications


Universität Stuttgart

Fraunhofer IPA

**STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 113**



Stefan Dörr

# Cloud-based Cooperative Long-Term SLAM for Mobile Robots in Industrial Applications

FRAUNHOFER VERLAG

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11/9 70-11 01
info@ipa.fraunhofer.de; www.ipa.fraunhofer.de

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG



Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über www.dnb.de abrufbar.
ISSN: 2195-2892
ISBN (Print): 978-3-8396-1645-1

D 93

Zugl.: Stuttgart, Univ., Diss., 2019

Druck: Mediendienstleistungen des Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.


Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon 07 11 9 70-25 00
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

# Cloud-based Cooperative Long-Term SLAM for Mobile Robots in Industrial Applications

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik

der Universität Stuttgart

zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Abhandlung

Vorgelegt von

**Dipl.-Ing. Stefan Dörr**

aus Stuttgart

| | |
|---|---|
| Hauptberichter: | Univ.-Prof. Dr.-Ing. Alexander Verl |
| Mitberichter: | Univ.-Prof. Dr.-Ing. Marco Huber |
| Tag der mündlichen Prüfung: | 10.12.2019 |

Institut für Steuerungstechnik der Werkzeugmaschinen
und Fertigungseinrichtungen (ISW) der Universität Stuttgart

2019

# Vorwort des Verfassers

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA in Stuttgart.

Ein großer Dank gilt Herrn Prof. Dr.-Ing. Dr. h. c. mult. Alexander Verl für die Unterstützung meiner wissenschaftlichen Arbeit und die Übernahme des Hauptberichts. Außerdem gilt mein Dank Prof. Dr.-Ing. Marco Huber für die Durchsicht der Arbeit und für die Übernahme des Mitberichts.

Für die Organisation des Promotionsprozesses möchte ich Frau Heide Kreuzburg danken, außerdem Luzia Schumacher für die Unterstützung bei der abschließenden Durchsicht der Arbeit.

Ein besonderer Dank gilt zudem Dr.-Ing. Kai Pfeiffer und Dr.-Ing. Felipe Garcia Lopez für die vielen thematischen Diskussionen, konstruktives Feedback sowie allgemeine Unterstützung, welche diese Arbeit mit geprägt haben.

In besonderem Maße danke ich außerdem meinen Eltern für ihre Förderung und Unterstützung, die sie mir während meiner schulischen Laufbahn, meines Studiums und meiner Promotion entgegengebracht haben. Zuletzt möchte ich mich von ganzem Herzen bei meiner Freundin Mascha bedanken, die mir während des gesamten Promotionszeitraums motivierend und mit viel Geduld zur Seite stand. Ihr sowie meinen Kindern Luc und Lila ist diese Arbeit gewidmet.

Stuttgart, im Februar 2020 Stefan Dörr

# Kurzinhalt

Mobile Roboter wie fahrerlose Transportsysteme (FTS) erfahren seit Jahrzehnten zunehmend Anwendung für intralogistische Aufgaben im Bereich der industriellen Produktion und Logistik. Um die benötigten Verfügbarkeiten dieser industriellen Anforderungen zu erreichen, benötigen derzeitig kommerziell verfügbare Navigationslösungen zumeist zusätzliche in die Umgebung eingebrachte Infrastruktur, wie magnetische Leitlinien oder retro-reflektive Marker, oder sind begrenzt auf Anwendungsgebiete mit stark strukturierten, unveränderlichen Umgebungen. Die derzeit steigende Nachfrage nach hochflexiblen FTS, welche auch im dynamischen Umfeld und in nicht-abgetrennten Arbeitsbereichen mit Menschen und anderen Fahrzeugen effizient navigieren, kann mit diesen Lösungen nur unzureichend befriedigt werden. Navigationslösungen, die diesen hohen Anforderungen gerecht werden, müssen vielmehr über einen gesteigerten Autonomiegrad verfügen und gleichzeitig bestehende Anforderungen hinsichtlich Verfügbarkeit und Präzision aufrechterhalten.

Diese Arbeit beschäftigt sich mit der Entwicklung entsprechender Navigationsverfahren unter Ausnutzung aktueller Entwicklungen im Bereich der Industrie 4.0 und Cloud-Robotik. Konkret wird ein cloud-basiertes, kooperatives Navigationsverfahren vorgestellt, welches die Vernetzung der Roboter untereinander sowie die Ausnutzung externer Rechenressourcen auf Cloud-Servern integriert. Um derzeitige Beschränkungen im Bereich der drahtlosen Datenübertragung zu berücksichtigen, verbleiben dabei Basis-Navigationsfähigkeiten auf den mobilen Robotern, wodurch deren Navigationsfähigkeit auch bei Netzwerkverbindungsabbrüchen aufrecht erhalten wird. Auf Cloud-Servern ausgelagerte, kooperative Navigationsverfahren sorgen hingegen für die notwendige Langzeitstabilität der Navigation, auch in herausfordernden Umgebungen.

Eine Schlüsselfähigkeit des kooperativen Navigationssystems ist das Generieren von stets aktuellen Umgebungskarten für Lokalisierung und Pfadplanung basierend auf den Sensorbeobachtungen der gesamten Roboterflotte. Hierfür wird ein kooperativer Long-Term Simultaneous Localization and Mapping (LT-SLAM)-Ansatz präsentiert, bei welchem jeder Roboter detektierte Kartenänderungen an den cloud-basierten LT-SLAM-Server überträgt. Dieser fusioniert die eingehenden Karteninformationen in eine konsistente, globale Karte und stellt diese den einzelnen Robotern in Form von Karten-Updates zur Verfügung. Zusätzlich wird ein Verfahren zur gegenseitigen Detektion der Roboter und zum Austausch von Lokalisierungsinformationen integriert. Ein weiterer Fokus der Arbeit liegt auf der Entwick-

lung einer Kartenrepräsentation, welche die vielfältigen und teils konträren Anforderungen dieser Applikationen erfüllt.

Sowohl Wirksamkeit als auch praktische Anwendbarkeit werden anschließend in verschiedenen Simulations- und Echtwelt-Experimenten nachgewiesen. Dabei überzeugt der vorgestellte Ansatz insbesondere in erhöhter Lokalisierungspräzision und -robustheit, kürzeren Navigationswegen gegenüber unvernetzten Ansätzen sowie moderaten Netzwerkauslastungen.

# Abstract

In the past decades, fleets of mobile robots have increasingly entered the sector of industrial production and warehousing, replacing conventional logistic systems like conveyor belts and human-controlled vehicles. To achieve the reliability needed for these industrial applications, current navigation solutions commonly rely on additional infrastructure (like magnetic wires or retro-reflective markers) or are limited to highly structured, non-changing environments. This circumstance, however, limits their flexibility with respect to modifications of the environment or altered transportation tasks as well as their efficiency when operating in shared workspaces with humans or other dynamic objects. Moreover, this forbids a further exploration into new applications of highly dynamic and changing environments. In order to overcome these limitations, navigation solutions with an increased level of autonomy without decreasing reliability or precision of current solutions are needed.

This thesis tackles this issue by leveraging current advances in the fields of cloud and networked robotics for the particular application of mobile robot navigation. We propose a cloud-based cooperative navigation architecture which enables knowledge sharing and remote computing for the mobile robots. The main concept of this architecture consists in keeping basic navigation functionalities on the mobile robots to make them temporal independent of the cloud server and provide long-term navigation capabilities through globally coherent navigation solutions running on the server side. Thereby, the mobile robots do not rely on low-latency or high-frequency server information and are able to maintain their operational capability in the presence of network disruptions. Since the availability of up-to-date map information is of crucial importance for both localization and path planning when facing dynamic and highly changing environments, the thesis' main focus consists in leveraging the shared sensor observations to build and maintain an up-to-date global map.

Consequently, the thesis proposes a cooperative long-term simultaneous localization and mapping (LT-SLAM) approach where each mobile robot shares its detected map changes with the cloud-based LT-SLAM server, which fuses the incoming map information into a consistent global map and provides map updates to the robots. A further emphasize of this work consists in deriving a map representation specifically tailored for this application and its manifold requirements. More concretely, we introduce low-resolution dynamic occupancy grid maps in combination with a cell-wise continuous representation of the object within the cell modeling its contour and reflectivity using mixture models. We then show how this

map representation is integrated into a local LT-SLAM approach running on each robot and providing high-frequency localization estimates as well as local map updates. Additionally, we demonstrate how the cooperative functionalities can be seamlessly added in terms of cooperative map updating and cooperative localization by mutual detection.

Numerous simulative and real-world experiments demonstrate both effectiveness and practicability of the cooperative LT-SLAM approach in terms of increased localization robustness and accuracy, improved navigation efficiency with reduced travel times as well as reasonable network loads.

# Contents

**Abbreviations and Symbols** **xii**

**List of Figures** **xv**

**List of Tables** **xviii**

**List of Algorithms** **xix**

**1 Introduction** **1**

1.1 Motivation . . . 1

1.1.1 Mobile Robots and Automated Guided Vehicles . . . 1

1.1.2 Autonomy of Mobile Robots . . . 2

1.1.3 Evolution of Navigation Solutions . . . 3

1.2 Problem Statement . . . 6

1.3 Objectives . . . 9

1.3.1 Approach and Contribution . . . 9

1.3.2 Delimitation . . . 10

1.4 Outline . . . 11

1.5 Publication Note and Collaboration . . . 11

**2 Fundamentals** **12**

2.1 Probabilistic State Estimation . . . 12

2.1.1 Bayes Filters . . . 13

2.1.2 Binary Bayes Filter . . . 14

2.1.3 Particle Filter . . . 14

2.1.4 Normal Distribution Parameter Estimation . . . 15

2.2 Navigation of Mobile Robots . . . 20

2.2.1 Sensors . . . 21

2.2.2 Map Representations . . . 23

2.2.3 Localization . . . 25

2.2.4 Mapping . . . 27

2.2.5 SLAM . . . 28
2.2.6 Long-Term SLAM . . . 29
2.2.7 Path Planning and Control . . . 30
2.3 Multi-Robot Systems . . . 31
2.3.1 Cooperative Robotics . . . 31
2.3.2 Networked Robotics . . . 32
2.3.3 Cloud Robotics . . . 32

**3 A Cloud-based Cooperative Navigation System 34**
3.1 Introduction . . . 34
3.1.1 Requirement Analysis . . . 35
3.1.2 Related Work . . . 36
3.2 System Architecture . . . 37
3.3 Global and Local Navigation Modules . . . 39
3.3.1 Local Perception . . . 39
3.3.2 Global Perception . . . 41
3.3.3 Global Path Planning . . . 41
3.3.4 Local Motion Control . . . 42
3.4 Conclusion . . . 43

**4 Environment Modeling for Cooperative Long-Term Map Estimation 45**
4.1 Introduction . . . 45
4.1.1 Requirement Analysis . . . 45
4.1.2 Related Work . . . 47
4.2 Adding the Normal Distribution Transform to Dynamic Occupancy Grid Maps 53
4.2.1 Long-Term Estimation of Hidden Markov Model Parameters . . . 54
4.2.2 Long-Term Estimation of Normal Distribution Parameters . . . 57
4.2.3 Updating Map Parameters with New Observations . . . 58
4.3 Gaussian Mixture Models for Non-Linear Contour Approximation . . . 59
4.3.1 Long-Term Estimation of Hidden Markov Model Parameters . . . 61
4.3.2 Long-Term Estimation of Gaussian Mixture Model Parameters . . . 61
4.4 Incorporating Object Reflectivity Information . . . 66
4.4.1 Binary Reflectivity Modeling . . . 69
4.4.2 Recursive Estimation of the Reflectivity State . . . 70
4.5 Transformation to High-Resolution Occupancy Grid Maps . . . 73
4.6 Conclusion . . . 74

**5 Multi-Robot Cooperative Long-Term SLAM and Mutual Localization 75**
5.1 Introduction . . . 75
5.1.1 Requirement Analysis . . . 75
5.1.2 Related work . . . 77
5.2 Local Long-Term SLAM . . . 81
5.2.1 Long-Term SLAM using Dual Confidence Map Layers . . . 82
5.2.2 Observation Likelihood Modeling . . . 84
5.3 Mutual Localization . . . 89
5.3.1 Mutual Localization with Known Associations . . . 91
5.3.2 Mutual Localization with Unknown Associations . . . 96
5.4 Cooperative Map Updating . . . 97
5.4.1 Map Upstreaming . . . 98
5.4.2 Map Fusion . . . 98
5.4.3 Map Downstreaming . . . 99
5.5 Conclusion . . . 99

**6 Evaluation 101**
6.1 Implementation . . . 101
6.2 Experiments . . . 104
6.2.1 Long-Term Map Estimation . . . 104
6.2.2 Local Long-Term SLAM . . . 109
6.2.3 Cooperative Map Updating . . . 118
6.2.4 Mutual Localization . . . 124

**7 Conclusion 127**
7.1 Summary . . . 127
7.2 Outlook . . . 129

**Bibliography 133**

# Abbreviations

| | |
|---|---|
| **AGV** | automated guided vehicle |
| **ATV** | autonomous transport vehicle |
| **C-LT-SLAM** | cooperative LT-SLAM |
| **C-SLAM** | cooperative SLAM |
| **CBS** | constraint-based search |
| **DATMO** | detection and tracking of moving objects |
| **DOGM** | dynamic OGM |
| **EIF** | extended information filter |
| **EKF** | extended KF |
| **EM** | expectation maximization |
| **EWMA** | exponentially weighted moving average |
| **FN** | false negative |
| **FP** | false positive |
| **GMM** | Gaussian mixture model |
| **GPS** | Global Positioning System |
| **HMM** | Hidden Markov Model |
| **HR** | high-resolution |
| **IF** | information filter |
| **IGMM** | incremental Gaussian mixture model |
| **IMU** | inertial measurement unit |
| **IoT** | Internet of Things |
| **IPC** | industrial PC |
| **IPS** | Indoor Positioning System |
| **KF** | Kalman filter |
| **KLD** | Kullback-Leibler divergence |
| **Lidar** | light detection and ranging |
| **LR** | low-resolution |
| **LT-SLAM** | long-term SLAM |
| **M2M** | machine-to-machine |
| **MAP** | maximum a posteriori |
| **MCL** | Monte-Carlo localization |
| **MPE** | mean position error |

| | |
|---|---|
| **MR** | mobile robot |
| **NDT** | normal distribution transform |
| **NIST** | National Institute for Standards and Technology |
| **OGM** | occupancy grid map |
| **ORGM** | occupancy reflectivity grid map |
| **PDF** | probability density function |
| **PF** | particle filter |
| **QR** | quick response |
| **QT** | quadtree |
| **RBPF** | Rao-Blackwellized PF |
| **RFID** | radio-frequency identification |
| **ROS** | Robot Operating System |
| **RSC** | recursive sample covariance |
| **SIR** | sampling importance resampling |
| **SLAM** | simultaneous localization and mapping |
| **SM** | split and merge |
| **SME** | small and medium-sized enterprise |
| **STR** | smart transport robot |
| **TOF** | time of flight |
| **TP** | true positive |
| **UMS** | unmanned system |
| **US** | ultrasonic |
| **VGN** | varying graphical network |
| **WLAN** | wireless local area network |
| **xGMM** | extended GMM |

# Symbols

## Latin Letters

| | |
|---|---|
| $A$ | occupancy state transition matrix |
| $B$ | occupancy state observation matrix |
| $D$ | dimension of $x$ |
| $F$ | F-measure |
| $H$ | Shannon entropy |
| $I$ | identity matrix |
| $L$ | likelihood |
| $\mathcal{M}$ | global map frame |
| $N$ | number |
| $Q$ | occupancy probability vector |
| $R$ | reflectivity probability vector |
| $\mathcal{R}$ | robot frame |
| $\mathcal{S}$ | sensor frame |
| $T$ | transformation matrix |
| $X$ | particle set |
| $a$ | association of measurement |
| $bel$ | belief over state |
| $c$ | grid cell |
| $d$ | distance on laser beam with respect to sensor origin |
| $f$ | free cell state |
| $k$ | gain parameter |
| $l$ | log odds |
| $m$ | map of the environment |
| $o$ | occupied cell state |
| $p$ | probability |
| $q$ | binary occupancy state of a grid cell being free or occupied, $q = \{f, o\}$ |

$r$ range measurement
$s$ summed posterior probability
$t$ time
$u$ odometry
$v$ verified
$\bar{v}$ non-verified
$w$ importance weight
$x$ state space
$z$ sensor observation

## Greek Letters

$Z$ reflectivity measurement vector
$\Theta$ cell parameter set
$\Sigma$ covariance matrix
$\Phi$ cumulative normal distribution function
$\alpha$ bearing measurement
$\gamma$ weigthing factor
$\delta$ Dirac function
$\zeta$ scan point
$\eta$ normalizing constant
$\theta$ model parameter set
$\lambda$ eigenvalue
$\mu$ mean
$\nu$ eigenvector
$\xi$ detector robot
$\pi$ posterior probability
$\rho$ reflectivity state
$\sigma$ variance
$\tau$ uncertainty measure of localization belief
$\phi$ iteration index
$\chi$ detected robot
$\upsilon$ robot of detected type
$\psi$ running index for components of mixture model

# List of Figures


1.1 Examples of commercial AGVs . . . . . 4

2.1 Map-based control scheme for navigation of MRs . . . . . 21
2.2 Visualization of different map types . . . . . 23

3.1 Basic concept of the cooperative navigation system . . . . . 38
3.2 System architecture of the cooperative navigation system . . . . . 40

4.1 Comparison of geometric map approaches . . . . . 52
4.2 Derivation of cell observation likelihoods . . . . . 55
4.3 Sensor beam traversing a cell from different directions . . . . . 56
4.4 Approximation of common geometric contours . . . . . 60
4.5 Evolvement of GMM using SM-IGMM in two selected scenarios . . . . . 63
4.6 Relation of 2D normal distribution parameters to the geometry of an ellipse 64
4.7 Example of MR traversing corridor with high reflective door frames. . . . . 67
4.8 Approximation of wall with retro-reflective marker using xGMM . . . . . 68
4.9 Conversion of LR xGMM-DOGM into LR/HR OGM . . . . . 73

5.1 Different observations of cell containing a wall . . . . . 86
5.2 Common robot shapes and possible implications for contour-based detection 90
5.3 Information flow for mutual localization with known associations . . . . . 94
5.4 Mutual detection procedure in simplified 1D example . . . . . 95
5.5 Information flow for mutual localization with unknown associations . . . . . 97

6.1 MRs of Fraunhofer IPA: rob@work 3 and Care-O-bot 4 . . . . . 102
6.2 Simulated MRs in Gazebo used for simulation experiments . . . . . 103
6.3 Setup of map estimation experiment . . . . . 105
6.4 Summarized results of *high-frequently changing corner* test case . . . . . 109
6.5 Simulated warehouse used for localization benchmarks . . . . . 110
6.6 Overview of results for *fully static* test case . . . . . 113
6.7 Position error in *fully static* test case . . . . . 113
6.8 Overview of results for different test cases . . . . . 115
6.9 Position error over time in reflectivity processing experiment . . . . . 117
6.10 Comparison of maps after 8-week operation in automotive production scene . 118
6.11 Overview of results of cooperative map updating for different test cases . . . 119

6.12 Position error over time in different test cases . . . . . 120
6.13 Setup of path planning experiment . . . . . 122
6.14 Test setup of mutual localization experiment . . . . . 125
6.15 Results of mutual localization experiment . . . . . 126

# List of Tables

4.1 Rating of relevant map representation approaches . . . . . 51
4.2 Rating of LR xGMM-DOGM . . . . . 52

6.1 Results from *wall* test cases . . . . . 108
6.2 Results from *corner* test cases . . . . . 108
6.3 Environment configuration for local LT-SLAM test cases. . . . . 111
6.4 Comparison of path planning performance . . . . . 122
6.5 Results from bandwith analysis . . . . . 124

# List of Algorithms

2.1 EWMA-RSC update algorithm . . . . . 16
2.2 EWMA-RSC update algorithm with multiple points . . . . . 17
2.3 EM algorithm for estimating a GMM from a complete data set . . . . . 18
2.4 IGMM algorithm for recursively estimating a GMM from a data stream . . . 20

4.1 NDT parameter update algorithm . . . . . 58
4.2 NDT-DOGM update algorithm . . . . . 59
4.3 SM algorithm for GMM estimation . . . . . 66

5.1 MCL with MAP pose mapping on dual-confidence maps update . . . . . 85
5.2 Observation likelihood for NDT-OGM . . . . . 87

# 1 Introduction

Due to their versatility, mobile robots (MRs) are on the rise in various fields such as production, agriculture, military, entertainment, and household with new applications popping up on a regular basis. Moreover, in some fields, they are given the potential of revolutionizing whole industries like autonomous cars in the transportation industry.

For decades, logistics has been and still is the predominant sector for applying MRs (Haegele 2017). In these applications, the MRs execute transportation tasks to realize a more flexible material flow with respect to conventional logistic solutions like conveyor belts. Within this context, the MRs are commonly referred to as automated guided vehicles (AGVs) and usually operate in fleets of a few up to hundreds of vehicles (Ullrich 2014)(D'Andrea 2012).

While current commercial solutions realize reliable transport systems for large-scale logistic centers or production plants, they commonly represent rather static and inflexible installations which can only be adapted to changes of the environment or to new transportation tasks with extensive efforts. This circumstance strongly conflicts with the major demand for alterable and flexible production and logistics. Moreover, it restricts current AGV systems to rather controlled and structured workspaces limiting the exploration of new applications. Lastly, it usually makes them unaffordable for small and medium-sized enterprises (SMEs) due to high investment costs.

Increasing the flexibility of MRs to meet these demands directly correlates with increasing their autonomy. Since mobility is the central service of an MR, advanced navigation capabilities are the key component to develop the next generation AGVs (Bubeck, Gruhler et al. 2017).

## 1.1 Motivation

### 1.1.1 Mobile Robots and Automated Guided Vehicles

As described in the previous section, AGVs represent a certain type of MRs designed for the particular task of transporting materials and goods in industrial sites. An MR can thereby be seen as an umbrella term for all kinds of mobile platforms that are capable of locomotion and optionally combined with one or several manipulators (ISO 8373 2012). Based on the

environment in which the MR is capable of operating, it can further be classified into ground, aerial, underwater or space robots. For an overview of various realizations in the different categories, please refer to (Siegwart and Nourbakhsh 2004).

Looking at ground robots, we can further distinguish between wheeled and legged types. While we currently find most legged robots in the research stage, their wheeled counterparts already have been on the market since the early 1950s in terms of the already described AGVs (Ullrich 2014). Although the term AGV is not used consistently throughout the literature, we follow the definition of ISO 8373 (2012) which defines an AGV as a (wheeled) mobile platform following predefined paths. The paths are usually indicated by guiding lines or other kind of markers on the floor (see Subsection 1.1.3 for a more detailed discussion of navigation solutions). AGVs are thereby capable of executing predefined transport tasks by following fixed paths. The commissioning and coordination of the AGV fleet are commonly carried out by a central control system that receives the transportation orders from a manufacturing or logistic management system. It also has to make sure that no conflicts among the AGVs occur since the AGVs are not capable of leaving the path to resolve conflicts by themselves.

As stated above, the term AGV is sometimes also used for MRs with higher navigation capabilities, e.g., an MR equipped with a dynamic path planning module that is able to resolve the previously mentioned type of conflict. However, this is misleading since the term AGV contains the term "guided" which is a property that poorly describes an MR capable of computing optimal paths during runtime. Instead, throughout this work, we stick to the classical definition of AGVs as line-guided MRs with rather low autonomy levels. For describing MRs with more advanced capabilities, the terms autonomous transport vehicle (ATV) or autonomous MR are commonly used. However, these terms raise the question at which stage an MR can be declared as autonomous.

### 1.1.2 Autonomy of Mobile Robots

For being able to rate the autonomy of an MR, we need a clarification what the term stands for within the context of MRs. When reviewing literature in that regard, we see that there is no detailed and common sense definition. ISO standard 8373 (ISO 8373 2012) briefly defines autonomy as the *"ability to perform intended tasks based on current state and sensing, without human intervention."* While this gives a first idea, this definition is still too vague for being able to derive different autonomy levels. Comparable to the ISO standard 8373, the U.S. National Institute for Standards and Technology (NIST) has defined standardized terms for unmanned system (UMS) in (Pavek, Smith et al. 2008). Following their definition, a fully autonomous UMS *"accomplishes its assigned mission, within a defined scope, without human intervention while adapting to operational and environmental conditions."* Still, this only defines a binary state, namely fully autonomous and non-autonomous, and not different

levels of autonomy. However, we can use the key insight of this definition in terms of the ability to handle situations when facing changes to make the following definition which is used to rate autonomy throughout this thesis:

**Definition: Autonomy of MRs**

The autonomy of an MR is defined as the ability to accomplish a given task when facing environmental or operational changes. The autonomy level of an MR can then be derived based on the possibly occurring changes in regard of its target application under which the MR is able to sustain its autonomy.

This definition puts the autonomy level of an MR in the context of the actual application which can lead to different autonomy levels of the same MR when operating in different applications. As an example, a mobile household robot may exhibit a high autonomy within its domestic environment but can hardly operate in an outdoor environment. Or more related to MRs in logistic applications, an MR designed to operate in a completely static environment without humans or other dynamic objects in its workspace with a predefined, unchanged task may achieve high autonomy. However, if we populate this environment with humans or other dynamic objects and apply constant changes in terms of moving or removing objects, the autonomy level usually decreases drastically. Since the major requirement for an MR operating in logistics or production scenes is mobility, the applied navigation solution is the predominant factor for its autonomy and will therefore be discussed in the following section.

Apart from the derived autonomy criteria, another aspect is often named in terms of the dependency on energy supply. Following this, an MR can only be defined as fully autonomous if it does not depend on external energy supply (e.g. via solar cells). While this seems an important aspect, e.g., for space robots, MRs operating in industrial applications normally have quick access to charging infrastructure in close proximity. Therefore, energy supply plays a subordinate role for their autonomy.

### 1.1.3 Evolution of Navigation Solutions

As already mentioned in Subsection 1.1.1, first solutions of AGVs have been on the market for decades and relied on (electrical, magnetic or optical) physical line guidance (see Figure 1.1 top left for an exemplary AGV). Until today, the major part of installed AGVs still uses this technology since it stands out in terms of reliability and comparably low hardware and software requirements. However, this comes at cost of flexibility. Altering a path induces high efforts to adjust the guiding line. When using optical lines, this effort can be lowered with the drawback of losing reliability since the optical line is prone to removals induced by fork lifts or other vehicles operating in the same environment. In


Figure 1.1: Examples of commercial AGVs: Optical line-guided AGV 'Weasel' of SSI Schäfer GmbH (SSI Schäfer GmbH 2018) (top left), automated tugger train with virtual line guidance and reflector marker-based localization from Jungheinrich AG (Jungheinrich AG 2018) (top right), virtual line-guided AGV of Bär Automation GmbH carrying car bodies in automotive manufacturing using safety light detection and ranging (Lidar) sensors and natural landmarks for localization (source: Bär Automation GmbH) (bottom left), virtual line-guided AGV "smart transport robot (STR)" for automotive logistics with Indoor Positioning System (IPS) localization (source: BMW AG) (bottom right).

general, physically line-guided AGV solutions exhibit only a low autonomy level since they can handle neither environmental nor operational changes that affect the static guiding line. Using grids of magnetic or radio-frequency identification (RFID) tags inserted into the floor can be seen as an enhancement of line guidance. While this technology increases flexibility in terms of selectable paths, the drawbacks of high installation and adaption costs remain. Due to this reason, in the second generation of AGVs, physical guiding lines or grids were replaced by virtual paths. For being able to follow these virtual paths, a localization system localizes the AGV with respect to a fixed reference frame using a specific localization infrastructure. Most popular solutions are based on specialized Lidar sensors which are able to precisely detect retro-reflective markers. Those markers need to be deployed throughout the desired workspace of the AGV. To avoid occlusions, commercial solutions commonly place the sensor and markers possibly high above the ground. The localization of the AGV can then be inferred by triangulation measurements of at least three

retro-reflective markers and a manually generated map of the markers. Indoor Positioning Systems (IPSs) are another technology for localization used for AGV solutions within the category of infrastructure-based localization systems.

Virtually guided AGVs exhibit an increased flexibility since the localization system in principle allows them to navigate on arbitrary paths covered by the localization system. Modifications of the path network can usually be carried out offline by a human via a user interface. When looking at autonomy, only slight advances with respect to the physically guided counterparts can be achieved. Slight environmental changes (e.g., an occlusion of few markers due to removed objects) can normally be compensated by the localization system while major occlusions or removed markers will lead to failures. Moreover, due to still following predefined paths, neither navigating to arbitrary goals nor bypassing blocked paths can be realized. Finally, the additionally needed software is more complex and thereby error-prone.

Enhancements of the previously described localization systems are approaches relying on natural landmarks, i.e., without adding any additional markers or infrastructure but using the given structure of the environment for localization, also referred to as infrastructure-less localization. Similar to the marker-based approaches, localization is carried out by matching current sensor observations to a given map of the environment. First commercially available solutions based on a manual mapping step to generate the map from building plans or from manual measurements. Instead, latest solutions use simultaneous localization and mapping (SLAM) approaches which are able to build up the map based on sensor observations gathered during a setup run of the AGV through the environment. In contrast to artificial markers, infrastructure-less localization using SLAM methods for initial mapping drastically reduces investment and installation costs. Further, a higher flexibility of the AGVs is achieved since new areas can quickly be added or existing be remapped once the environment has changed. However, the robustness and error rate of the localization system strongly correlates with the availability of clearly identifiable landmarks and the divergence of the current environment from the time the map was created, prohibiting applications in changing environments. Apart from that, increased software complexity is another downside of these solutions.

A further increase in autonomy is achieved with navigation systems which are able to dynamically compute and optimize the MR's path, commonly referred to as dynamic path planning. First, this enables the capability to navigate to arbitrary goals within the given environment. Second, by becoming able to adjust the paths dynamically with respect to observes obstacles and blockades, environmental changes can be handled in a substantially increased manner. Typical approaches continuously evaluate sensor observations to build an obstacle map and perform reactive collision avoidance, see, e.g., (LaValle 2006) for an overview of respective approaches and algorithms. An object that is blocking the path

of the MR can thereby easily be bypassed. However, since those approaches usually are not able to distinguish between static and dynamic obstacles, the applicability decreases with an increased dynamics of the environment. Although, the reactive behavior of the path planner allows it to still handle most situations, the computed paths are usually less efficient. Moreover, in narrow or highly dynamic areas, the planner might not find feasible paths leading in worst case to mutual blocking. Cooperative or predictive path planning approaches take the dynamics of the environment into account and predict its evolution to resolve the described situations in a more efficient manner. However, due to the complexity, most of these approaches are still in the research stage.

Still, navigation systems using infrastructure-less localization and dynamic path planning provide a significant increased autonomy level of the MR and hence forge ahead from robotic research into commercial solutions.

## 1.2 Problem Statement

As described in the previous section, there has been major advances in the field of MR navigation to increase the scalability and flexibility as well as to reduce investment and installation costs of the overall MR system. However, it also has been shown that current solutions still rely on structured environments with limited dynamics and modifications. This, however, is complementary to the increasing demand for versatile MRs in production and logistics. The environmental and operational conditions that the MR needs to resolve can be summed up to the following properties of our target applications (see also Bubeck, Gruhler et al. (2017) and Andreasson, Bouguerra et al. (2015)):

- Efficient operation in shared workspaces with humans, other MRs or human-controlled vehicles, e.g., in automotive assembly lines.
- Efficient operation in changing and large-scale environments, e.g., busy warehouses.
- Fast and easy set-up as well as flexible and adaptive usage of the MR, e.g., in terms of changing transportation tasks.
- Efficient navigation without or only slight intervention in the environment.
- Scalability and economic efficiency of the overall MR system.

Finding a suitable navigation system that fulfills these conditions can be basically broken down to two core components which are infrastructure-less localization and predictive path planning. Infrastructure-less localization is needed to provide a sufficiently accurate and robust localization without relying on artificial landmarks or additional sensors exclusively deployed for localization. Predictive path planning is then responsible to efficiently navigate the MR through the highly dynamic and changing environment.

**Categorization of object types**

- Static objects: Objects whose location cannot change like walls, essential pillars or similar objects.
- Semi-static objects: Objects that are temporarily static but their location can change over time like boxes, pallets, furniture, etc.
- Dynamic objects: Objects that move continuously like people, vehicles or MRs.

If we have a look at scientific approaches for these two components, we realize that both components strongly depend on sufficient information about the current state of the environment. Due to the environment's dynamics, this information needs to be continuously gathered and updated from sensor observations.

Long-term SLAM (LT-SLAM) approaches tackle the issue of updating the environment's map according to detected changes while simultaneously localize the robot within this map. By doing this, the localization is able to handle changing environments to a higher degree compared to the previously discussed localization approaches relying on a static map. Although some scientific approaches have shown promising results in related fields, the complexity of our target applications, mainly in terms of the high degree of changes as well as the large-scale size of the environment that the MR may be exposed to, impose major challenges to these approaches (see Subsection 5.1.2 for a detailed discussion). Further enhancing these approaches to face these challenges may improve their applicability to some degree but the progress of these improvements will sooner or later saturate due to a profound problem. The problem is that the restricted spatial and temporal information horizon is solely basing on local (i.e., on-board) hardware and sensors. Or in other words, the information extractable from the local sensor data of the MR are critical to realize stable and robust implementations for our target applications. Furthermore, the high complexity of the required algorithms would drastically increase the hardware requirements reducing the economical efficiency of these solutions. Same problems hold for the field of detection, tracking and prediction of dynamic objects needed by a predictive path planner.

As a specific example, let us have a look at the AGV carrying car bodies in automotive manufacturing (see Figure 1.1 bottom left) where the author was involved in designing and implementing the localization system, a process that highly motivated this work. The basic idea behind this logistic system is to replace the conventional conveyor belt with a fleet of 18 AGVs and thereby to be able to handle variation diversity in an improved manner as well as to simplify the process of introducing new manufacturing steps. From the navigation perspective, approaches with less infrastructure needs are demanded to fulfill this overall goal. The environment, however, is highly adverse for these kinds of approaches since it is

populated in most parts with semi-static and dynamic objects like workers, tool trolleys, movable Kanban shelfs and other logistic vehicles.

Furthermore, due to the task of carrying car bodies and the resulting hardware design, possible sensor mounting positions are limited to a small region closely above the ground inhibiting the deployment of solutions that base on observing more static parts of the environment on higher heights or the ceiling. These circumstances vigorously motivate the usage of an (LT-)SLAM approach. However, since the AGVs are traveling on a fixed round course with round-trip times of at least eight hours, an area may strongly change during this time period challenging state-of-the-art LT-SLAM techniques to achieve the required accuracy and disposability within this industrial installation.

To overcome these issues and get the system running, much more infrastructure (in terms of RFID tags embedded in the floor) was needed. This, however, adversely affected the flexibility and alterability of the overall installation as well as the sensitivity for further environmental changes. In this setup, the major problem for an LT-SLAM relying on local hardware and sensors are the continuous changes of the environment which can only be observed partially due to the large-scale size of the environment, the limited field of view of the on-board sensors and the slow operational velocity of the AGVs. In contrast, when changing the perspective from a single AGV to the fleet, we find 18 AGVs traveling nearly equally distributed along the round course, each of them continuously gathering sensor data of its local environment. Each AGV has a predecessor that is currently observing the area that the AGV will enter in a few moments whereas the AGV's information about the upcoming area might be highly deprecated. Thus, the lack of observations is a local problem of the AGV whereas it vanishes on the global (fleet) level. However, due to the missing capability of sharing sensor information among the fleet, this circumstance remains unexploited and inhibits realizing a highly efficient, reliable and flexible localization system.

Local lack of knowledge and computational resources are common issues tackled by the upcoming fields of networked and cloud robotics aiming at enabling capabilities that would be out of scope for a single robot by providing remote sensing, control and computation possibilities. Applied to our problem of generating up-to-date maps and overcoming the lack of global knowledge, cloud robotics provides the possibility of sharing sensor and state information among the fleet as well as deploying computational intensive algorithms on remote cloud servers. Furthermore, this aligns with current developments of Industry 4.0 and Internet of Things (IoT) (Hermann, Pentek et al. 2016) (Vogel-Heuser, Bauernhansl et al. 2014) which does not only push corresponding technologies (such as wireless networks and cloud computing) into nowadays industrial sites but also the availability of sensor data and further information of the industrial site within the network.

## 1.3 Objectives

This thesis deals with developing a navigation system for MR in industrial applications by exploiting recent advances in cloud robotics to overcome limitations of current local solutions. Herein, the focus lies on localization and mapping. As indicated in the problem statement of Section 1.2, the thesis' work bases on the following two hypotheses:

1. The collective of sensors from the MR fleet and the industrial environment delivers sufficient information to allow generating and maintaining up-to-date maps fulfilling the requirements of both localization and path planning.

2. Cloud computing and IoT serve as the technological enablers to process and share relevant information among the network of MRs and stationary sensors.

Based on these hypotheses, the central research question that this work tackles is:

> How can the available sensor information be exploited to cooperatively generate up-to-date maps suitable for dynamic and large-scale industrial applications as a base for infrastructure-less localization and predictive path planning?

Since localization and mapping are closely connected, an integrated approach is needed that is able to provide robust and accurate localization for the whole fleet and simultaneously updates a global map by fusing all available sensor information.

### 1.3.1 Approach and Contribution

We approach the demanded capabilities with a multi-robot cooperative LT-SLAM (C-LT-SLAM) consisting of a local LT-SLAM and a central LT-SLAM server. The local LT-SLAM is deployed on the MRs to provide (a) accurate, robust, low-latency and high-frequency localization, (b) a local map for the local path planner, and (c) local map updates and mutual detections for the central LT-SLAM server. The central LT-SLAM server is deployed on a cloud server and collects the detected map changes and mutual observations from all MRs, fuses them into a consistent global map and provides individual map and localization updates to the MRs. Furthermore, the global map serves as an input for the cooperative global path planner responsible for coordinating the fleet from the cloud.

The main contributions of this thesis with respect to the state-of-the-art in respective fields are as follows:

- System design of the overall cooperative multi-robot navigation system with a focus on the C-LT-SLAM by carefully considering current possibilities but also limitations of cloud computing and (wireless) networking.

- Development of a map representation suitable for non-static, large-scale environments to be used by both localization and path planning considering multiple, complementary requirements (e.g. geometric precision, detail level or processing efficiency) combining and extending several recent approaches as well as providing incremental update routines for learning the map parameters based on continuous noisy sensor observations.
- The development of a suitable LT-SLAM approach fusing the information from different on-board sensors as well as mutual detection measurements of the MRs and providing reliable map updates for the LT-SLAM server by extending recent approaches.
- Data fusion and distribution concepts for the LT-SLAM server to be able to fulfill its task of knowledge fusion and providing it to the robots.

### 1.3.2 Delimitation

While the topic tackled in this thesis is highly relevant for different fields of MR, we focus on MRs for non-public industrial sites, operating indoor or close to buildings. 2D safety Lidar sensors are the current de-facto standard safety solution for these AGVs/MRs once their workspace is not intended to be fenced to eliminate human contact (Ullrich 2014). Dual using the raw data from these sensors for navigation is therefore highly beneficial in terms of economical efficiency but challenging since the data density, range and quality is rather poor compared to non-safety (2D or 3D) counterparts commonly used, e.g., in autonomous driving. Since our concept builds upon the idea that the sensor network is able to overcome the limitations of local knowledge, we put the 2D safety Lidar in a primary position when designing suitable localization and mapping approaches, and we derive concrete algorithms to process this kind of sensor information. Nevertheless, the concept also involves the idea of exploiting every sensor information available in the network or respectively equip certain locations or single MRs with additional sensors (e.g. vision-based sensors for object classification) that could be valuable for the navigation system. Being able to process this multi-type sensor information will be considered when designing and selecting suitable algorithms. However, their specific implementation is out of scope for this work.

Furthermore, this thesis develops concepts and algorithms using a cloud architecture. This induces the availability of a centralized, wireless communication network with the cloud server as the central instance and the possibility to connect (at least theoretically) an arbitrary number of clients (mostly MR and stationary sensors) to it. Other communication networks like machine-to-machine (M2M) will not be considered. This does not mean that the developed concepts within this work are unsuitable for those communication networks but rather that the applicability is not further investigated.

## 1.4 Outline

The remainder of the document is organized as follows. Chapter 2 introduces some fundamentals in the field of probabilistic state estimation as well as basics of MR navigation and multi-robot systems. In Chapters 3–5, the actual approach of this work is presented starting with designing the overall navigation system and deriving its components and interfaces (Chapter 3), followed by developing the map representation (Chapter 4) which is used by the C-LT-SLAM presented in (Chapter 5). Each of these chapters is structured in a way that we first examine the respective requirements, discuss related work within the specific field and consequently derive the approach. The approach is extensively tested and evaluated in different simulative and real-world experiments in Chapter 6. Finally, this work is concluded in Chapter 7.

## 1.5 Publication Note and Collaboration

Parts of this work have already been published or presented on conferences and journals, i.e., the basic concept of the C-LT-SLAM in (Dörr, Barsch et al. 2016) and parts of the mutual localization approach (see Section 5.3) in (Dietrich and Dörr 2019).

Further note that the overall navigation system design presented in Chapter 3 is the result of joint work with research fellows Felipe Garcia Lopez and Jannik Abbenseth and was partly already published in (Abbenseth, Lopez et al. 2017). Further details about the cooperative global path planner, briefly presented in Subsection 3.3.3, can be taken from (Abbenseth 2016). The development of cooperative local motion control (Subsection 3.3.4) is part of the work of Garcia Lopez (2018). Additionally, the dynamic roadmap server briefly described in Subsection 3.3.1 is the result of the master thesis of Falk Engmann supervised by the author of this thesis.

# 2 Fundamentals

*This chapter briefly reviews basic concepts, notations and formulas in the field of navigation of MRs with a focus on localization and SLAM techniques. We start with a brief introduction on probabilistic state estimation (Section 2.1) as the most important mathematical background of this work. Afterwards, the general topic of MR navigation (Section 2.2) as well as typically used sensors (Subsection 2.2.1) are presented, followed by a discussion of map representations (Subsection 2.2.2) and a description of most relevant tasks: localization (Subsection 2.2.3), mapping (Subsection 2.2.4), SLAM (Subsection 2.2.5), LT-SLAM (Subsection 2.2.6) and path planning (Subsection 2.2.7). Furthermore, the final Section 2.3 deals with multi-robot systems and respective cooperative and networked approaches as well as cloud robotics.*

## 2.1 Probabilistic State Estimation

In this section, the basic concepts of probabilistic state estimation altogether with algorithms relevant for this work are presented. For a deeper insight and derivations of the given formulas, the reader may be referred to Thrun, Fox et al. (2005) and Haug (2012).

Bayes' theorem as the most fundamental equation used throughout this work describes the probability of an event $A$ to occur given event $B$:

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)} \tag{2.1}$$

and serves as the base for Bayesian inference, i.e., inferring probabilistic states from evidence.

Bayes' theorem is commonly applied to probabilistic state estimation, which deals with the problem of estimating the probability density function (PDF) over state $x_t$ given all observations $z_{1:t}$ up to time $t$:

$$p(x_t \mid z_{1:t}) \tag{2.2}$$

which is also referred to as the belief $bel(x_t)$.

### 2.1.1 Bayes Filters

Bayes filters are recursive state estimators that solve Equation (2.2) by assuming the investigated dynamic system to be Markovian or a Hidden Markov Model (HMM), respectively. This implies that the current state $x_t$ only depends on the previous state $x_{t-1}$, i.e., it is conditionally independent from earlier states $x_{0:t-2}$:

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}) = p(x_t \mid x_{t-1}). \tag{2.3}$$

The second Markovian assumption states that the current observation $z_t$ only depends on $x_t$ but not on previous observations $z_{1:t-1}$:

$$p(z_t \mid x_{0:t}, z_{1:t-1}) = p(z_t \mid x_t). \tag{2.4}$$

By applying Bayes theorem and using Equation (2.3) and (2.4), we can derive the recursive state update formula fundamental to all Bayes' filter:

$$p(x_t \mid z_{1:t}) = \eta \, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}) \, p(x_{t-1} \mid z_{1:t-1}) \, dx_{t-1} \tag{2.5}$$

where $\eta$ is a normalizing constant, $p(z_t \mid x_t)$ the observation model and $p(x_t \mid x_{t-1})$ the state transition model. If we have access to a control input $u$ that controls the state $x$, Equation (2.5) commonly expands to:

$$p(x_t \mid z_{1:t}, u_{1:t}) = \eta \, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) \, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \, dx_{t-1}. \tag{2.6}$$

The most popular and wide-spread implementation of Bayes filter is the Kalman filter (KF) (Kalman 1960) that works in continuous state space $x_t$ and linear system models assuming normal (or Gaussian) system and measurement noise as well as $x_0$ being normally distributed. The belief is represented in moment representation of the Gaussian (see also Subsection 2.1.4 for more details on normal distributions). For non-linear systems, the extended KF (EKF) can be applied which approximates the non-linearities with a first-order Taylor expansion. Closely related to KF and EKF, we find the information filter (IF) and the extended information filter (EIF) which differ from the previous by their canonical representation.

In contrast to Gaussian filters, non-parametric filters do not model the posterior with a fixed functional form but rather use approximation techniques. Due to their importance for this work, we will review two implementations of non-parametric Bayes Filter with more details starting with the Binary Bayes Filter (which belongs to the family of Histogram Filters) in Subsection 2.1.2 and particle filter (PF) in Subsection 2.1.3.

### 2.1.2 Binary Bayes Filter

The binary Bayes Filter can be applied for estimating a static binary state (whose two possible values are denoted with $x$ and $\bar{x}$) that is observed by a sequence of observations $z_{1:t}$:

$$p(x \mid z_{1:t}) = 1 - p(\bar{x} \mid z_{1:t}) \tag{2.7}$$

The binary property of Equation (2.7) is commonly exploited by applying the log odds ratio:

$$l(x) = \log \frac{p(x)}{1 - p(x)} \tag{2.8}$$

which provides a computationally more stable representation when dealing with edge probabilities close to 0 or 1. The probability of $x_t$ can be recovered from $l_t(x)$ according to:

$$p(x_t \mid z_{1:t}) = 1 - \frac{1}{1 + \exp l_t(x)}. \tag{2.9}$$

A recursive estimator that computes the log odds at time $t$ can then be derived:

$$l_t(x) = \log \frac{p(x \mid z_t)}{1 - p(x \mid z_t)} + l_{t-1}(x) - l_0(x) \tag{2.10}$$

where $l_0(x)$ is the prior probability. $p(x \mid z_t)$ is called the inverse observation model.

### 2.1.3 Particle Filter

The key idea of a PF is to approximate the belief by a set of particles $X_t = \{x_t^{[k]}, w_t^{[k]}\}_{k=1}^{N_k}$ with:

$$bel(x_t) \approx \sum_{k=1}^{N_k} w_t^{[k]} \delta \left( x_t - x_t^{[k]} \right) \tag{2.11}$$

where $x_t^{[k]}$ is the sample of the state space and $w_t^{[k]}$ the (importance) weight of particle $k$ with:

$$w^{[k]} \geq 0 \tag{2.12}$$

and

$$\sum_{k=1}^{N_k} w_t^{[k]} = 1. \tag{2.13}$$

While there exist numerous variants, the most relevant PF algorithm for this thesis is the sampling importance resampling (SIR)-PF algorithm (Rubin 1987), which recursively executes the following three steps:

1. **Sampling**: Particles from $X_{t-1}$ are sampled from the state transition model:

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}). \tag{2.14}$$

2. **Importance Weighting**: The particle weights are computed based on the current observation $z_t$ and the observation model:

$$w_t^{[k]} = p(z_t \mid x_t^{[k]}). \tag{2.15}$$

3. **Resampling**: To force the particles back to the posterior belief, a resampling is executed, i.e., a new set $X_t$ is drawn where the likelihood for a particle to be drawn is proportional to its weight. The weights are then reset equally.

While the resampling step is important to keep the finite number of particles near the target distribution, it can sometimes also lead to an impoverishment of valuable hypotheses. Low-variance resampling (Grisetti, Stachniss et al. 2007) tries to tackle this problem by only executing the resampling step if the variance of the importance weights fall below a threshold. Apart from computing the variance to decide on executing the resampling, the weight computation from Equation (2.15) is altered according to:

$$w_t^{[k]} = w_{t-1}^{[k]}\, p(z_t \mid x_t^{[k]}). \tag{2.16}$$

### 2.1.4 Normal Distribution Parameter Estimation

The normal (or Gaussian) distribution is the most common continuous probability distribution. It models the distribution of a random state variable $x$ of dimension $D$ with the following PDF:

$$p(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma)}} \exp\left[-\frac{1}{2}(x-\mu)\, \Sigma^{-1} (x-\mu)^T\right] \tag{2.17}$$

where we find the mean $\mu$ and covariance $\Sigma$ as the normal distribution's parameters using its moment representation.

For non-static state variables, the previously mentioned KF and EKF can be employed as a recursive state estimator. In the special case of a static system, an unbiased estimation of the normal distribution parameters given a data set of samples $z = \{\zeta_i\}_{i=1}^{N_z}$ can be computed with:

$$\mu = \frac{1}{N_z} \sum_{i=1}^{N_z} \zeta_i \tag{2.18}$$

and

$$\Sigma = \frac{1}{N_z - 1} \sum_{i=1}^{N_z} (\zeta_i - \mu)(\zeta_i - \mu)^T. \tag{2.19}$$

In many real-world applications, we need to deal with continuous data streams instead of complete data sets. Using Equation (2.18) and Equation (2.19) for these kind of applications is inefficient or even impracticable since it requires to store and process all observations.

**Algorithm 2.1** EWMA-RSC update algorithm

```
1: procedure EWMA_RSC(ζ_t, μ_{t-1}, Σ_{t-1}, N_{t-1})
2:     γ_t = N_{t-1} / (N_{t-1}+1)
3:     μ_t = γ_t μ_{t-1} + (1 − γ_t) ζ_t
4:     Σ_t = γ_t Σ_{t-1} + γ_t (1 − γ_t)(μ_t − ζ_t)(μ_t − ζ_t)^T
5:     if N_{t-1} ≤ k_max then
6:         N_t = N_{t-1} + 1
7:     else
8:         N_t = N_{t-1}
9:     end if
10:    return μ_t, Σ_t, N_t
11: end procedure
```

Instead, recursive sample covariance (RSC) update methods iteratively update the normal distribution parameters. An example is the exponentially weighted moving average (EWMA)-RSC update algorithm. It provides a recursive update for $\mu$ and $\Sigma$ based on the current sample point $\zeta_t$:

$$\mu_t = \gamma_t \mu_{t-1} + (1 - \gamma_t)\zeta_t \tag{2.20}$$

$$\Sigma_t = \gamma_t \Sigma_{t-1} + \gamma_t(1 - \gamma_t)(\mu_t - \zeta_t)(\mu_t - \zeta_t)^T \tag{2.21}$$

with:

$$\gamma_t = \frac{N_{t-1}}{N_{t-1} + 1}. \tag{2.22}$$

Pseudocode of the EWMA-RSC update algorithm is given in Algorithm 2.1. Based on $\zeta_t$, current mean $\mu_t$ (line 3) and covariance $\Sigma_t$ (line 4) is updated. To avoid numerical instability, $N_t$ is limited to an upper threshold $k_{max}$ in lines 5–6.

If we need to process more than a single sample point per time step but a set of points $z_t = \{\zeta_i\}_{i=1}^{N_z}$, the algorithm is altered as shown in Algorithm 2.2. First, the mean $\mu_z$ (line 2) and covariance $\Sigma_z$ (line 3) of the current $z_t$ is computed, e.g., using Equation (2.18) and Equation (2.19). Consequently, the normal distribution parameters are updated.

### Gaussian Mixture Models

Gaussian mixture models (GMMs) are an extension of normal distributions to overcome their unimodality. Hence, they can be applied to a wider range of problems. In general, the PDF of a mixture model with $N$ components and model parameters $\theta = \left\langle \{w_\psi, \theta_\psi\}_{\psi=1}^{N} \right\rangle$ is defined as:

$$p(x \mid \theta) = \sum_{\psi=1}^{N} w_\psi \, p(x \mid \theta_\psi). \tag{2.23}$$

**Algorithm 2.2** EWMA-RSC update algorithm with multiple points

**procedure** EWMA_RSC_MULT($z_t, \mu_{t-1}, \Sigma_{t-1}, N_{t-1}$)
  $\mu_z = \frac{1}{N_z} \sum_{i=1}^{N_z} \zeta_i$
  $\Sigma_z = \frac{1}{N_z - 1} \sum_{i=1}^{N_z} (\zeta_i - \mu)(\zeta_i - \mu)^T$
  $\gamma_t = \frac{N_{t-1}}{N_{t-1} + N_z}$
  $\mu_t = \gamma_t \mu_{t-1} + (1 - \gamma_t)\mu_z$
  $\Sigma_t = \gamma_t \Sigma_{t-1} + \gamma_t (1 - \gamma_t)(\mu_t - \mu_z)(\mu_t - \mu_z)^T$
  **if** $N_{t-1} \leq k_{max}$ **then**
    $N_t = N_{t-1} + N_z$
  **else**
    $N_t = N_{t-1}$
  **end if**
  **return** $\mu_t, \Sigma_t, N_t$
**end procedure**

where $\theta_\psi$ are the model parameters of the $\psi^{th}$ component and $w_\psi$ its weight which is defined as:

$$w_\psi \geq 0 \tag{2.24}$$

and

$$\sum_{\psi=1}^{N} w_\psi = 1. \tag{2.25}$$

A GMM is a special form of a mixture model where $\theta_\psi = \langle \mu_\psi, \Sigma_\psi \rangle$ and $p(x \mid \theta_\psi)$ is modeled using the Gaussian PDF of Equation (2.17).

A common approach for estimating the GMM parameters based on a given data set $z = \{\zeta_i\}_{i=1}^{N_z}$ is the expectation maximization (EM) (Dempster and Laird 1977) algorithm. It recursively estimates the parameters $\theta^{[\phi]}$ of the GMM in two steps with $\phi$ being the current iteration. In the expectation step, the posterior probability of data point $\zeta_i$ being produced by component $\psi$ is computed with:

$$\pi_\psi^i = p(\psi \mid \zeta_i, \theta^{[\phi-1]}) = \frac{w_\psi^{[\phi-1]}\, p(\zeta_i \mid \theta_\psi^{[\phi-1]})}{\sum_{j=1}^{N} w_j^{[\phi-1]}\, p(\zeta_i \mid \theta_j^{[\phi-1]})} \tag{2.26}$$

Consequently, in the maximization step, $\theta^{[\phi]}$ is updated with:

$$w_\psi^{[\phi]} = \frac{1}{N_z} \sum_{i=1}^{N_z} \pi_\psi^i \tag{2.27}$$

$$\mu_\psi^{[\phi]} = \frac{\sum_{i=1}^{N_z} \zeta_i \pi_\psi^i}{\sum_{i=1}^{N_z} \pi_\psi^i} \tag{2.28}$$

**Algorithm 2.3** EM algorithm for estimating a GMM from a complete data set

**procedure** $EM\_GMM\_estimation(z, \theta^0, k_\epsilon)$
  $L^{[\phi-1]} = 0$
  **repeat**
    **for all** $\zeta_i$ **do**
      $\pi_\psi^i = \frac{w_\psi \, p(\zeta_i \mid \theta_\psi)}{\sum_{j=1}^{N} w_j \, p(\zeta_i \mid \theta_j)}$
    **end for**
    **for all** mixture components $\psi$ **do**
      $s_\psi = \sum_{i=1}^{N_z} \pi_\psi^i$
      $w_\psi = \frac{1}{N_z} s_\psi$
      $\mu_\psi = \frac{1}{s_\psi} \sum_{i=1}^{N_z} \zeta_i \pi_\psi^i$
      $\Sigma_\psi = \frac{1}{s_\psi} \sum_{i=1}^{N_z} (\zeta_i - \mu_\psi^{[\phi]})(\zeta_i - \mu_\psi^{[\phi]})^T \pi_\psi^i$
      $L_\psi = \sum_{i=0}^{N_z} \log p(\zeta_i \mid \theta_\psi)$
    **end for**
    $L^{[\phi]} = \sum_{i=0}^{N} L_\psi$
  **until** $(L^{[\phi]} - L^{[\phi-1]}) \leq k_\epsilon$
  **return** $\theta$
**end procedure**

$$\Sigma_\psi^{[\phi]} = \frac{\sum_{i=1}^{N_z} (\zeta_i - \mu_\psi^{[\phi]})(\zeta_i - \mu_\psi^{[\phi]})^T \pi_\psi^i}{\sum_{i=1}^{N_z} \pi_\psi^i} \tag{2.29}$$

To check for convergence, the log likelihood $L^{[\phi]}$ of the data set $z$:

$$L^{[\phi]} = \sum_{i=0}^{N_z} \log p(\zeta_i \mid \theta^{[\phi]}) \tag{2.30}$$

is commonly evaluated. Once the difference between the log likelihoods of succeeding iterations does not exceed a threshold $k_\epsilon$, the iteration process is stopped.

Pseudocode is given in Algorithm 2.3. Note that the given formulas assume that the number of mixture components $N$ is constant and known a priori to initialize the parameter set with the prior $\theta^0$. However, in most applications, we do not know the ideal number of components but need to find it altogether with the normal density parameters. A common way to approach this is using a split and merge (SM)EM, see for instance (Ueda, Nakano et al. 1998) as an exemplary work. The basic idea is to check at the end of each iteration if two components can be merged into one or if a component can be split into two, based on some application specific SM criteria.

While Algorithm 2.3 estimates the GMM from a complete data set, it is not suitable for online estimation problems. Same as for the EWMA-RSC, a recursive update algorithm is needed for these kind of applications. In general, there exist several approaches which tackle this issue under different constraints. An example highly relevant for this work is

the incremental Gaussian mixture model (IGMM) algorithm for unsupervised incremental learning of GMMs presented in (Engel and Heinen 2010).

The IGMM algorithm is initialized at $t_0$ with a single component using the first incoming data point $\zeta_0$:

$$\theta_0 = \langle \{ \mu_0 = \zeta_0 \ , \ \Sigma_0 = \sigma_0 I \ , \ w_0 = 1 \} \rangle \tag{2.31}$$

where $\sigma_0$ is the initial variance and $I$ the identity matrix.

After this initialization, all succeeding data points are either used to update the existing components or to create a new component. The decision is made based on the novelty of the data point using the criterion:

$$p(\zeta_t \mid \mu_\psi, \Sigma_\psi) < \frac{k_{nov}}{(2\pi)^{\frac{D}{2}} \det(\Sigma_\psi)^{\frac{1}{2}}} \ \ \forall \ \psi \tag{2.32}$$

with the user-defined minimum probability parameter $k_{nov}$. If Equation (2.32) holds true for all components $\psi$, a new component $\psi + 1$ is created with initial values $\{\mu_{\psi+1} = \zeta_t, \Sigma_{\psi+1} = \sigma_0 I, w_{\psi+1} = w_0\}$ where $w_0$ is another user-defined parameter. Otherwise, $\zeta_t$ is used to update the existing components. This is done by first computing the posterior probabilities $\pi_\psi^t$ using Equation (2.26). Afterwards, $\mu_\psi$ and $\Sigma_\psi$ are updated using Equation (2.20) and (2.21) where the computation of $\gamma_\psi^t$ is altered to:

$$\gamma_\psi^t = \frac{s_\psi^{t-1}}{s_\psi^{t-1} + \pi_\psi^t} \tag{2.33}$$

with $s_\psi$ as the sum over the posterior probabilities:

$$s_\psi^t = s_\psi^{t-1} + \pi_\psi^t. \tag{2.34}$$

The update of the component's weight concludes the update step:

$$w_\psi^t = \frac{s_\psi^t}{\sum_{i=1}^{N_\psi} s_i^t}. \tag{2.35}$$

Note that the notation of the given formulas differ from the original algorithm presented in (Engel and Heinen 2010) in order to align it with the overall notation used in this chapter and throughout this work. The complete IGMM algorithm is given in Algorithm 2.4.

**Algorithm 2.4** IGMM algorithm for recursively estimating a GMM from a data stream

**procedure** $IGMM\_update(\zeta_t, \theta_{t-1})$
  **for all** mixture components $\psi$ **do**
    $p_\psi = p(\zeta_t \mid \mu_\psi, \Sigma_\psi)$
  **end for**
  **if** $p_\psi < k_{nov}(2\pi)^{\frac{-D}{2}} det(\Sigma_\psi)^{-\frac{1}{2}} \; \forall \; \psi$ **then**
    $\theta_t = initialize\_new\_component(\zeta_t, \theta_{t-1})$
    **return** $\theta_t$
  **end if**
  $p_{sum} = \sum_{\psi=1}^{N_{gmm}} w_\psi^{t-1} \; p_\psi$
  $s_{sum} = 0$
  **for all** mixture components $\psi$ **do**
    $\pi_\psi^t = \frac{1}{p_{sum}} w_{\psi^{t-1}} \; p_\psi$
    $\gamma_\psi^t = s_\psi^{t-1}(s_\psi^{t-1} + \pi_\psi^t)^{-1}$
    $\mu_t = \gamma_t \mu_{t-1} + (1 - \gamma_t)\zeta_t$
    $\Sigma_t = \gamma_t \Sigma_{t-1} + \gamma_t(1 - \gamma_t)(\mu_t - \zeta_t)(\mu_t - \zeta_t)^T$
    $s_\psi^t = s_\psi^{t-1} + \pi_\psi^t$
    $s_{sum} = s_{sum} + s_\psi^t$
  **end for**
  **for all** mixture components $\psi$ **do**
    $w_\psi^t = \frac{1}{s_{sum}} s_\psi^t$
  **end for**
  **return** $\theta_t$
**end procedure**

## 2.2 Navigation of Mobile Robots

The task of MR navigation aims for controlling the mobile platform to reach a desired goal pose, i.e., position and orientation of the MR, considering further constraints, e.g., desired velocities or given one or several intermediate poses. The goal pose and the path constraints are often summarized as the mission. Choosing a reasonable control scheme strongly depends on the overall navigation approach, e.g., available sensors and desired autonomy capabilities. For instance, line-guided AGVs, in general, only need to sense the lateral derivation of the AGV with respect to the guided path in combination with a controller to keep the derivation to a minimum while following the path. For MR with higher navigation capabilities relevant for this work, there exists a wide spread and de-facto standard control scheme which is applied to most of the current solutions of highly autonomous MRs (Siegwart and Nourbakhsh 2004), see Figure 2.1 for an illustration. The control scheme basically consists in a closed-loop control of the mobile platform with the perception modules on the left half and the motion control modules on the right half of Figure 2.1. On the bottom of the perception side, we find a set of sensors measuring the environment (see also Subsection 2.2.1 for a description of commonly used sensors) as the

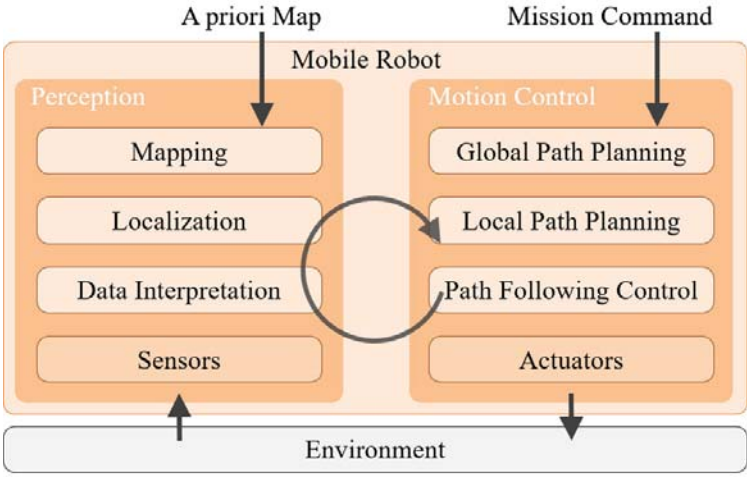



Figure 2.1: Map-based control scheme for navigation of MRs, freely adapted from Siegwart and Nourbakhsh (2004).

input of a sensor data preprocessing and interpretation module. The preprocessed sensor data is then used for localizing the MR within the environment, i.e., estimating the pose with respect to some fixed reference frames given an a priori known environment map which will be further described in Subsection 2.2.3. Finally, we find the mapping module which builds and updates an environment model based on the newly gathered data. On the planning side, we find the path planning module which is responsible to compute the robot's path based on its current localization and the commanded mission. Path planning is usually divided into global and local path planning to meet the contrary requirements of optimal paths and highly reactive behavior, further discussed in Subsection 2.2.7. Finally, path control is responsible to precisely control the MR on the desired path by computing steering and motion commands as the input for the actuators. The overall control loop cascades into several low to high level control loops where, in general, we find a bottom-up increase of processed information as well as spatial/temporal horizons and a decrease of control rates.

### 2.2.1 Sensors

The set of sensors used for navigation has a major impact on the design of a suitable navigation system. Therefore, we will briefly discuss some relevant sensors with respect to our target application, i.e., MRs in industrial applications. For a more detailed discussion, the reader is referred to Siegwart and Nourbakhsh (2004) or Siciliano and Khatib (2016).

In general, we can distinguish between proprioceptive sensors and exteroceptive sensors. Proprioceptive sensors measure the internal state of the MR, e.g., battery status, load, joint angles, etc. The most important proprioceptive sensors for navigation are wheel

encoders used for odometry calculation and inertial measurement units (IMUs) to measure translational as well as angular velocities.

Exteroceptive sensors make observations of the robot's environment. Most relevant for our application are Lidar and vision-based sensors. Lidar sensors (or laser rangefinders) belong to the category of time of flight (TOF) sensors which basically are able to determine the distance to objects within the sensor's field of view by transmitting electromagnetic waves and measuring the time until they return. Some sensors additionally provide information about the intensity of the reflected beam which indicates the reflectivity of the object. 2D Lidars commonly use a rotation mechanism to generate 2D scans with a field of views up to 360 degrees. They are commercially available as safety devices for obstacle detection used for collision avoidance of the MR. For assuring safety, i.e., assuring the detection of any obstacle with a defined minimum diameter in some safety distance around the MR, the laser beam is expanded. While this property is important for safety functionality, it adversely affects the accuracy of the range measurements. Moreover, due to their primary safety task, safety Lidars are usually mounted at leg height so that objects on this height cannot be overseen by the sensors. This position, however, is unsuitable for localization which for most applications prefers to have the sensor possibly high above the ground in order to improve the sight and observing less dynamic or changing objects. While these circumstances makes this sensor unsuitable for localization, there is an important advantage which aligns with the requirements needed within this work in terms of economic efficiency. Since safety scanners are nowadays the standard sensors mounted to achieve collision safety of the MR, this sensor comes with almost no extra cost when used for navigation.

An expansion of the 2D version are multi-layered or 3D Lidars which generate several 2D scans with different tilting angles resulting in a more or less dense 3D pointcloud. Available solutions often also provide higher sensing ranges making them the currently predominant sensors in the field of autonomous driving despite high purchasing prices. Although less expensive versions may follow in the near future, the full potential of those sensors is only exploited when mounting them at an appropriate position (e.g. for autonomous driving mainly on top of the car) which is often not possible for ATVs due to the constructional constraints induced by the transport tasks, i.e., carrying loads on top of the vehicle.

Vision-based sensors are further exteroceptive sensors suitable for navigation providing rich information about the environment. Most basic single camera systems deliver 2D images usually used for object detection and classification or place recognition. The disadvantage of those sensors lies in the missing depth information. To overcome this, depth information can be gathered from motion with some additional computation effort. Alternatively, stereo camera systems or TOF cameras are able to deliver that information. However, in general, the accuracy and range of the depth information is far lower than that of Lidar sensors.

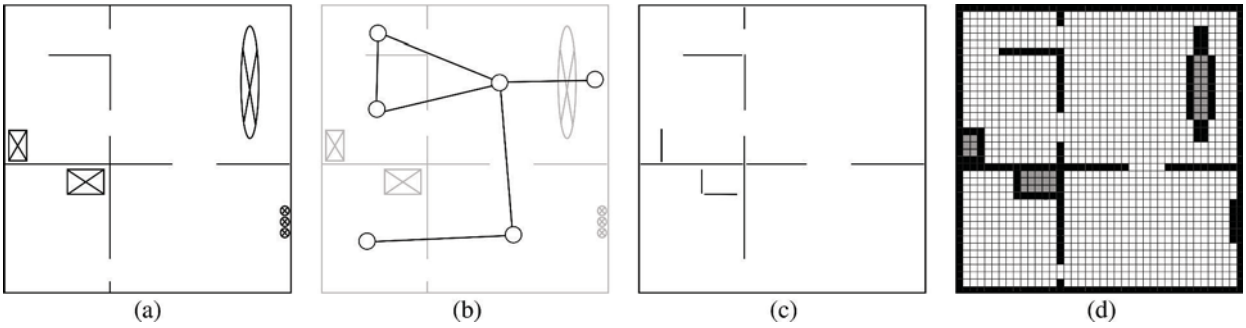



Figure 2.2: Visualization of different map types: (a) real environment with walls and furniture, (b) topological map indicating room connections, (c) landmark map using line landmarks, (d) occupancy grid map.

Further exteroceptive sensors worth mentioning are Global Positioning System (GPS) or IPS[1] (colloquially also known as Indoor-GPS), RFID and ultrasonic (US) sensors.

### 2.2.2 Map Representations

In the map-based control scheme depicted in Figure 2.1, the map is a fundamental component. The localization module relies on it for self-referencing (see Subsection 2.2.3) and the path planning module needs it to search for optimal paths (see Subsection 2.2.7).

In general, a suitable map representation of the environment needs to fulfill the following requirements:

1. The map accuracy and level of detail must fit to the modules' requirements (e.g. the achievable localization accuracy strongly correlates with the map accuracy).

2. The map type must fit to the available sensor data.

3. Since the map complexity has a deep impact on computational and memory demands, it needs to be aligned with the approaches of the modules using the map and available hardware resources.

Most relevant map types can be divided into topological, landmark and grid maps. Topological maps represent the environment by a graph with nodes and edges describing certain locations of the environment and their connectivity to others, also known as a roadmap graph. In that way, they offer a compact representation with sparse geometrical information. Due to this property, they are mainly applied to path planning (LaValle 2006) or combined with a geometrical representation (Kuipers and Byun 1991) (Abraham, Ge et al. 2009). Topological maps can be automatically constructed from geometrical maps, e.g., using Voronoi diagrams (Thrun and Bücken 1996).

[1] Systems to locate the objects in indoor environments commonly using different kind of radio waves (see Curran, Furey et al. (2011) for an overview)

Landmark (or feature) maps provide continuous representations of the environment described by a set of spatially located landmarks. Landmark types are defined by selecting a certain property of objects or structures within the environment which are observable by the used sensors. Common landmark types use the geometric shape (e.g. lines, corners, planes), reflectivity, color or other observable properties. Examples of landmark maps are line maps (Connette, Meister et al. 2007) or polygon maps (Latombe 1991) used for Lidar-based navigation in buildings, or SIFT (Lowe 2004) features for vision-based localization and SLAM (Se, Lowe et al. 2002) (Endres, Hess et al. 2012). The advantage of landmark maps lies in their compact and yet accurate description which enables an efficient processing, also for large-scale environments. However, the landmark definition requires a certain structure of the environment that is a priori known and extractable from sensor observations which limits the applicability of this representation. Furthermore, when used for localization, landmarks need to be uniquely identified, a process commonly referred to as data association.

Grid maps provide a discretized decomposition of the environment into a set of 2D or 3D fixed size cells. Most relevant for MR navigation are occupancy grid maps (OGMs) (Elfes 1989) where each cell is assigned a binary variable holding information about the occupancy of the cell. OGMs are a commonly used map representation for Lidar-based localization and SLAM. Moreover, they serve as the base to construct costmaps for path planning. Costmaps expand the OGM by bringing in additional information based on the robot's geometry or other path planning relevant contexts, see, e.g., Lu, Hershberger et al. (2014). Due to their ability to offer accurate geometrical maps of arbitrary shaped environments and their versatility to be used by both localization and path planning, grid maps nowadays gain great popularity. However, the accuracy firmly relates with the resolution of the grid maps, i.e., the size of the cells. High resolutions ensure good accuracies but significantly increase computational and memory demands, especially in large-scale environments. Approaches tackling this issue usually remove the fixed cell property to reduce the number of cells in homogeneous areas of the environment, e.g., quadtree (QT) OGM (Kraetzschmar, Pagès Gassull et al. 2004).

Finding a suitable map type for the navigation system strongly depends on the approach chosen for localization and path planning discussed in the next sections. Since those approaches may impose different requirements on the map representation, it is not uncommon to use different map types for the different modules of the navigations systems. For instance, a localization approach may depend on an accurate metric description, e.g., in terms of a high-resolution OGM, while a path planner may rely on a more compact description which reduces its search space in order to achieve suitable planning cycle times.

### 2.2.3 Localization

A suitable localization module needs to compute the pose of the robot with respect to a fixed reference frame of the environment with sufficient accuracy. The localization of the robot is the most basic prerequisite information for the path planner to be able to fulfill its task. It is not only needed to find a feasible path from the current location to the desired goal but also for continuously being able to precisely follow the computed path during path execution. The path following control responsible for this task thereby needs high-frequency[2] localization updates in order to properly work.

Odometry or dead reckoning is the most basic method used to tackle the localization problem of MRs. It uses proprioceptive sensors to infer the robot's pose to a given initial pose by integrating over the internal states (velocities and accelerations) of the robot. Due to sensor noise, slip as well as inaccuracies between configured and actual wheel diameters, the localization error of this method accumulates over time leading to localization drifts that cannot be recovered. Therefore, this method can rarely be used as a stand-alone localization system but rather to overcome spatial or temporal gaps when no other sensor information is available for localization. Furthermore, map-based approaches pursue this idea and commonly use odometry to have a frequent and prior estimate of the robot's pose relative to the estimate of the previous time step.

If we assume the environmental map is a priori known, the main challenges of localization relative to this map are given in term of sensor noise and sensor aliasing. Since there is no perfect sensor, each sensor data contain some noise resulting from the measurement method itself and data processing. Sensor aliasing describes the incompleteness or ambiguity of the sensor information, i.e., even with noise-free sensor information, a single sensor reading generally contains not enough information to be indistinguishable mapped to a unique state or in case of the localization problem, to a unique pose in the environment. Generally, vision-based sensors appear to have a lower sensor aliasing due to their information depth in contrast to Lidar sensors where, e.g., symmetrical geometries of the environment (e.g. in buildings) make the respective sensor readings indistinguishable from one another. State-of-the-art localization approaches tackle this problem mainly in two ways. First of all, the localization is usually based not only on one sensor input but by fusing different sensor information. Secondly, probabilistic approaches are applied to handle the uncertainty and incompleteness of the data.

The Bayes filter described in Subsection 2.1.1 is the predominant approach for the localization problem which is estimating the pose $x_t$ of the MR given current sensor observation $z_t$, odometry $u_t$ and an a priori known map $m$ of the environment. For formulating the

[2]Depending on the robot's velocities, usually around 100 Hz

pose belief estimation, Equation (2.6) is enhanced by introducing a dependency of the observation model on $m$:

$$p(x_t \mid z_t, u_t, m) = p(z_t \mid x_t, m) \int p(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{t-1}, u_{t-1})\, dx_{t-1}. \tag{2.36}$$

In this application, the transition model $p(x_t \mid x_{t-1}, u_t)$ is referred to as the motion model. Note that the map $m$ is assumed to be static, an assumption, we will skip later on in Subsection 2.2.5.

First approaches commonly relied on EKF (see, e.g., (Cox 1991), (Durrant-Whyte 1994), (Forsberg, Larsson et al. 1995)) due to their computational efficiency and ease of implementation. The basic strategy is to extract landmarks (e.g. lines) from the raw sensor data and match those to a given landmark map. While those landmark-based approaches stand out in terms of computational and memory efficiency, the main drawback is that they depend on the structure of the environment and on the availability of extractable landmarks that can be robustly matched against the map. Additionally, the performance of the filter depends on how good the linearization of the EKF is able to approximate non-linearities of the applied motion and observation model. These factors together with an increased availability of computational power and memory has displaced the EKF as the predominant approach by PF-based localization approaches, introduced by Dellaert, Fox et al. (1999) and Thrun, Fox et al. (2001) as Monte-Carlo localization (MCL).

The ability to incorporate arbitrary linear and non-linear motion and observation model altogether with some computational improvements has made MCL as the nowadays most used localization approach, at least when using Lidar as the primary sensor input source. MCL is suitable to be used in combination with OGMs (see Subsection 2.2.2) and thereby overcomes the inflexibility of landmark-based approaches discussed in the last section.

MCL basically uses the SIR-PF algorithm from Subsection 2.1.3 for localization where $x_t$ represents the pose of the robot. Sampling is performed with the motion model incorperating current odometry information:

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t). \tag{2.37}$$

Afterwards, the importance weights are computed with the sensor observation model which incorperates the map:

$$w_t^{[k]} = p(z_t \mid x_t^{[k]}, m). \tag{2.38}$$

A further enhancement of MCL was introduced by Fox (2003) with Kullback-Leibler divergence (KLD)-sampling to adjust the particle set size dynamically based on the current pose belief.

Note that so far, we did not discuss the initialization of the prior $p(x_0)$ or initial pose belief. Having no prior knowledge about the robot pose is commonly referred to as global localization. MCL is in principle able to resolve this by using a high number of particles uniformly distributed over the environment. After some time steps, the particles converge to the area around the actual pose of the robot. From this moment on, the pose estimation results in a tracking problem where significantly less particles are needed. If the initial pose is known with limited uncertainty, the localization can directly run in tracking mode. Although MCL covers both modes, the computational intensity during global localization gets quite high, especially in large-scale environments, due to the high amount of needed particles. Since for most industrial applications, receiving an initial pose in the initialization phase of the robot can simply be integrated in the overall logistic process, global localization is of minor interest and will therefore not be investigated explicitly in this work.

### 2.2.4 Mapping

In Subsection 2.2.3, we defined the map to be a priori known to the localization algorithm. Sometimes the map can be generated manually from building plans. However, this requires the availability of respective data with sufficient accuracy as well as some effort to transform them into the desired map type.

Another approach is to infer the map from sensor data. For the same reason as for the localization problem formulation (i.e., sensor noise and aliasing), the mapping problem is commonly formulated as a probabilistic problem in terms of computing the belief of the map $m$ given a set of observations $z_{1:t}$ and knowing the robot's trajectory $x_{1:t}$ (an assumption that is skipped in the next section):

$$p(m \mid z_{1:t}, x_{1:t}). \tag{2.39}$$

Solutions for Equation (2.39) depend on the map type of $m$.

Due to the relevance throughout this work, we will describe one of the most fundamental mapping algorithms which is occupancy grid mapping. As described in Subsection 2.2.2, an OGM models the environment with a finite set of cells $m = \{c_i\}$. Each cell is assigned the binary occupancy value $q_i = \{f, o\}$ of either being free or occupied. Since the cells are defined as conditionally independent, the problem of estimating $m$ can be broken down into estimating $p(q_i \mid z_{1:t}, x_{1:t})$ for each cell. Furthermore, estimating the binary occupancy value from sensor data can now be carried out by the recursive formula of the binary Bayes filter described in Subsection 2.1.2:

$$l_t(q_i) = \log \frac{p(q_i \mid z_t, x_t)}{1 - p(q_i \mid z_t, x_t)} + l_{t-1}(q_i) - l_0(q_i) \tag{2.40}$$

The crucial part of this approach is to design a suitable inverse observation model for the applied sensor. A simple example for ranging sensors is given in Thrun, Fox et al. (2005).

### 2.2.5 SLAM

In the last section, we made the assumption of the robot's complete trajectory $x_{1:t}$ to be known when we formulated the probabilistic approach for mapping in Equation (2.39). This assumption is rarely met, especially for MR applications. SLAM approaches tackle this issue by integrating the estimation of $x_{1:t}$ into the estimation problem.

The probabilistic formulation of the SLAM problem is given with:

$$p(m, x_{1:t} \mid z_{1:t}, u_{1:t}). \tag{2.41}$$

Eq. (2.41) integrates the belief over the whole trajectory $x_{1:t}$ which is referred to as full (or offline) SLAM whereas in online SLAM only the most recent pose is relevant:

$$p(m, x_t \mid z_{1:t}, u_{1:t}). \tag{2.42}$$

Consequently, approaches tackling the SLAM problem can also be divided into these two categories.

The most relevant offline SLAM approach is GraphSLAM introduced by Thrun and Montemerlo (2006). The basic idea is to model the robot's trajectory $x_{1:t}$ as nodes into a single graph. Edges in the graph are then incorporated in terms of constraints induced by corresponding sensor observations $z_{1:t}$ using the observation model as well as odometry information $u_{t-1}$ using the motion model. Solving the SLAM problem then becomes an optimization problem, i.e., minimizing the error of the constraints introduced by the edges in the graph. Since the graph can grow really large, especially in large-scale environments, sparsification techniques are applied to reduce the density of the graph (see, e.g., the work of Lu and Milios (1997)) to reduce computational demands. Still, graph-based approaches remain computational intensive and are therefore only usable for offline map building. Recent advances tackle this problem by separating it into a global and local optimization problem. The local optimizer builds local maps with a subset of observations and provides frequent localization updates whereas the global optimizer aligns the local maps into a global map and executes loop closure, see for instance (Konolige, Grisetti et al. 2010) and (Hess, Kohler et al. 2016). Loop closure herein names a subproblem of SLAM to recognize previously visited locations after the robot and is of crucial importance for generating consistent global maps in large-scale environments.

Filter-based approaches are predominant for tackling the online SLAM problem. EKF-SLAM was here the first solution not only for online SLAM but for SLAM in general

introduced by Smith and Cheeseman (1986). EKF-SLAM basically extends the EKF localization approach by adding the set of map features into the state space. The properties are thereby similar to EKF localization described in Subsection 2.2.3, i.e., computational efficiency and ease of implementation while the drawbacks of used feature maps remain in terms of reduced applicability and data association problems. Additionally, the computational complexity grows quadratically with the number of features making it in general impracticable for large-scale environments.

Applying PFs to SLAM was introduced by Murphy (2000a) and is nowadays known as FastSLAM. A landmark-based solution was first shown by Montemerlo, Thrun et al. (2002) whereas Hahnel, Burgard et al. (2003) describes a solution based on grid maps. Further improvements have been made by (Grisetti, Stachniss et al. 2007). Since PFs are unsuitable for high dimensional state spaces, a technique known as Rao-Blackwellized PF (RBPF) is applied to factorize Equation (2.42) into:

$$p(m, x_t \mid z_{1:t}, u_{1:t}) = p(x_t \mid z_{1:t}, u_{1:t})\, p(m \mid z_{1:t}, x_{1:t}). \tag{2.43}$$

This factorization drastically eases the problem since it separates into a localization problem $p(x_t \mid z_{1:t}, u_{1:t})$ and a mapping problem $p(m \mid z_{1:t}, x_{1:t})$. Practically, this is implemented by each particle holding its own map estimate $m^{[k]}$:

$$X_t = \{u_{t-1}^{[k]}, w_t^{[k]}, m^{[k]}\}_{k=1}^{N}. \tag{2.44}$$

and has led to first practical implementations of PF-based online SLAM. However, the computational effort of this approach still strongly correlates with the number of particles. When dealing with noisy odometry information, a high number of particles is crucial in order to have a sufficient density of particles around the robot's actual pose after sampling from the proposal distribution $p(x_t \mid x_{t-1}^{[k]}, u_t)$. To overcome this complementary requirement, FastSLAM 2.0 (Montemerlo, Thrun et al. 2003) uses a proposal distribution $p(x_t \mid z_t, x_{t-1}^{[k]}, u_t)$ which directly incorporates the current observation $z_t$.

### 2.2.6 Long-Term SLAM

In Subsections 2.2.3–2.2.5, we assumed the map to be static (thus so far $m$ had no dependency on $t$). However, only a rare amount of environments consist of only static objects but rather include a varying amount of semi-static and dynamic objects. Nevertheless, the so far described SLAM approaches have shown good results in a variety of environments which also contain non-static components. The good results can be explained with the primary usage of SLAM as an initial mapping tool executed with data gathered from a single run of the MR through the environment. Within these temporary limited observations, usually

only occasional changes of the semi-static environment occur while observations of dynamic objects can be treated as outliers up to a certain degree. Similarly, the described localization approaches can compensate minor changes of the environment as long as the map contains enough valid components in each location the MR is operating. Obviously, these properties do not hold for any type of environment and application. As described in Section 1.1, in some scenarios, the MR needs to deal with highly changing environments where major changes can also occur in short time periods. Moreover, if we relax the temporal condition, i.e., build the map or perform localization over a longer time period, also low-changing environments are likely to exhibit major changes which can impose significant problems to the so far described localization and SLAM approaches. Long-term SLAM (also known as long-term localization or lifelong localization) tackles the problem of localization and SLAM for long-term operations of MRs in changing environments. Since the MR needs to provide a robust localization based on a map that is changing, it can be seen as an intermediate problem between localization and SLAM. Consequently, approaches tackling this problem build open existing approaches of localization and SLAM in combination with a map representation that is able to model the map dynamics. A detailed discussion of recent approaches will be given Subsection 5.1.2.

### 2.2.7 Path Planning and Control

Although path planning is not the scope of this work, this section shortly reviews the basics of the field relevant for the system design of the overall cloud-based navigation system presented in Chapter 3 and to understand the requirements of localization and mapping for path planning. For an elaborated overview of path planning techniques, the reader may be referred to (Latombe 1991), (LaValle 2006) or (Siciliano and Khatib 2016).

Robot motion planning aims at finding a set of configurations given the current configuration and a desired goal configuration by considering the optimality criteria (e.g. time, distance to obstacles, motion smoothness, etc.). The set of all possible robot configurations for an environment is called the configuration space. For 2D motion planning of MRs, the configuration is usually defined as the robot's pose $(x, y, \theta)$. The most basic motion planning problem is then defined as finding the shortest path from the current location to the desired goal without colliding with any objects of the environment, commonly denoted as path planning. Depending on the size of the environment and the chosen map representation, the search space of the path planning algorithm quickly grows to large scale. Therefore, a wide-spread strategy is to divide the path planning problem into global and local path planning. In this concept, global path planning computes complete paths from start to goal but is only executed when a new goal is received or when the local planner is not able to resolve a local conflict. Additionally, global path planning often operates on a reduced search space (e.g. a topological map) to speed up computation. The local path

planner optimizes the global path during runtime and performs dynamic collision avoidance based on recent sensor observations on a limited spatial and temporal horizon. The optimized local path is then passed to the path following controller to compute control inputs for the actuators.

## 2.3 Multi-Robot Systems

So far, the presented concepts and approaches only considered single robot systems. However, in many applications of MRs, several robots operate within the same environment. Examples of those multi-robot systems can be found in assembly (Simmons, Singh et al. 2001), search and rescue (Murphy 2000b) or household (Parker 2003). In these applications, cooperation can be either a necessity or a system design choice. Exemplary, in the scenario of two MRs sharing a narrow workspace while performing completely separated tasks, cooperation is needed in order to avoid mutual blocking whereas for two robot manipulators cooperation could be a choice in order to accomplish an assembly task that a single robot would not be capable of solving.

### 2.3.1 Cooperative Robotics

Following Siciliano and Khatib (2016), cooperative multi-robot systems can be divided into collective swarm systems and intentionally cooperative systems. Collective swarms are defined as large number of homogeneous, physically rather simple robots which have marginal knowledge about their team members. In contrast, intentionally cooperative systems are characterized as teams of (potentially) heterogeneous robots with profound knowledge about their team members which act together to achieve a common goal. Heterogeneity can appear in terms of different sensors, actuators or computational hardware.

Cooperation bases on a certain amount of communication among the robots and can be realized implicit (e.g. by sensing states and actions of other robots) or explicit (e.g. by exchanging information using wireless networking). Deciding on the optimal amount of communication is a key challenge in the design of cooperative systems and will be further discussed in Subsection 2.3.2.

Most common architectures for cooperative robots can be classified into centralized, hierarchical, decentralized, distributed and hybrid. Centralized architectures exhibit a single point of controlling the multi-robot system. While this is often impractical in terms of single point of failure and communication issues for real-time control, their advantage usually consists in the global knowledge of the overall system that the centralized controller has access for decision making and coordination of the robots. In contrast, hierarchical architectures exhibit hierarchical control levels, usually with a top-down decrease of number of

robots controlled by the respective layer. In decentralized architectures, the robots largely control their actions by themselves involving mutual negotiation with spatially co-located robots. While this architecture removes the problem of single point of failures, global coherent solutions can rarely be achieved. Finally, hybrid architectures aim at combining the different global and local control levels to leverage their advantages.

### 2.3.2 Networked Robotics

Networked robots are a special form of cooperative multi-robot systems where an explicit communication among the robots takes place through a wired or wireless network. The network may also involve stationary sensors, embedded hardware or human users. Exchanging information by networked communication opens up completely new possibilities to achieve capabilities that the single robot cannot achieve through coordination, extended sensing and remote computation. For instance, MRs can react to changes of the environment perceived by other MRs at a remote location.

The main challenge when designing a networked robot system is to find the optimal extent of information communicated through the network that maximizes the performance of the robots and considering communication properties like network bandwidths, delays and disruptions.

In the context of MR navigation, networking approaches can basically be categorized into coordination and cooperative path planning as well as cooperative perception and localization. In the following, some examples of approaches in this field are mentioned while a more detailed review of recent approaches relevant for this work will be given in Subsection 5.1.2. Task allocation and global path planning of the MRs are often carried out from a central controller, see for instance (Gerkey and Mataric 2002)(Luna and Bekris 2011)(Sharon, Stern et al. 2015) while cooperative local path planning and collision avoidance are more subject to distributed or hierarchical or hybrid architectures (Gregoire, Bonnabel et al. 2013) (Frese and Beyerer 2011). For networked perception, relevant subjects are cooperative exploration and SLAM (Burgard, Moors et al. 2000)(Atanasov, Le Ny et al. 2015), cooperative localization (Howard, Matark et al. 2002) and object tracking (Sabattini, Cardarelli et al. 2015).

### 2.3.3 Cloud Robotics

Cloud Robotics extends the traditional networked robotics described in Subsection 2.3.2 by applying cloud computing, Big data as well as further IoT-technologies and was first introduced by Kuffner (2010).

NIST defines cloud computing as *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks,*

*servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."* (Mell and Grance 2009). Further, there exist different deployment models:

- **Private cloud**: A cloud infrastructure for exclusive usage by a single organization (which is operated by the organization itself or a third party).
- **Public cloud**: A cloud infrastructure for open usage by the public.
- **Community cloud**: A cloud infrastructure for exclusive usage by members of a community.
- **Hybrid cloud**: A cloud infrastructure which inhibits a combination of private, public and community clouds.

Cloud robotics tries to exploit the massive computational, storage and communication resources provided by the cloud. More concretely, it enables the pooling of massive, remotely located multi-robot systems and other users (e.g. humans or other machines and sensors) to access and share knowledge which would be impracticable for traditional networks. Furthermore, more complex and large-scale parallel computing applications for perception, coordination and control of several robots can be applied by overcoming the limited and expensive local computational and memory resources.

RoboEarth (RoboEarth 2018) can be seen as the first implementation of a cloud robotics platform. Since then, numerous applications of robots with extended or new capabilities using cloud platforms have been successfully demonstrated, e.g object grasping (Kehoe, Matsukawa et al. 2013) (Bohg, Morales et al. 2014) or SLAM (Riazuelo, Civera et al. 2014). A more detailed overview of current approaches and projects is given by Kehoe, Patil et al. (2015) and Wan, Tang et al. (2016).

# 3 A Cloud-based Cooperative Navigation System

*This chapter aims at describing the design and components of the overall multi-robot navigation system in which the C-LT-SLAM, as the main focus of this work, is embedded. Starting with an introduction (Section 3.1) of the field including requirement analysis (Subsection 3.1.1) as well as reviewing related work (Subsection 3.1.2), the architecture is presented in section in Section 3.2. A description of the local and global perception modules as well as the corresponding path and motion planning counterparts in Section 3.3 concludes the chapter.*

## 3.1 Introduction

As motivated in Section 1.2, local navigation solutions suffer from limited knowledge and computation resources which we want to overcome by selected networking and cloud computing techniques. The essential questions when designing these kinds of cloud-based and cooperative systems are: Which information shall be shared and to which extent? Where shall the different components and algorithms be deployed, on the local hardware or on the cloud server? Sharing maximal information over the network makes global coherent solutions possible but commonly imposes infeasible communication requirements. Applied to MR fleet navigation, the most global inherent and centralized design imposes each MR to transmit its raw sensor data over the wireless network to the cloud server which then has maximal system knowledge to infer globally optimized navigation solutions. The results are then communicated to the MRs in terms of motion commands for their actuators. In such a theoretical system, only minimum computational hardware is needed locally whereas the massive resources in the cloud could be exploited making it a highly economic and efficient navigation system. However, it also induces heavy requirements on wireless networking. Even if the bandwidth is sufficient and latencies tolerable, every disruption of the network leads to a complete downtime of the MRs which are forced to instantly stop once the connection is broken. The other extremum comes close to the status quo discussed in Section 1.1, i.e., navigation systems completely running locally with minimal knowledge sharing and low coordination among the robots reversing the previously described assets

and drawbacks. Obviously, none of these extremes are suitable designs when trying to develop a practicable navigation system that is able to handle the conditions of our target applications with nowadays available hardware and sensor technology. Instead, we have to develop a suitable design that lies somewhere in between those extremes and that satisfies the given requirements on all levels.

In order to derive a suitable architecture compromising on the contrary requirements, we start with investigating the previously presented control scheme (see Figure 2.1) as the nowadays de-facto standard system architecture successfully applied to a wide range of mobile robots in different applications. This control scheme thereby serves as a base line to derive the system requirements for cooperative navigation systems.

### 3.1.1 Requirement Analysis

The map-based control scheme of Figure 2.1 can basically be broken down into three major control loops. On the lowest level, we find the path following control loop. It tries to minimize the control error given the desired path and current localization. The needed control rate depends on the velocity and further dynamics of the MR. For typical MRs traveling with 1–3 m/s, control rates are usually set in the range of 100 to 300 Hz. Latencies of the control input, i.e., current localization estimate, lead to an increased control error. For example, a latency of 100 ms when traveling with 3 m/s possibly lets the MR overshoot a point of interest by 30 cm. Moreover, a disruption of the input signal usually forces the controller to stop the MR. Due to these properties, the path following control loop which includes the localization and path following controller seems unsuitable to be deployed somewhere else than on the local hardware.

On intermediate level, we find the local path planning control loop. It cyclically optimizes the global path on a spatially and temporally limited horizon (depending on the actual approach, commonly in the range of 5 to 30 m and 5 to 25 sec (Garcia Lopez 2018)) based on the current localization and a local costmap including the most recent sensor observations. Typical control rates are in the range of 20–50 Hz. Latencies or disruptions of the localization or costmap input can generally be handled to a higher extent than in the path following control loop. During a short disruption, the local path cannot be updated, but this circumstance does not necessarily force the MR to stop since basic collision avoidance is usually still given by the safety system. However, both input latencies and disruptions decrease the reactive behavior of the local path planner to unforeseen or dynamic obstacles. Since a global coherent solution of this control loop would profit from the knowledge and coordination possibilities of other MRs or further dynamic objects in the vicinity of the MR, a straightforward recommendation about the design of control loop cannot be derived. Instead, a differentiated solution needs to be developed with a special focus on the actual system and application properties.

The top-level global path planning control loop bases on localization information of the fleet and an up-to-date global map. Control rates are low comparable to lower control loops as well as acyclic. On this level, global solutions are clearly superior to local ones due to several reason. First, a global solution highly profits from a consistent and up-to-date global map including the latest changes of the environment. Second, especially with increasing fleet sizes, the coordination of the fleet becomes more important which is strongly facilitated when having access to global knowledge about the state of the fleet (i.e., current poses, tasks, battery load and further internal states). Finally, latencies and disruptions within this top-level control loop have a comparable low impact on subjacent control loops.

Another important aspect to consider in the design of the multi-robot system is the potential heterogeneity of the fleet. Most of current installed MR solutions consist in rather homogeneous fleets of MRs. However, current trends tend to have a few "experts" among the fleet (i.e., a few MRs with specialized sensors, actors or other hardware to increase the versatility of the fleet) or different types of MRs (e.g. autonomous fork lifts, tugger trains and mobile manipulators) aggregated and coordinated by a single navigation system. Furthermore, heterogeneity can also be introduced to enable new possibilities for the navigation system. As described in Section 1.3, safety Lidars are the base sensors of the MRs. The sensors' comparable low measurement quality also limits the accuracy of the overall navigation system. In some applications, this accuracy could be insufficient at least in some particular areas, e.g., for precisely docking at a transition point. In a networked system, this can easily be resolved by placing a high precision sensor, e.g., a motion tracking system, in this particular area which transmits its observations through the network to the localization system of the respective MR. Since the sensor information can now be used by all MRs when entering this area, highly scaling and efficient solutions can be realized. However, it also induces to necessity to handle heterogeneous sensor data.

### 3.1.2 Related Work

In Section 2.3, we already listed some related work of cloud-based or cooperative navigation systems. In the following, we want to take on reviewing related approaches with a more technical focus and by considering the requirement analysis of the previous section. While there exist numerous references for the overall area of cooperative robotics, we concentrate on cloud-based approaches in the application of MR navigation.

In (Arumugam, Enti et al. 2010), a cloud-computing framework for service robots is presented. This work mainly focuses on exploiting computational resources on the cloud server by parallel computing. The authors demonstrate how computation time of a SLAM algorithm can be sped up when running on the cloud server. In their application, the MR transmits raw sensor data (odometry and 2D Lidar data) to the cloud server. However, the work investigates neither multi-robots scenarios nor communication and real-time issues.

In contrast, Riazuelo, Civera et al. (2014) present a cloud framework for cooperative visual SLAM where several cameras build a common map of the environment. In their approach, the real-time critical part of localization remains on the local hardware whereas the computational intensive part of map building is shifted onto the cloud server. By only transmitting images related to map changes, an average data rate of 1 MB/s per robot is achieved. Mohanarajah, Usenko et al. (2015) enhances this work using the cloud framework "Rapyuta". By applying compression techniques and by only transmitting key frame images, they are able to half the needed bandwidth.

Even closer to our target application, Cardarelli, Sabattini et al. (2015) present a cloud-based approach for MR navigation in modern factories. The authors motivate their work with the issue of restricted views and knowledge of a single MR about the global environment to perform effective path planning in a dynamic environment on a predefined roadmap graph. To overcome this, a cooperative obstacle tracking approach is presented where the MR fleet and further stationary sensors in the environment employ local tracking modules to extract and track objects from the local sensor data. The tracked object information is then streamed to the cloud server which processes this information using a hierarchical fusion approach. The resulting "Global Live View" is then provided to the MRs for making proper decisions once the desired route is blocked. The work is extended in (Cardarelli, Digani et al. 2017) by integrating a control and mission manager which coordinates the MRs on the roadmap graph.

## 3.2 System Architecture

Based on the requirement analysis and reviewing existing work, we now want to derive a suitable system architecture of the overall cloud-based cooperative navigation system. When looking at the analyzed control loops in Subsection 3.1.1, the design of the top-level and low-level control loop becomes more or less straightforward while the decision on the intermediate control loop is rather complex.

Starting with low-level control loop, i.e., the path following control loop, the high control rates, low tolerance to latencies and disruptions as well as low dependency on global knowledge, highly motivates its deployment on the local hardware of the MR. In contrast, the top-level global path planning loop highly benefits from running as a central instance on the cloud server.

For the intermediate level, extended knowledge is indispensable to face the challenges of our target application. However, a remote computation of its components imposes strong requirements on the wireless communication system. More pivotal, it would shift most important autonomy functionalities to the cloud server making the MR incapable of acting in case of a network disruption. These contrary requirements impede a simple local or global

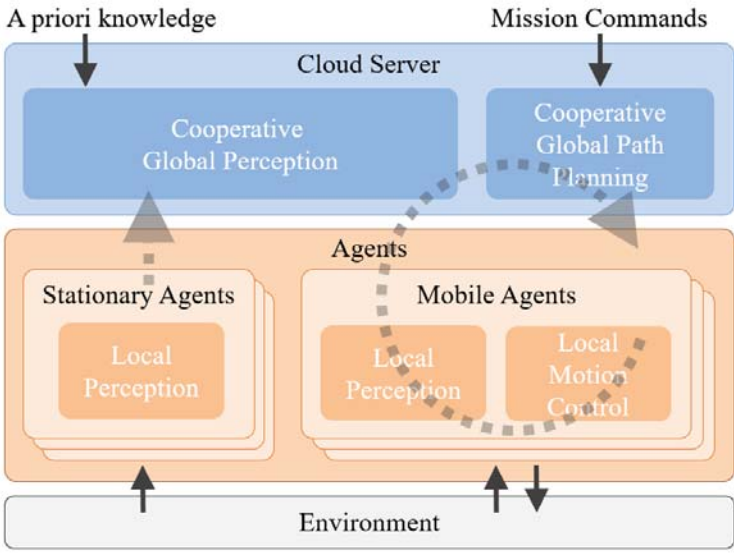



Figure 3.1: Basic concept of the cooperative navigation system with mobile and stationary agents being connected to a central cloud server.

solution but induces a more differentiated one. If we have a closer look, the problems and challenges of our target application (as described in Section 1.1) mostly appear on a larger scale. For instance, the problem of insufficient sensor information to keep the map of a changing environment up-to-date only becomes crucial in environments which strongly exceed the sensors' field of view of the MR's on-board sensors while it mostly disappears in the local environment around the robot. Roughly speaking, the local area around the MR can mostly be managed by the MR itself using its locally available knowledge and appropriate navigation algorithms. The lack of global knowledge only becomes significant after leaving this area and intensifies with increasing scale of the environment.

Using this insight, a suitable architecture allows the MRs to keep basic navigation components on-board in order to achieve a spatially and temporally limited autonomy to the cloud server whereas advanced navigation components are deployed on the cloud server providing extended and long-term navigation capabilities. Figure 3.1 presents a simplified illustration of this system architecture where we introduce the term of an agent. The reason is that we do not only want to connect the MRs to the cloud-server but also further stationary sensors which we increasingly find in nowadays industrial sites, e.g., at robotic cells. An agent is thereby simply defined as a combined sensing and computing unit in our network. Agents are then distinguished in mobile agents, i.e., the MRs, and stationary agents, i.e., stationary sensors in combination with embedded hardware for sensor data processing and communication.

Each agent employs a local perception module consisting of different perception components, see Subsection 3.3.1 for further details. In addition, mobile agents are equipped with a local motion control module performing local motion planning and path following

control (see Subsection 3.3.4). Similarly, we find a (cooperative) global perception module and a (cooperative) global path planning module on the cloud server. The global perception module (see Subsection 3.3.2) is responsible for collecting the information shared from the local perception modules of the agents and fusing the information into a global environment model. This global model is then provided to (a) the agents to update their local models and (b) to the (cooperative) global path planning module which exploits this information to coordinate the mobile agents in terms of computing a set of globally optimized global paths. To reduce communication latencies and disruptions as well as security[1] issues, a private cloud that runs on local servers of the industrial site is recommended for our navigation system.

The heterogeneity of the agents is tackled by choosing appropriate approaches and algorithms within the different modules that are capable of handling this kind of heterogeneity, e.g., perception components that are capable of handling different sensor inputs sourcing from different sensor types or path planning components that are capable of handling different kinematics and dynamics (kinodynamics).

A more detailed illustration of the architecture for the particular case of a mobile agent is given in Figure 3.2. The herein mentioned modules and components are made subject of discussion in the following section.

## 3.3 Global and Local Navigation Modules

This section describes the employed global and local system modules including their components and chosen approaches in order to give the reader a full overview of the functionalities and capabilities as well as the interfaces among the components.

### 3.3.1 Local Perception

The local perception module mainly consists of two components that feed the local environment model with extracted information from raw sensor data (depicted on the bottom left side of Figure 3.2). The local LT-SLAM is responsible for providing accurate and robust pose estimates as well as for incorporating detected changes into the local map. Since it is a core element of this work and will be extensively presented in Section 5.2, we only introduce the basic properties of this component at this point. In the basic setup, the local LT-SLAM fuses odometry and 2D Lidar (or 2D scan) data using an adapted version of the well-known MCL approach and is thereby able to provide high-frequency pose estimates that satisfy both the local path planning and path following control loop. In principle, the local LT-SLAM is thereby capable of working as a standalone localization module.

[1] A detailed discussion on security is out of scope of this work.

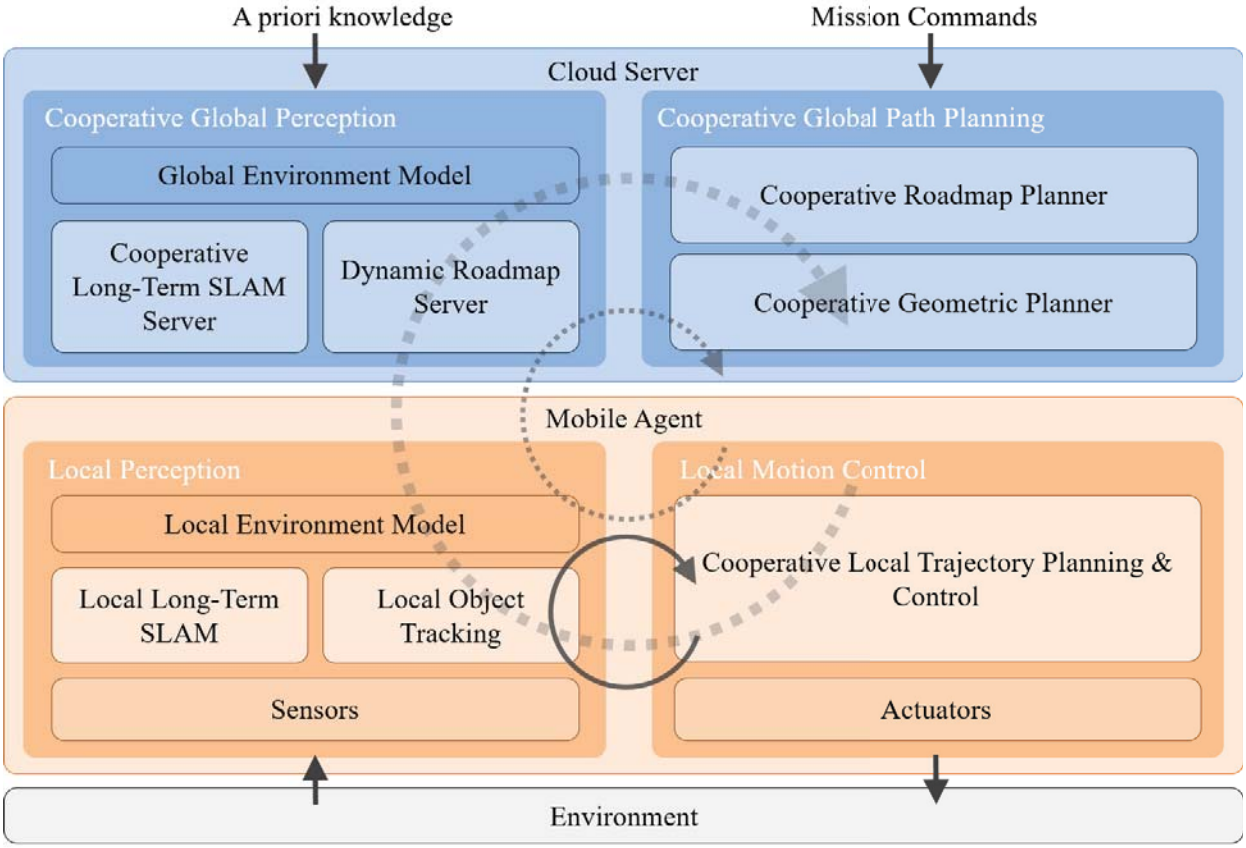



Figure 3.2: System architecture of the cooperative navigation system with local and global perception as well as path planning and motion control modules. A description of the individual modules and components is given in Section 3.3.

However, as described, the complexity of our target environment requires further input to achieve acceptable long-term performance. This is tackled by the following two properties. First, the component is designed to process and fuse heterogeneous sensor data and thereby increase the local field of view as well as compensate weaknesses of different sensor types. Second, it is exchanging map and localization information with the cooperative LT-SLAM server. More precisely, the cloud server provides map updates for the LT-SLAM's local map. Thereby, the mobile agent is assured to have an up-to-date map before entering a new area. Moreover, if the mobile agent has been detected by other (mobile or stationary) agents, this localization information is passed via the server to this agent and used to update its pose estimate.

Additionally, the local perception module employs an object detection and tracking module for perceiving the dynamic objects in the local sensor field. While some (commercial) sensor systems are able to directly provide this information (e.g. motion tracking systems providing high precision object pose information), the information usually must be inferred from raw sensor data (commonly referred to as detection and tracking of moving objects (DATMO)). While the actual design and implementation of a DATMO algorithm is out of

scope for this work, the interested reader may be referred to (Schulz, Burgard et al. 2001) (Khan, Balch et al. 2005) (Breitenstein, Reichlin et al. 2011) for examples of DATMO approaches using scan or vision data.

Both, the map data from the LT-SLAM as well as from the tracked objects are incorporated into the local environment model which is used by the local motion control to compute the costmap as the base for optimizing the local path (see Subsection 3.3.4 for more information).

### 3.3.2 Global Perception

The global perception module serves the local perception modules in terms of extended measurement and knowledge input as well as the global path planning module as a source for global environment knowledge. The already mentioned cooperative LT-SLAM server is located within this module. Its primary task is to fuse the incoming local map changes into the global map and provide agents' individual map updates. Additionally, it collects and distributes potential mutual detections from the local object tracking modules. A detailed description of this module will be given in Sections 5.3 and 5.4.

Additionally, we find the dynamic roadmap server which is responsible to compute and update a global roadmap graph of the environment based on the geometrical map provided by the cooperative LT-SLAM server (see Chapter 4 for a detailed discussion on the map representation developed in this work). It therefore directly interfaces with the cooperative LT-SLAM server to get the latest changes of the geometrical map and adjusts the roadmap within the changed areas accordingly. In brief, the chosen approach computes distance maps from Voronoi diagrams and constructs the graph based on the points with largest distance to the obstacles in the map, see (Engmann 2016) for more detailed information on this approach.

### 3.3.3 Global Path Planning

The overall job of the global path planning module is to find a set of globally optimized paths for the fleet based on the global environment model and currently pending tasks given by an overlying logistic management system. In general, this job involves several steps. First, based on the set of tasks and available mobile agents (with potentially different capabilities of transporting and sensing due to their heterogeneity), each task needs to be assigned to a specific agent, commonly referred to as task allocation. However, in order to concentrate on core navigation capabilities, we assume the task allocation to be given in a way that each mobile agent is already assigned a mission with a specific goal location. Note that a task allocation module can easily be integrated in the given navigation architecture on top of the current cooperative global path planning module. The second step now involves finding

an actual path for each agent based on its assigned task. As described, with increasing fleet sizes, local solutions usually lead to very suboptimal paths. Instead, we aim for a globally coherent solution for the coordination and global path planning problem. The chosen approach for our navigation system is presented in Abbenseth, Lopez et al. (2017) and will be briefly described in the following.

The main challenge consists in the high dimension of the search space which in general grows exponentially with increasing environment and fleet size. Therefore, a hierarchical approach is chosen with a top-level cooperative roadmap planner operating on the roadmap graph given by the dynamic roadmap server (as described in Subsection 3.3.2) and a cooperative geometric planner operating on the geometric global map provided by the cooperative LT-SLAM. The roadmap planner employs an adapted constraint-based search (CBS) which first computes isolated global paths for each mobile agent and afterwards checks these paths for potential conflicts resulting from two or more agents occupying an edge or vertice of the roadmap graph at the same (predicted) time interval. The naive approach to resolve these collisions consists in finding alternative paths on the roadmap without any conflicts. However, this could lead to suboptimal paths or in worst case to no valid solution at all. For instance, let us consider the conflict induced by two agents passing each other on a broad corridor. Forcing one of the agents to avoid this vertice at that time would lead to a significantly longer bypass route. However, on a geometrical level, the conflict could be minor since there is enough space for passing each other in this particular area, due to the point where the cooperative geometrical planner gets involved. Invoked by the roadmap planner, it computes the costs of the given conflicts on geometric level considering the agent's kinodynamic properties as well as the freespace in the area. The results are then fed back into the roadmap graph as time-dependent costs of that vertice which enables the CBS to implicitly decide on the optimal bypass maneuver by minimizing the overall costs of the updated roadmap graph. To be able to compute the costs of conflicts on geometrical level, the cooperative geometric planner runs simulations of the conflict using the cooperative local motion planning approach described in Subsection 3.3.4.

### 3.3.4 Local Motion Control

The primary task of the local motion control module is to optimize and adapt the global path given by the cooperative global planner (described in Subsection 3.3.3) based on the local environment model (see Subsection 3.3.1) and internal states on a limited spatial and temporal horizon. Furthermore, to be applicable for heterogeneous mobile agents, it has to cope with different footprints (the shape of the robot projected on the ground plane) and kinodynamics. While these are more or less standard requirements resolved by proven state-of-the-art local path planning algorithms (see, e.g., Fox, Burgard et al. (1997)), the additional requirement of incorporating predicted paths of other dynamic obstacles for

efficient navigation in busy and narrow passages as well as intersections calls for a more advanced concept and motion planning approach.

A globally optimized solution would suggest a centralized approach resolving this issue on the central cloud server with maximum global knowledge. However, as discussed, the communication and autonomy requirements contradicts this solution to be applicable in our target application. Instead, we choose a distributed approach where the participating agents involved in a local conflict resolve the conflict by themselves with minimal centralized coordination needed. This is mainly carried out by mutually exchanging planned trajectories and considering the received trajectories in the agent's predictive and cooperative trajectory planning module as presented in (Garcia Lopez, Abbenseth et al. 2017).

The cooperative trajectory planner bases on a hierarchical approach using an elastic band based corridor planner and a model predictive trajectory planner. The corridor planner computes a time-based collision-free corridor considering the received dynamic objects' trajectories while the underlying trajectory planner computes a feasible trajectory within this corridor as well as respective motion commands considering the agent's kinodynamic or additional constraints. The approach thereby unifies the local path planning and path following control in a single module.

While the cooperative trajectory planner is able to cooperatively optimize the trajectories of a group of agents involved in a local conflict, it does not rely on the exchanged information to work in general. In case of a complete fallout of the wireless network, in the by far most situations, the local motion planner would still be able to generate feasible, collision-free paths. The additional knowledge about the planned trajectories of other agents are however relevant to increase the optimality of the paths. Additionally, this enables the possibility for a central coordination input. This can be realized by a property that has not been mentioned so far. In order to prioritize the agents among each other, e.g., for giving an agent right of way, each agent is assigned a priority score. Per default, the scores are set equally. However, a central coordinator is able to overwrite the priority scores, e.g., to push a higher priority to an urgent transport task.

## 3.4 Conclusion

This chapter presented the overall cooperative navigation system crucially compromising on possibilities but also real-world limitations of cloud and networked robotics within the field of MR navigation. The main challenge consisted in balancing the advantages of global coherent solutions with limitations of wireless communication in terms of available bandwidth, delays and network disruptions. We approached this problem by analyzing the properties and requirements of the three major control loops of map-based navigation architectures. While the properties of both, the top and low-level control loop, clearly indicated a purely

global and a purely local solution respectively, the medium-level control loop demanded for a more elaborated solution. We resolved the conflicting requirements on this level with the key insight that the lack of global information on this control level mainly appears on extended spatial and temporal scale. Following this insight, we defined the core principle of our navigation architecture in terms of enabling a spatially restricted and temporal autonomy of the robots (also referred to as mobile agents) by locally running respective navigation modules whereas the long-term autonomy is realized through cooperation and interaction with the cloud server.

Following this principle, we were able to present the detailed architecture of our cooperative navigation system including a brief description of all relevant components on both the global (cloud) level as well as on the local (agent) level as well as their interfaces. On local level, we defined the local perception and motion control modules realizing a local navigation system which provides the autonomy on temporally and spatially limited horizon. On the global level, i.e., the cloud server, we found the global perception module providing globally coherent environment information as well as the global path planning module.

With a global and local component, we already presented the core concept of the C-LT-SLAM in terms of a hierarchical approach to the cooperative localization and mapping problem which will be subject to a detailed investigation in the following two chapters. We will start in Chapter 4 with presenting a suitable map representation that will serve for map-based localization on the agents, mapping, exchange and fusion of detected changes of the environment as well as input to path planning. Furthermore, based on this map representation, we will discuss the detailed approach for the C-LT-SLAM in terms of the local LT-SLAM and derive the concepts and algorithms for cooperative map updating and mutual localization.

# 4 Environment Modeling for Cooperative Long-Term Map Estimation

*As motivated in Section 1.3, building and updating maps to be used by both localization and path planning is a central objective of this work. In this chapter, based on the work's objectives as well as the presented system architecture, we derive a suitable map representation that holds all requirements for our target application. Similar to the previous chapter, an introduction of the topic is given by analyzing respective requirements of the map representation (Subsection 4.1.1) and discussing recent approaches (Subsection 4.1.2). Sections 4.2–4.5 describe how their limitations with respect to our target application can be overcome by a sound combination and extension of these approaches.*

## 4.1 Introduction

As discussed in Section 1.2, the environment map is a key element of the navigation system. It is of crucial importance to infer the current localization of the robot with recent sensor observations as well as to compute save and efficient paths and motion commands through the environment to the desired target location. Moreover, both tasks heavily rely on the up-to-dateness of the map which imposes the requirement to continuously update a given map model in dynamic and changing environments. This chapter deals with deriving a suitable map representation together with concepts and algorithms for long-term estimating the map parameters. The main challenge thereby consists in considering the different and partly contrary requirements discussed in the following section.

### 4.1.1 Requirement Analysis

From the system architecture depicted in Figure 3.2, we already got a first idea on the application and its requirements of the map on different layers in the respective modules. We now want to collate the different requirements for being able to rate related work and

derive a suitable map representation for the cooperative navigation system. When looking at our target application, the following requirements can be identified:

- **Computational demands:** The complexity of the map representation strongly impacts computational, memory as well as communication demands, especially in large-scale environments. To limit local hardware requirements for local components working with the map as well as to meet communication requirements for (wirelessly) communicating map information between server and agents, the complexity needs to be chosen carefully.
- **Geometric accuracy:** An accurate representation of the environment is crucial in order to meet the requirements of the respective components using the map. For instance, the precision of the localization component or the optimality of the computed paths strongly depend on the map accuracy. Accuracy is herein mostly considered in a metrical sense, i.e., the accuracy of the geometric description of the environment. However, in general, it also applies to non-geometric properties, see below.
- **Modeling dynamics:** Handling dynamic and changing environments is one key requirement of the overall navigation system. Therefore, it is fundamental to be able to model the dynamics in the map representation. Moreover, different gradiations of dynamics (static, semi-static and dynamic) must be representable by the map.
- **Ease of fusion:** Fusing maps or map information appears at different levels. Primary, the cooperative LT-SLAM server continuously fuses map changes detected by the agents into the global map. Additionally, to benefit from this up-to-data map information, the agents fuse global map updates from the server into their local map representations. Therefore, the ability and complexity of the fusion process is highly relevant to design a suitable map representation.
- **Non-geometric properties:** While geometrically modeling the environment is certainly most important for both localization and path planning, in some cases, pure geometric information can be too sparse for the respective component to robustly work. For instance, areas with sparse or symmetrical geometric information like corridors can be critical for the localization system to keep its demanded precision of the pose estimate. Processing non-geometric properties (e.g. the reflectivity of the objects) of the environment can reduce sensor aliasing effects and remedy the problem. Therefore, a suitable map representation needs to be capable of incorporating such additional properties.

While meeting all listed requirements already imposes a great challenge, their complementarity further complicates finding an appropriate map representation. As an example, a high map accuracy, in general, drastically increases the map complexity. The same holds for modeling the dynamics and incorporating non-geometric properties. Therefore, the decision on the map representation needs to base on a carefully chosen trade-off meeting all requirements to an appropriate level.

### 4.1.2 Related Work

In Subsection 2.2.2, we already discussed basic concepts of map representations in terms of topological, landmark and grid maps along with their advantages and disadvantages. In this section, we want to discuss more recent approaches focusing on dynamic environments and the requirements defined in the last section.

Modeling the dynamics of the environments has been the scope of numerous work. For example, the authors of (Biber and Duckett 2009) introduce a sample-based model to represent a dynamic environment on multiple time-scales. In Churchill and Newman (2012), an experience-based approach for visual long-term navigation in changing environments is presented. Instead of keeping a global map, dynamic parts of the environment are captured by a set of experiences. A non-Markov approach with a varying graphical network (VGN) is described in Biswas and Veloso (2014). During so-called episodes, all observations are used to classify and to reason about static, semi-static or dynamic objects. While these approaches yield promising results, they all lack a unified representation of the map to make it applicable to both localization and path planning and further impedes the fusion and merging of different maps.

In (Meyer-Delius 2011)(Meyer-Delius, Beinhofer et al. 2012), dynamic OGMs (DOGMs) are introduced to overcome the problems of OGM in changing environments due to the insufficient handling of map dynamics when using the occupancy grid mapping algorithm as presented in Subsection 2.2.4. Based on its static environment assumption, a cell that has been observed as occupied for a whole day will in general need to be observed as free for another whole day until its occupancy state switches back to free. Obviously, this is problematic for our target application where semi-static objects can be removed instantly after having been in the same place for a longer time-period. Instead of a static state, in DOGM, the occupancy state can have different dynamics modeled in the state transition probabilities of a HMM. The state transition probabilities $p(q_t \mid q_{t-1})$ model the probability of $q$ changing its state from the previous to current time-step. Using matrix notation, $q$ can be computed using the following recursive update rule:

$$Q_t = \eta \; B_z \; A_t \; Q_{t-1} \tag{4.1}$$

where $\eta$ is a normalizing factor. Additionally, (4.1) contains the occupancy probability vector $Q_t$:

$$Q_t = \begin{bmatrix} p(q_t = o) \\ p(q_t = f) \end{bmatrix} = \begin{bmatrix} p(q_t = o) \\ 1 - p(q_t = o) \end{bmatrix}, \tag{4.2}$$

the occupancy observation matrix $B_z$:

$$B_z = \begin{bmatrix} p(\tilde{z}_t \mid q_t = o) & 0 \\ 0 & p(\tilde{z}_t \mid q_t = f) \end{bmatrix} \tag{4.3}$$

and the state transition matrix $A_t$:

$$\begin{aligned} A_t &= \begin{bmatrix} p(q_t = o \mid q_{t-1} = o) & p(q_t = o \mid q_{t-1} = f) \\ p(q_t = f \mid q_{t-1} = o) & p(q_t = f \mid q_{t-1} = f) \end{bmatrix} \\ &= \begin{bmatrix} 1 - p(q_t = o \mid q_{t-1} = f) & p(q_t = o \mid q_{t-1} = f) \\ p(q_t = f \mid q_{t-1} = o) & 1 - p(q_t = f \mid q_{t-1} = o) \end{bmatrix}. \end{aligned} \tag{4.4}$$

The parameters of $B_z$, i.e., the two observation probabilities in Equation (4.3), are selected depending on $z_t$. More detailed, the selection bases on the artificial cell observation state $\tilde{z}_t = \{hit, miss, no\}$ which is determined for each cell by ray tracing of the actual scan measurement $z_t$. For each laser beam, the ray tracing process delivers information if a cell was hit by the laser beam (i.e., the end point of the laser beam falls within the boundaries of the cell), missed by the laser beam (i.e., traversed by the laser beam) or if no observation took place (i.e., the cell was neither hit nor missed). Based on the cell observation state, $B_z$ is then filled with the corresponding observation probabilities which are defined as static and manually chosen based on sensor properties. In contrast, the transition parameters of $A_t$ are estimated online by applying a specialized EM algorithm as described in (Mongillo and Deneve 2008).

While normal OGMs already stand out in terms of their flexibility to be applicable to different environments (and not depending on a certain structure of the environment), DOGM expand this property by additionally handling of dynamic and changing environments. However, like for OGMs, the map complexity and map accuracy in DOGM strongly depend on the grid resolution, i.e., the sizes of the grid cells. While with high resolutions, this discretized environment representation produces highly accurate maps, the accuracy decreases with lower resolutions. In contrast, the influence on the computational complexity behaves reversely.

The normal distribution transform (NDT)-OGM is introduced in (Saarinen, Andreasson et al. 2013a) and further extended in (Saarinen, Stoyanov et al. 2013)(Saarinen, Andreasson et al. 2013c). It bases on the NDT initially presented by Biber and Strasser (2003) for scan matching.

The basic idea is to approximate the scan points given in Cartesian coordinates, $\zeta = (x, y)$, from a scan observation $z_t = \{\zeta\}$ into a set of normal distributions:

$$z_t \sim \{\mu_i, \Sigma_i\}_{i=1}^{N} \tag{4.5}$$

where $\mu_i$ and $\Sigma_i$ are computed by laying out a local grid over the sensor observation field, collecting the scan points $\{\zeta\}_i$ that fall into each cell $c_i$ and then use Equation (2.18) and (2.19) to compute the normal distribution parameters. NDT-OGM takes on this idea by having a global grid map with each cell holding an NDT.

To account for the dynamics of the environment, in (Saarinen, Andreasson et al. 2013c), an adaptive RSC update is presented that is able to increase the weights of recent observations with respect to older ones and thereby increase the adaption rate of the map to changes of the environments. Furthermore, an occupancy observation model is derived that is able to better cope with low-resolution (LR) grids by considering the NDTs within the cell. The authors showed that NDT-OGMs outperform normal OGMs when used with MCL in terms of accuracy, robustness to noisy sensor inputs and performance of the localization system (Saarinen, Andreasson et al. 2013b).

NDT-OGM can be seen as hybrid grid and landmark maps combining the advantages of both map types. Since NDTs yield good approximations for various object shapes, the disadvantage of common landmark maps is mostly eliminated. Furthermore, due to the geometrical description of the object within the cell, the dependency of high grid resolutions to yield good map accuracies is reduced. Thereby LR grid maps can be applied which largely increase the computational efficiency. Yet, the quality of the approximation decreases in general with lower resolutions and non-linear object contours (see Figure 4.1.c). Moreover, in contrast to the DOGM presented in (Meyer-Delius, Beinhofer et al. 2012), the dynamic of the grid cells is not inferred from observations, but set statically and equally for the whole environment by a user-defined parameter.

A further extension to NDT-OGM is presented in (Einhorn and Gross 2015). The novelty mainly lies in applying QT to the NDT-OGMs (see Figure 4.1.d for a visualization). Thereby, the previously mentioned inaccuracies with lower resolutions and non-linear object contours can further be decreased without increasing or even with further decreasing the number of cells of the map (depending on the actual environment). This comes at cost of an increased map management effort and also complicates map fusion. Moreover, by using a EWMA as the RSC update method (see Algorithm 2.1), this approach also lacks the ability to estimate cell individual dynamics from observations.

The so far discussed approaches, only considered the geometric properties of the environment. As stated in Subsection 4.1.1, adding non-geometric properties can be of high value in environments with sparse or symmetric geometric structures. With respect to

Lidar sensors, the reflectivity of an object is of special interest. It is measured by common Lidar sensors through evaluating the intensity of the reflected laser beam. In (Beinschob and Reinke 2015) and (Leofante, Le Moal et al. 2015), an additional landmark map in terms of reflective point features is added to the OGM. Using these artificial landmarks in terms of retro-reflective markers, they show how localization and SLAM can be improved with respect to pure geometrical maps. Modeling reflectivity as point features however restrict the usage to reflective objects of punctiform shape. Moreover, storing geometrical and reflectivity information in separate maps prevents profiting from the advantages of unified approaches in terms of profiting from shared information knowledge. For instance, a retro-reflective marker may also induce a structure in the geometrical map. Finally, the presented approaches do not consider changing environments.

A unified map model for geometrical and reflectivity information is presented in (Khan, Member et al. 2016) which proposes to add a reflectivity state to the existing occupancy state of each cell in an OGM. Additionally, an incremental update routine is presented. The work also considers the pose dependency of the reflectivity measurement, i.e., how the distance and angle of incidence highly influence the reflectivity measurement of an object. While this approach overcomes the problems of the non-unified representation and profits from the general advantages of grid map approaches, it also suffers from their disadvantages. Furthermore, it also does not model the cell dynamics.

From the introduction into map representations (Subsection 2.2.2) up to the recent discussion of related work, a key insight is that all presented map types and approaches exhibit certain advantages but also disadvantages with respect to the requirements of Subsection 4.1.1. Hence, finding or developing a single map representation that fulfills all these requirements seems hardly achievable. To overcome this problem, we aim to combine different types and approaches on several map layers while enabling information flow among the different layers in order to spread newly gathered or updated information from one layer to the other.

As already indicated in Section 3.2 when describing the navigation system architecture and its modules, on a meta level, we follow a classical dual-layered map approach combining a low-level metric layer with a high-level roadmap/object layer. On the metric layer, we find detailed geometrical and reflective information about the environment. This layer is mostly filled with information by the LT-SLAM and LT-SLAM server respectively and mainly fulfills the requirements of accuracy needed by localization and geometric path planning. On the roadmap/object layer, we find more abstracted information in terms of a roadmap graph and detected dynamic objects. The abstracted information mainly serves the global path planner to work on a reduced search space and get information about dynamic objects on the graph. As already described in Subsection 3.3.2, the roadmap graph is generated and updated from the dynamic roadmap server using the information of

Table 4.1: Rating of relevant map representation approaches with respect to the requirements defined in Subsection 4.1.1.

| **Approach** | **Complexity** | **Accuracy** | **Dynamics** | **Fusion** | **Reflectivity** |
|---|---|---|---|---|---|
| HR DOGM | −− | ++ | ++ | ++ | −− |
| LR NDT-OGM | + | + | o | + | −− |
| QT NDT-OGM | + | ++ | o | − | −− |
| HR ORGM | −− | ++ | − | + | ++ |

Note: Scores are defined as satisfying the specific requirements not at all (−−), hardly (−), partly (o), mostly (+) and completely (++).

the metrical layer as input. Since the combination of a topological map and a metric map is a rather common approach, we will now focus on the design of the metrical layer. Table 4.1 shows a rating of promising approaches from Subsection 4.1.2. Here we find high-resolution (HR)-DOGM as presented in (Meyer-Delius, Beinhofer et al. 2012), LR NDT-OGM as presented in (Saarinen, Andreasson et al. 2013c), QT NDT-OGM as presented in (Einhorn and Gross 2015) and finally HR occupancy reflectivity grid map (ORGM) as presented in (Khan, Member et al. 2016). We directly see that none of these approaches score well in all fields, but rather each exhibits a major lack (in terms of satisfying a requirement hardly or not at all). If we have a more detailed look, we find that HR DOGMs satisfy the major part of the requirements in terms of modeling dynamics, accuracy and ease of fusion. However, due to its high resolution, the complexity in large-scale environments prohibits a usage for our target application. Decreasing the resolution, on the one hand, lowers this complexity, but on the other hand this will significantly decreases the accuracy making it no valuable option. LR NDT-OGM is able to overcome the complementary accuracy-complexity dependency and come off reasonably in all categories, except the reflectivity modeling that is not considered at all. In comparison, QT NDT-OGM can further improve the accuracy and in some environments maybe also the complexity but cannot overcome the major issues of LR NDT-OGM. The only approach which considers reflectivity information is given with HR ORGM. Although the HR ORGM approach seems promising in terms of a unified map representation, the complexity issue of the underlying grid map prevents a further usage. To sum up, none of these four approaches is able to fulfill the strong requirements for our metric layer. Instead, we found a set of approaches where each stands out in different categories.

Therefore, combining different approaches and concepts therefore seems like an expedient solution. When evaluating at Table 4.1, integrating the NDT into DOGMs and thereby making DOGMs applicable to low resolutions (hereafter referred to as LR NDT-DOGM) represents a simple and effective way to resolve their complexity issue as well as further profit from the advantages of NDT-based grid maps. Moreover, additionally integrating

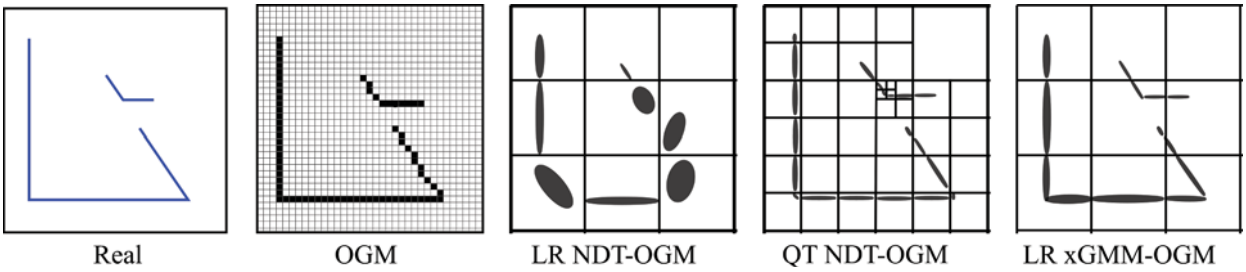



Figure 4.1: Comparison of geometric map approaches (with solely occupancy information being visualized).

Table 4.2: Rating of LR xGMM-DOGM with respect to the requirements defined in Subsection 4.1.1.

| **Approach** | **Complexity** | **Accuracy** | **Dynamics** | **Fusion** | **Reflectivity** |
|---|---|---|---|---|---|
| LR xGMM-DOGM | + | ++ | + | + | ++ |

the concept of ORGMs would also resolve the remaining missing capability of handling reflectivity information. However, the approach of Khan, Member et al. (2016) relies on HR grid maps making it incompatible to the proposed LR NDT-DOGM. Instead, we propose a new approach in terms of replacing the NDT with a more versatile mixture model that (a) is able to model the objects' reflectivity in a suitable manner and (b) improves the geometric approximation of the objects' shape compared to the NDT, especially for highly non-linear geometries. The derived mixture model extends the well-known GMM with a non-Gaussian reflectivity state and is therefore simply referred to as extended GMM (xGMM). The chosen map model of the metric layer used within our cooperative navigation system finally integrates the xGMM into LR DOGM (hereafter referred to as LR xGMM-DOGM). The superior rating of the LR xGMM-DOGM is depicted in Table 4.2. Moreover, for a better understanding of the discussed map approaches and motivation of the chosen model, Figure 4.1 illustrates their capability of geometric contour approximation on a simple environment example.

The following sections derive the proposed LR xGMM-DOGM as well as present formulas and algorithms for long-term estimation of its model parameters based on continuous sensor observations. For better comprehension as well as further motivating the final approach, we start with describing the combination of DOGMs and NDT-OGMs with a special focus on our target application in Section 4.2, followed by introducing the GMM instead of the NDT (Section 4.3) and finally, we present the expansion of the GMM to xGMM in Section 4.4.

## 4.2 Adding the Normal Distribution Transform to Dynamic Occupancy Grid Maps

As motivated in the previous section, a combination of DOGM with NDT-OGM balances out some of the respective weaknesses. While the DOGM models the occupancy state and its dynamic in detail, the NDT-OGMs focuses on adding a continuous geometric model into the cell. Combining the two approaches does not only add respective capabilities into a single map representation but also makes valuable information accessible among the two models and thereby improves the estimation process of internal states.

The combination is carried out by assigning each grid cell $c_i$ with the combined parameter set, i.e., the parameters of the HMM, $\theta_t^{HMM,i} = \langle Q_t^i, A_t^i \rangle$, and the NDT parameters, $\theta_t^{NDT,i} = \langle \mu_t^i, \Sigma_t^i, N_t^i \rangle$ resulting in the overall parameter set:

$$\Theta_t^i = \left\langle \theta_t^{HMM,i}, \theta_t^{NDT,i}, t_z^i \right\rangle = \left\langle Q_t^i, A_t^i, \mu_t^i, \Sigma_t^i, N_t^i, t_z^i \right\rangle \tag{4.6}$$

where we additionally find the time stamp $t_z^i$ of the most recent observation of the cell which gets involved when fusing cell information described in Section 5.4. In the following, due to clarity, the cell index $i$ is not mentioned explicitly unless necessary for clarification.

In principal, the HMM and the NDT could now be updated separately using the update routines as described in Subsection 4.1.2. However, this would not pay off the full potential of this combined approach. As described in Subsection 4.1.2, the NDT update generally suffers from not explicitly having knowledge about the cell's dynamic. Moreover, the occupancy observation model given in (Meyer-Delius, Beinhofer et al. 2012) relies on HR grid maps to achieve accurate results. For low resolutions, it is beneficial to consider the geometry of the object within the cell when deriving observation likelihoods, see also Saarinen, Andreasson et al. (2013c) for a further motivation of that problem. Fortunately, in our combined approach, all this needed information is available within the cell. For having an estimate of the cell's dynamic when updating the NDT parameters, we can rely on the transition probabilities of the HMM. Reversely, the NDT parameters contain valuable geometric information about the object when processing observations for updating the HMM parameters.

Based on this idea, the parameter estimation and update formulas for long-term mapping are derived in the following Sections 4.2.1–4.2.2. As motivated in Section 1.3, we will focus on noisy scan measurements from 2D Lidars. However, an extension to 3D pointclouds as generated from 3D Lidars or depth vision sensors can simply be obtained by extending the relevant formulas to the third dimension.

### 4.2.1 Long-Term Estimation of Hidden Markov Model Parameters

For long-term estimating the HMM parameters, namely the occupancy probability vector $Q_t$ and the parameters of the transition matrix $A_t$, we first consider the update formulas given in (Meyer-Delius, Beinhofer et al. 2012). With Equation (4.1), we find the update formula of $Q_t$ given a new measurement $z_t = \{\zeta_j\}_{j=1}^{N_z}$. It only requires selecting the respective observation matrix $B_t$ based on $\tilde{z}_t$. Subsequently, the transition matrix is updated with the mentioned EM approach. Although this update algorithm stands out in terms of simplicity and should be a good approximation when processing highly precise scan data, it poorly considers sensor noise and map discretization effects making it unsuitable for LR NDT-DOGMs. To overcome this issue, a more elaborated sensor observation model based on the current observation and the NDT parameters is needed. We take on the idea of having a cell observation $\tilde{z}_t = \{\tilde{\zeta}_{i,j}\}_{i,j=1}^{N_m,N_z}$ as an artificial state to compute the observation likelihoods:

$$\tilde{\zeta}_{i,j} = \begin{cases} hit & \text{if } \zeta_j \text{ falls within the boundaries of } c_i, \\ miss & \text{if beam of } \zeta_j \text{ traverses } c_i, \\ no & \text{otherwise.} \end{cases} \tag{4.7}$$

However, instead of selecting the state of $\tilde{z}_t$ for each cell directly from the actual sensor observation $z_t$, we estimate the probability of each state of $\tilde{z}_t$ depending on $z_t$. Additionally, we add the most recent information about the object within the cell. This results in the approximation of the observation likelihood with the following model:

$$p(z_t \mid q_t^i) \approx \eta \sum_{\tilde{z}_t} p(q_t^i \mid \tilde{z}_t, z_t, \mu_{t-1}^i, \Sigma_{t-1}^i)\; p(\tilde{z}_t \mid z_t). \tag{4.8}$$

By treating the scan points $\zeta_j$ of $z_t$ as conditionally independent as in (Meyer-Delius 2011), the observation probability of each $\zeta_j$ can be computed separately and the final observation probability of $p(z_t \mid q_t^i)$ can be obtained by multiplying over the single point probabilities:

$$p(z_t \mid q_t^i) \approx \eta^* \prod_{j=1}^{N_z} \sum_{\tilde{\zeta}_{i,j}} p(q_t^i \mid \tilde{\zeta}_{i,j}, \zeta_j, \mu_{t-1}^i, \Sigma_{t-1}^i)\; p(\tilde{\zeta}_{i,j} \mid \zeta_j). \tag{4.9}$$

To account for sensor noise when computing $p(\tilde{\zeta}_{i,j} \mid \zeta_j)$, we model the probability of the actual hit point as normal distributed. However, to reduce complexity, we neglect the noise of the angular measurement $\alpha$ resulting in a one-dimensional normal distribution along the beam of a scan point with mean in its range measurement $r$ and the sensor specific variance $\sigma_r$:

$$p(d_j \mid \zeta_j) \sim \mathcal{N}(r_j, \sigma_r) \tag{4.10}$$

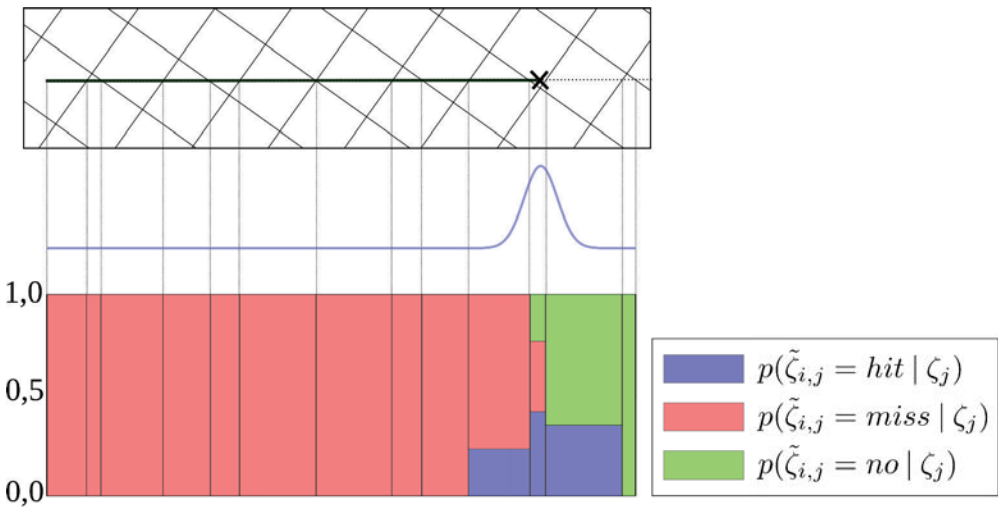



Figure 4.2: Derivation of cell observation likelihoods: Sensor beam traversing the grid from left to right (top), corresponding modeled probability distribution of the range measurement (middle), cell observation probabilities for each traversed cell where the height of the bars represents the probability value (bottom).

where $d_j$ is the distance on the beam of $\zeta_j$ with respect to the sensor origin. The approximation $\sigma_\alpha \approx 0$ can be justified by the property of common Lidar sensors whose noise on $\alpha$ is minor.

Using this property, we can now apply a ray tracing on the beam of $\zeta_j$ within the distance $d_j \in [0, r_j + 3\sigma_r]$ and compute the entrance distance $d_{i,j}^{en}$ and exit distance $d_{i,j}^{ex}$ for each traversed cell. Subsequently, for each cell on the ray, the cell observation likelihoods are computed with:

$$p(\tilde{\zeta}_{i,j} = hit \mid \zeta_j) = \Phi(d_{i,j}^{ex}) - \Phi(d_{i,j}^{en}) \tag{4.11}$$

$$p(\tilde{\zeta}_{i,j} = miss \mid \zeta_j) = 1 - \Phi(d_{i,j}^{ex}) \tag{4.12}$$

$$p(\tilde{\zeta}_{i,j} = no \mid \zeta_j) = \Phi(d_{i,j}^{en}) \tag{4.13}$$

where $\Phi(d)$ is the cumulative normal distribution function:

$$\Phi(d) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{d - r}{\sqrt{2\sigma_r}}\right)\right). \tag{4.14}$$

with $\operatorname{erf}(x)$ as the error function which is defined as:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\tau}\, d\tau. \tag{4.15}$$

The process of deriving the cell observation likelihoods is also illustrated in Figure 4.2.

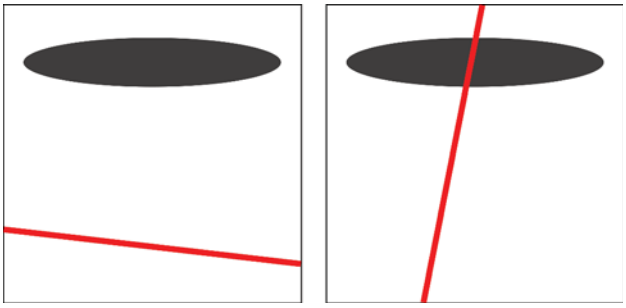


Figure 4.3: Sensor beam traversing a cell from different directions. Left: Beam traverses cell far off the object estimated by the NDT giving no further information if the object is still occupying the cell. Right: Beam clearly traverses object indicating that the object has vanished and the cell to be free.

The remaining step involves modeling $p(q_t^i \mid \tilde{\zeta}_{i,j})$ for the three possible states of $\tilde{\zeta}_{i,j}$. If $\tilde{\zeta}_{i,j} = no$, no new information about the occupancy state is available. Consequently, we model equal probabilities for the two occupancy states of $q_t$:

$$p(q_t^i \mid \tilde{\zeta}_{i,j} = no) = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix}^T. \tag{4.16}$$

For $\tilde{\zeta}_{i,j} = hit$, we also model a constant probability defined by the predefined parameter $k_{hit} \in [0, 1]$:

$$p(q_t^i \mid \tilde{\zeta}_{i,j} = hit) = \begin{pmatrix} k_{hit} & 1 - k_{hit} \end{pmatrix}^T. \tag{4.17}$$

By choosing $k_{hit} < 1$, we are able to introduce further uncertainties which have not been covered in the observation model so far.

The final and special case consists in $\tilde{\zeta}_{i,j} = miss$. In contrast to the previously defined cell observation probabilities, selecting static probabilities does not cover well the variety of possible configurations that may appear. Especially when working with low grid resolutions, a cell may still be occupied although the beam traversed the cell. Figure 4.3 illustrates this issue with a cell holding an NDT which is traversed by two laser beams from different directions. We see that the distance of the laser beam to the NDT is of crucial importance when trying to infer its occupancy probability. Consequently, we incorporate the distance of the beam to the object occupying the cell into the likelihood model by adding a direct dependency on $\zeta_j$ and the NDT parameters:

$$p(q_t^i \mid \tilde{\zeta}_{i,j} = miss, \zeta_j, \mu_{t-1}^i, \Sigma_{t-1}^i) = \begin{pmatrix} \kappa(\zeta_j, \mu_{t-1}^i, \Sigma_{t-1}^i) & 1 - \kappa(\zeta_j, \mu_{t-1}^i, \Sigma_{t-1}^i) \end{pmatrix}^T. \tag{4.18}$$

Closely following (Saarinen, Andreasson et al. 2013c), we define:

$$\kappa(\zeta_j, \mu^i_{t-1}, \Sigma^i_{t-1}) = \begin{cases} 1 - p(d^{ml}_{i,j} \mid \mu^i_{t-1}, \Sigma^i_{t-1}) & \text{if } c_i \text{ owns an NDT,} \\ k_{trav} & \text{otherwise.} \end{cases} \tag{4.19}$$

where $d^{ml}_{i,j}$ is the maximum likelihood distance on the beam of $\zeta_j$ traversing $c_i$, i.e., the distance point that maximizes $p(d_{i,j} \mid \mu^i_{t-1}, \Sigma^i_{t-1})$. An algorithm on how to efficiently compute $d^{ml}_{i,j}$ is presented in (Saarinen, Andreasson et al. 2013c). $k_{trav}$ is a user-defined parameter which is applied if the cell does not own an NDT.

### 4.2.2 Long-Term Estimation of Normal Distribution Parameters

Given $z^i_t = \{\zeta_j\}_{j=1}^{N^i_z}$, i.e., the subset of points of the observation $z_t$ that fall into cell $c_i$ (which is a byproduct of computing the observation matrix described in the previous chapter), the EWMA-RSC update as given in Algorithm 2.2 of Subsection 2.1.4 can be used to update the NDT parameters. However, this algorithm assumes the estimated state to be static. As a result, after a long-term observation of the cell, the influence of new observations on the NDT parameters is so small that it hardly adjusts to actual changes of the state. This makes it highly unsuitable to be applied in changing environments. In Einhorn and Gross (2015), the algorithm is therefore adjusted in a way that new observations have a higher impact with respect to older ones. This is carried out by reducing $k_{max}$ (see line 7 of Algorithm 2.2). While this parameter's primary usage is to avoid numerical instabilities when the sample size $N^i_t$ has grown to large numbers, it influences the weight of the current sample point and thereby its impact on the NDT parameters. Thereby, $k_{max}$ can be used to control the adaption rate of the parameters to recent observations. While this in general improves handling dynamic environments, the problem remains in finding a suitable value for $k_{max}$. A low $k_{max}$ and the resulting fast adaption rate is reasonable for highly dynamic cells but increases noise and error-proneness to outliers. In contrast, as discussed, a high $k_{max}$ yields good robustness to sensor noise but does not well adapt to changes. More complicating, in most environments, the different areas exhibit different dynamics and may change over time.

With our NDT-DOGMs, we can overcome this problem by incorporating the current estimate of the cell's dynamics (in terms of the transition matrix $A^i_t$ of the HMM) into the EWMA-RSC algorithm. More specifically, with $p(q^i_t = o \mid q^i_{t-1} = o)$, we find a direct estimate of the cell being static. We exploit this information by replacing the computation of $\gamma^i_t$ of the EWMA-RSC (see line 2 in Algorithm 2.2) as follows:

$$\gamma^i_t = p(q^i_t = o \mid q^i_{t-1} = o) \frac{N^i_t}{N^i_t + N^i_z}. \tag{4.20}$$

**Algorithm 4.1** NDT parameter update algorithm

**procedure** UPDATE_NDT($z_t^i, A_t^i, \theta_{t-1}^{NDT,i}$)
  $\mu_z^i = \frac{1}{N_z^i} \sum_{j=1}^{N_z^i} \zeta_j$
  $\Sigma_z^i = \frac{1}{N_z^i - 1} \sum_{j=1}^{N_z^i} (\zeta_j - \mu)(\zeta_j - \mu)^T$
  $\gamma_t^i = p(q_t^i = o \mid q_{t-1}^i = o) \frac{N_{t-1}^i}{N_{t-1}^i + N_z^i}$
  $\mu_t^i = \gamma_t^i \mu_{t-1}^i + (1 - \gamma_t^i)\mu_z^i$
  $\Sigma_t^i = \gamma_t^i \Sigma_{t-1}^i + \gamma_t^i (1 - \gamma_t^i)(\mu_t^i - \mu_z^i)(\mu_t^i - \mu_z^i)^T$
  **if** $N_{t-1}^i \leq k_{max}$ **then**
    $N_t^i = N_{t-1}^i + N_z^i$
  **else**
    $N_t^i = N_{t-1}^i$
  **end if**
  **return** $\theta_t^{NDT,i}$
**end procedure**

To make the effect of this adaption apparent, let us consider two extreme cases. In the first case of a static cell, i.e., $p(q_t^i = o \mid q_{t-1}^i = o) \approx 1$, $\gamma_t^i$ is not influenced remaining its suitable properties for static states. In contrast, for highly dynamic cells, $p(q_t^i = o | q_{t-1}^i = o)$ gets close to zero. This drastically reduces the weight of older observations with respect to most recent ones realizing a quick adaption to latest changes. The complete update procedure of the NDT parameters is given in Algorithm 4.1 which only slightly differs from Algorithm 2.2.

### 4.2.3 Updating Map Parameters with New Observations

With the algorithms presented in Subsection 4.2.1 and 4.2.2, we can now define the update routine for long-term mapping with NDT-DOGMs.

The update process is triggered once a new scan observation $z_t$ is available. A scan contains a set of $N_z$ scan points $\{\zeta_j\}_{j=1}^{N_z}$ given in either polar or Cartesian coordinates $[\zeta_j]^{\mathcal{S}} = [r, \alpha]^{\mathcal{S}} \rightarrow [x, y]^{\mathcal{S}}$ of the sensor frame $\mathcal{S}$ . With the information of $\mathcal{S}$ relative to the robot frame $\mathcal{R}$ and current robot pose $x_t$, the scan points are first transformed into the map frame $\mathcal{M}$, $[\zeta_j]^{\mathcal{M}} = T_{\mathcal{S}}^{\mathcal{M}} \ [\zeta_j]^{fS}$, where $T_{\mathcal{S}}^{\mathcal{M}}$ is the transformation matrix from $\mathcal{S}$ to $\mathcal{M}$. Since from this point on, all relevant variables are given in $\mathcal{M}$, the frame superscripts are omitted in favor of clarity.

Algorithm 4.2 depicts the subsequent update routine. The first step involves computing the cell observation likelihoods $B_t^i$ and cell hit points $z_t^i$ as described in Subsection 4.2.1. The results are then used to update the HMM parameters $\theta_t^{HMM,i}$ updated based on Equation (4.1). Additionally, the transition matrix is recomputed by executing the EM algorithm from (Mongillo and Deneve 2008). Afterwards, the update routine concludes with an update of the NDT parameters $\theta_t^{NDT,i}$ as given in Algorithm 4.1.

**Algorithm 4.2** NDT-DOGM update algorithm

1: **procedure** UPDATE_NDT_DOGM($z_t, m_{t-1}$)
2: $\{B_t^i, z_t^i\} = compute\_observation\_matrix(z_t, m_{t-1})$
3: **for all** $c_i$ **do**
4: $\theta_t^{HMM,i} = update\_HMM(B_t^i, \theta_{t-1}^{HMM,i})$
5: $\theta_t^{NDT,i} = update\_NDT(z_t^i, A_t^i, \theta_{t-1}^{NDT,i})$
6: **end for**
7: **return** $m_t$
8: **end procedure**

## 4.3 Gaussian Mixture Models for Non-Linear Contour Approximation

While the presented LR NDT-DOGM exhibit sound properties in terms of handling dynamics and a suitable map complexity, the geometric accuracy modeled with the NDT depends on the actual environment structure and chosen grid resolution, see also in Figure 4.1. In this example, the left wall can be approximated well with the ellipses of the NDT within the cell whereas we find a poor approximation of the right part of the environment. Although Saarinen, Andreasson et al. (2013b) demonstrates that the NDT-OGM can drastically lower the dependency of the localization accuracy on the grid solution with respect to normal OGMs, NDT-DOGM cannot prevent an increasing error with decreasing resolutions. While for localization tasks, this may be a minor issue, the path planner working with this kind of maps may face difficulties finding optimal paths. For instance, let us consider the upper left part of the environment in Figure 4.1. Due to the adverse location of the objects in the middle/middle-right cell of the NDT-OGM, the resulting ellipses block the transit area impeding the path planner to use this transition as a potential path.

The basic problem in this example consists in a single NDT trying to approximate several parts of the surrounding walls resulting in the illustrated bulky ellipses that contain a mixture of sample points of all these parts. Although not given in Figure 4.1, the same problem arises when dealing with non-linear contours like corners or circular objects. In general, the approximation quality of the NDT depends on the level of non-linearity of the object contour. As a matter of fact, the approximation error increases with growing cell sizes due to the fact that a step-wise linear approximation of a non-linear object generally gets worse with increasing step sizes. For many of these scenarios, having few additional NDTs drastically increases the approximation quality. For instance, a corner geometry can be approximated accurately using two NDT components. Figure 4.4 illustrates the issue of approximating different common object shapes with a varying number of NDT components.

Using multiple NDT components for representing the probability distribution of a random variable is commonly known as a GMM which has been introduced in Subsection 2.1.4).

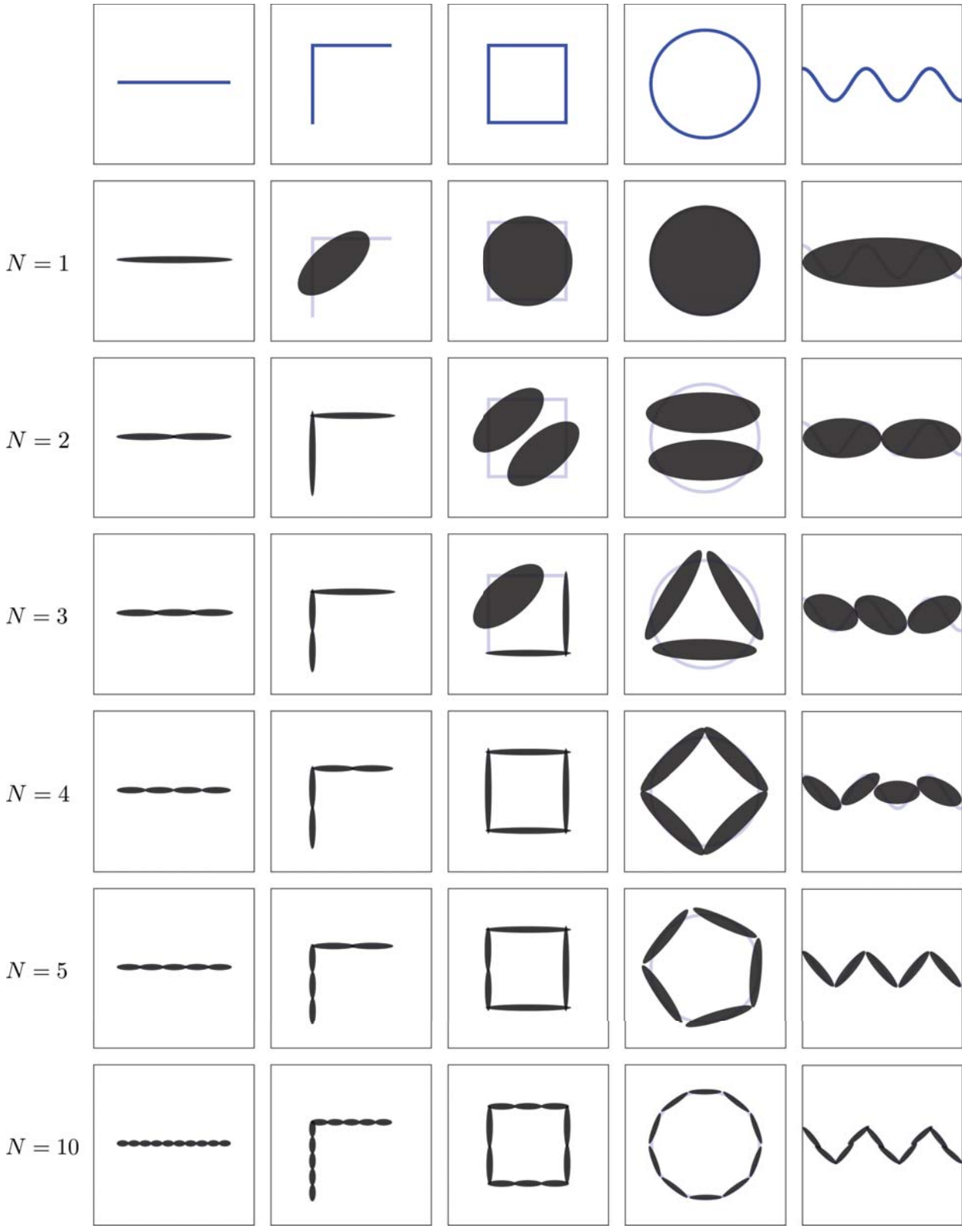



Figure 4.4: Approximation of common geometric contours with different numbers of normal distribution components (represented as grey ellipses).

While GMMs are widely used for applications where a single normal distribution is not sufficient, to the author's knowledge, they have not been applied to this particular task although exhibiting suitable properties in terms of computational efficiency and available algorithms for estimating the GMM from sample points. Improving the geometrical accuracy is only one problem that may be addressed with GMMs. Additionally, it opens up more flexibility on generally dividing the state space within the cells based on further (non-Gaussian) states, e.g., its color, to realize enriched map representations. We will use this property to derive the xGMM for reflectivity mapping later on in Section 4.4.

In the first place, replacing the NDT with GMM is carried out by simply replacing in $\Theta_i$ the NDT parameters with the GMM parameters $\theta^{GMM,i} = \left\langle \{w_\psi^i, \mu_\psi^i, \Sigma_\psi^i\}_{\psi=1}^{N_{gmm}^i} \right\rangle$. Note that in the following sections, the cell index $i$ is only explicitly mentioned where needed for comprehension to improve the clarity of the formulas. The influence of GMM on the parameter estimation process is subject to the following Sections 4.3.1 and 4.3.2.

### 4.3.1 Long-Term Estimation of Hidden Markov Model Parameters

The update routine of the HMM is only slightly affected by the extension to GMM. More specifically, an adaption is only needed for computing the cell observation probability $\tilde{\zeta}_{i,j} = miss$ since this is the only point where the geometric approximation of the object is used in terms of finding the distance of the sensor beam to the object. When using the NDT, this was carried out by computing the maximum likelihood distance $d_{i,j}^{ml}$ of the beam to the NDT. Since in general we now find numerous components in the GMM, $d_{i,j,\psi}^{ml}$ is now computed for each component $\psi$ and the maximum value, $d_{i,j}^{ml} = \max_\psi d_{i,j,\psi}^{ml}$, inserted in Equation (4.19).

### 4.3.2 Long-Term Estimation of Gaussian Mixture Model Parameters

For incrementally updating the GMM, we already reviewed the IGMM algorithm in the fundamentals in Subsection 2.1.4. While this algorithm provides basic functionality, it exhibits three main drawbacks when applied to our target application:

1. The algorithm does not explicitly model the dynamics of the system.

2. There is no control over the component number leading to a constant increase of components.

3. It can get stuck in local minima.

The lacking control of the component number is not only problematic for computational reasons but also because it keeps potentially irrelevant or erroneous components forever. For instance, a component created by an outlier measurement will never be removed. The local minimum issue is illustrated in Figure 4.5 where two simple scenarios in terms of a cell containing a corner and a straight wall respectively are considered. In both of these scenarios, the IGMM algorithm is prone to produce suboptimal estimation results. In case of the corner scenario, a single component with a bulky ellipse (see Figure 4.5.1b) is created instead of the desired two-component GMM with elongated ellipses (as displayed in Figure 4.5.1d). In contrast, in case of the straight wall scenario, two components are created (see Figure 4.5.2b) while one would be more efficient to approximate this geometry. Note that these examples base on an adverse order of incoming observations. A different order may also lead to better or even optimal solutions. However, since we cannot accept that the order of the incoming data have major effects on the approximation result, this problem needs to be resolved in order to achieve suitable behavior of the algorithm for our application. Both, the local minimum as well as the uncontrolled increase of components, can be tackled by performing so called SM operations which control the component number in a more intelligent way, i.e., maximizing the data likelihood of the GMM with a minimum number of components. An exemplary work of an SM algorithm applied to GMMs is given in Ueda, Nakano et al. (1998).

In order to develop a suitable update algorithm for the GMM, we will first tackle the problem of handling dynamics with the concept already applied to NDT parameter estimation in Section 4.2 which we will refer to as dynamic observation weighting. Consequently, we will expand this concept to a dynamic weighting SM-IGMM specifically tailored for our target application.

### Dynamic Observation Weighting

As mentioned in Section 4.2, we exploit the fact that we have information about the dynamics in order to control the weights of recent observations with respect to elder ones. For updating the normal distribution parameters $\mu_\psi$ and $\Sigma_\psi$ of component $\psi$, we replace $\gamma_\psi^t$ given in Equation (2.33) of the IGMM algorithm with $\hat{\gamma}_\psi^t$ in a similar manner as we did in Equation (4.20) for adjusting the EWMA-RSC algorithm:

$$\hat{\gamma}_\psi^t = p(q_t^i = o \mid q_{t-1}^i = o) \frac{s_\psi^{t-1}}{s_\psi^{t-1} + \pi_\psi^t} \tag{4.21}$$

Equation (4.21) is then used to update the normal distribution parameters using Equation (2.20) and (2.21) of the EWMA-RSC algorithm. Moreover, the weight update is altered as follows:

$$w_\psi^t = \gamma_\psi^t \; w_\psi^{t-1} + (1 - \gamma_\psi^t) \; \pi_\psi^t. \tag{4.22}$$

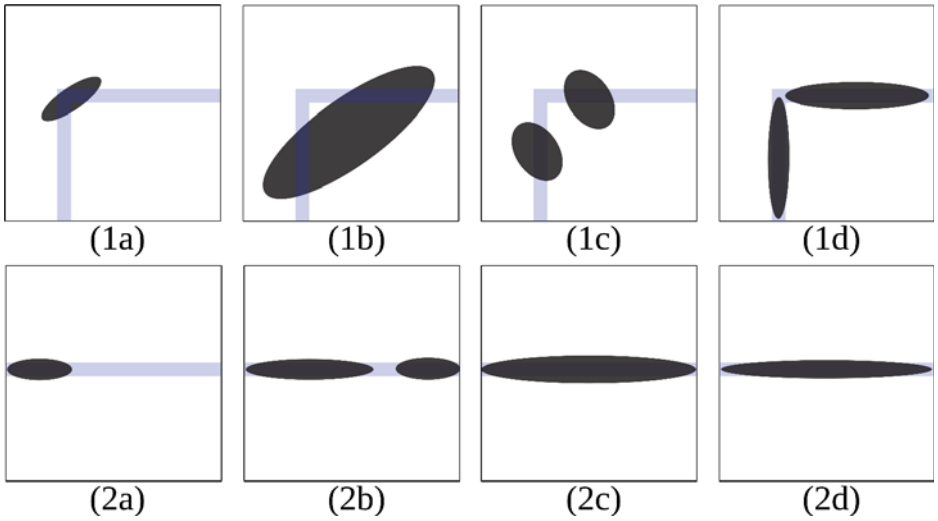



Figure 4.5: Evolvement of GMM using SM-IGMM in two selected scenarios. Incremental update of the GMM in cell with corner shaped object (top row): single component initialized with observations near corner point (1a), component after successive updates with observations from inner to outer area of corner (1b), two components created from split procedure (1c), convergence of components after successive observations (1d). Incremental update of the GMM in cell with straight line shaped object (bottom row): single component initialized with observations at left border (2a), second component created based on novelty criterion after consequent observations from right border (2b), resulting merged component right after merging (2c), convergence of component after successive observations (2d).

where $\gamma_{\psi}^{t}$ is computed according Equation (4.20) with $N_t$ as the overall observation count of the GMM.

### Controlling Component Number through Split and Merge

In Figure 4.4, we see that common object geometries can be approximated with a few normal distribution components. In most cases, further increasing the component number does not lead to a significant increase of the approximation quality but increases the computational complexity. For instances, a line segment geometry can be well approximated with a single component, a corner geometry with two components and a rectangle with four components. Therefore, the main task of the SM-IGMM consists in finding the optimal number of components for the specific object geometry in each cell. More formally, we are interested in minimizing the negative data likelihood of the GMM with a minimum number of components as given by the following cost function:

$$L(\theta^{GMM}) = -k_1 \log p(\zeta_{1:t} \mid \theta^{GMM}) + k_2 N_{gmm} \tag{4.23}$$

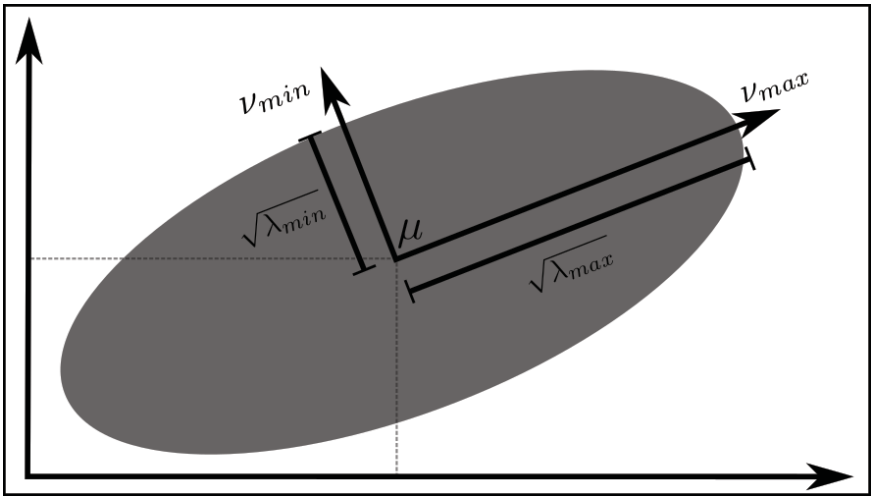



Figure 4.6: Relation of 2D normal distribution parameters to the geometry of an ellipse with mean $\mu$ as the center point of the ellipse with respect to the reference frame, minor/major eigenvectors $\nu_{min,max}$ defining the orientation of the ellipse and minor/major eigenvalues $\lambda_{min,max}$ determining the length of ellipse's minor/major semi-axes.

where $k_1$ and $k_2$ are weighting parameters of the two terms. Instead of actually finding analytical or numerical solutions of the actual minimization problem, SM-IGMM approximates the solution by performing SM operations after each update step of the GMM in terms of checking all components for splitting or merging with another component to approximate the minimization solution.

Developing a suitable SM procedure for our target application involves finding appropriate SM criteria responsible for triggering the actual split or merge process. The SM criteria needs to detect a component being stuck in a local minimum and resolve it by splitting or merging the component in a way that it is able to converge with succeeding observations to a proper approximation of the actual object's contour. Since the components represent the distribution of the object's contour, not the object as a whole, a well-formed ellipse of the component's covariance matrix has punctual or elongated shape, as can be seen from Figure 4.4. This desired geometric property of the ellipse can be decoded in the requirement of $\Sigma_\psi$ exhibiting a minor eigenvalue with a marginal value (see Figure 4.6 for an illustration of the relation between the normal distribution parameters and the geometry of the corresponding ellipse). Using this insight, the following split criteria is derived.

A component $\psi$ is split if its minor eigenvalue $\lambda_{\psi,min}$ exceeds the split threshold parameter $k_{split}$:

$$\lambda_{\psi,min} > k_{split}. \tag{4.24}$$

The split of component $\psi$ into the components $\psi_{1,2}$ is then carried out along its major axis:

$$\mu^t_{\psi\rightarrow\psi_{1,2}} = \mu^t_\psi \pm \frac{1}{2}\sqrt{\lambda_{\psi,max}}\,\nu_{\psi,max} \tag{4.25}$$

where $\nu_{\psi,max}$ is the major eigenvector. The covariance matrices and weights of the resulting components are set equally according to:

$$\Sigma^t_{\psi\rightarrow\psi_{1,2}} = \begin{bmatrix} \lambda_{\psi,min} & 0 \\ 0 & \frac{1}{4}\lambda_{\psi,max} \end{bmatrix}. \tag{4.26}$$

$$w^t_{\psi\rightarrow\psi_{1,2}} = \frac{1}{2} w^t_{\psi}. \tag{4.27}$$

In contrast, merging of two components $\psi_1$ and $\psi_2$ is executed if the minor eigenvalue $\lambda_{\psi_{1,2},min}$ of their merged component does not significantly increase with respect to the smaller minor eigenvalue of $\psi_1$ and $\psi_2$:

$$\lambda_{\psi_{1,2},min} < \Big((1 + k_{merge})\ min\{\lambda_{1,min}; \lambda_{2,min}\}\Big) \tag{4.28}$$

where $k_{merge}$ defines the threshold, reasonably set slightly above zero. In order to evaluate (4.27), we need to compute the parameters $\theta_{\psi_{1,2}}$ of the merged component according to (Zhang, Chen et al. 2003):

$$w^t_{\psi_{1,2}} = w^t_{\psi_1} + w^t_{\psi_2} \tag{4.29}$$

$$\mu^t_{\psi_{1,2}} = \frac{w^t_{\psi_1}\mu^t_{\psi_1} + w^t_{\psi_2}\mu^t_{\psi_2}}{w^t_{\psi_{1,2}}} \tag{4.30}$$

$$\Sigma^t_{\psi_{1,2}} = \frac{1}{w^t_{\psi_{1,2}}}\left(w^t_{\psi_1}\left(\Sigma^t_{\psi_1} + \mu^t_{\psi_1}(\mu^t_{\psi_1})^T\right) + w^t_{\psi_2}\left(\Sigma^t_{\psi_2} + \mu^t_{\psi_2}(\mu^t_{\psi_2})^T\right)\right) - \mu^t_{\psi_{1,2}}(\mu^t_{\psi_{1,2}})^T \tag{4.31}$$

However, checking all component pairs for potential merging by computing the merged component parameters is computationally expensive, especially for increasing component numbers $N_{gmm}$. Therefore, we apply a prefilter step in oder to select promising candidate pairs. This is carried out by a less computational intensive line segment distance check. Thereby, we define the line segment of a component by its start/end point $p^{[start,end]}_{\psi}$:

$$p^{[start,end]}_{\psi} = \mu^t_{\psi} \pm 3\sqrt{\lambda_{\psi,max}}\ \nu_{\psi,max}. \tag{4.32}$$

Subsequently, the distance between the line segments for each pair is computed and only those pairs are considered for potential merging whose distance is below a threshold.

Figure 4.5 illustrates how the described SM procedure resolves the previously described local minimum problem of a corner and straight wall respectively. The GMM of the corner cell that is stuck in a single component GMM (as shown in Figure 4.5.1b) is resolved by a split procedure of the component (see Figure 4.5.1c/1d for the results) while the straight

**Algorithm 4.3** SM algorithm for GMM estimation

**procedure** SPLIT_AND_MERGE($\bar{\theta}_t^{GMM}$)
  **if** $N_{gmm} < k_{N_{max}}$ **then** ▷ Overall limitation of $N_{gmm}$
    **for all** components $\psi$ **do**
      **if** $\lambda_{\psi,min} > k_{split}$ **then** ▷ Splitting criteria
        $(\theta_{\psi_1}, \theta_{\psi_1}) = compute\_split\_components(\theta_\psi)$
        $remove\_component(\psi)$
        $add\_component(\psi_1)$
        $add\_component(\psi_2)$
        $N_{gmm} = N_{gmm} + 1$
      **end if**
    **end for**
  **end if**
  **for all** component pairs $\psi_{1,2}$ **do**
    **if** $line\_segment\_distance(\psi_{1,2}) < k_{max\_dist}$ **then** ▷ Prefiltering candidates
      $\theta_{\psi_{1,2}} = compute\_merged\_component(\theta_{\psi_1}, \theta_{\psi_1})$
      **if** $\lambda_{\psi_{1,2},min} < \Big((1 + k_{merge})\ min\{\lambda_{i,min}; \lambda_{j,min}\}\Big)$ **then**
        $remove\_component(\psi_1)$
        $remove\_component(\psi_2)$
        $add\_component(\psi_{1,2})$
        $N_{gmm} = N_{gmm} - 1$
      **else**
        *continue*
      **end if**
    **end if**
  **end for**
  **return** $\theta_t^{GMM}$
**end procedure**

wall cell is resolved by merging the two components of its GMM (in Figure 4.5.2b) to a single component (see Figure 4.5.2c/2d for the results).

Pseudocode of the SM procedure executed after each update step of the IGMM is given in Algorithm 4.3.

## 4.4 Incorporating Object Reflectivity Information

As described in Subsection 4.1.1, integrating additional properties to the so far purely geometric map of the metric layer can be of significant value, e.g., for achieving sufficient precision and robustness of the localization system in areas of the environment where sensor aliasing is critical due to sparse geometrical map information or symmetric structures. In general, this could be any type of additional measurement that the sensor outputs. For Lidar sensors, the intensity of the reflected laser beam is often measured by commercially

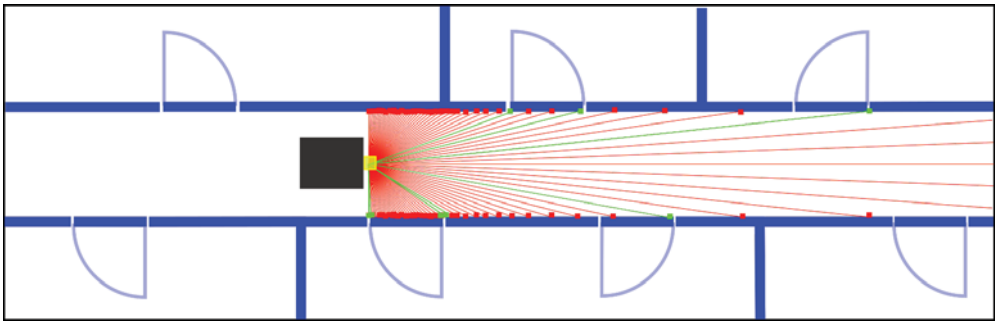


Figure 4.7: Example of MR (black rectangle) traversing corridor with high reflective door frames and resulting scan observation with low reflective (red) and high reflective measurement points (green).

available sensors indicating the reflectivity of the measured object. Depending on the actual sensor, this information can be binary, i.e., distinguishing high from low reflective objects, or quasi-continuous, i.e., a fine-discretized set of possible values $\in [0, \rho_{max}]$ where $\rho_{max}$ is the maximum measurable reflectivity. This reflectivity measurement helps distinguishing different objects or parts of a single object that have a similar geometrical contour but different reflectivity due to the material of its surface. Thereby, sensor aliasing is reduced which eases localization in these areas. As an example, let us consider the straight corridor of Figure 4.7. Solely using geometrical information of the scan data, i.e., only its range data, conclusions on the position of the robot in the longitudinal direction of the corridor cannot be inferred. Localization algorithms as described in Subsection 2.2.3 will therefore experience an increasing localization error in this dimension resulting from the accumulated error of the odometry. If we now assume that the corridor exhibits different surface materials, e.g., wallpapered walls and metallic door frames like in the example of Figure 4.7, and the localization or SLAM approach is able to process the reflectivity measurements, the localization error may be drastically reduced. While it is not unusual for industrial environments to exhibit objects which (significantly) differ in their reflectivity, we cannot always rely on having a sufficient number of them to compensate in geometrically sparse areas. However, by enabling the localization/SLAM approach to process this kind of information, we open up the possibility to additionally use retro-reflective markers as a fallback strategy to stabilize areas which exhibit neither sufficient geometrical structure nor reflective objects. At first sight, this violates the objective of deploying additional infrastructure. However, in this particular use case, the costs and efforts involved are marginal due to the following reasons. First of all, investment and installation costs of retro-reflective markers are negligible compared to other solutions. Moreover, with an algorithm which is able to process the reflectivity measurements and to integrate them into the long-term mapping process, the effort of initial mapping and remapping (in the case of major environmental changes) can be omitted.

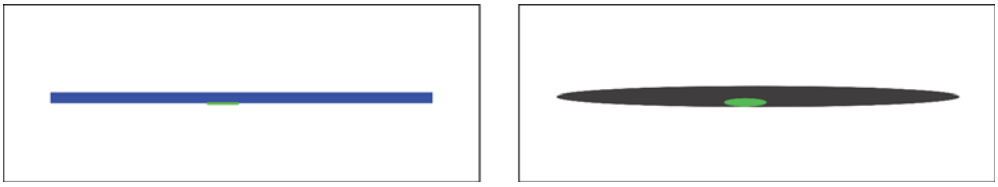


Figure 4.8: Cell containing wall (blue) and a marker of retro-reflective tape (green) (left), ellipses of two component xGMM where green color indicates a high reflective probability of the component (right).

With the previously presented GMM of Section 4.3, we already find a good base for adding further non-geometrical properties to the map representation and map estimation process. The PDF of a GMM (see Equation (2.23)) can thereby extended by the PDF of the non-geometric state variable with model parameters $\theta^*$:

$$p(x \mid \theta^{xGMM}) = \sum_{\psi=1}^{N_{xgmm}} w_\psi \; p(x \mid \theta_\psi^{NDT}) \; p(x \mid \theta_\psi^*). \tag{4.33}$$

Note that in Equation (4.33), we assume the two PDFs to be conditionally independent. We will further discuss this assumption when deriving the reflectivity model. Adding a non-geometrical state would also have been possible in the previously described NDT-DOGM. However, since there is only a single NDT component per cell, we can only assign the property to the whole object within the cell neglecting the fact that the property's value may be unequal throughout the cell's area, e.g., an object with a significant color gradient. Especially with increased cell sizes, this problem becomes highly relevant. If we look at the particular application of reflectivity, we often find objects that exhibit different surface materials with potentially different reflectivity properties.

A simple example is shown in Figure 4.8 where a cell contains a part of a wall with a thin retro-reflective marker on it. Measurements resulting from the non-marker area will appear low reflectivity while the marker area will induce high reflective measurements. However, with a single NDT component, we cannot distinguish between these two areas resulting in a blurred reflectivity estimation that will most likely indicate a low reflectivity of the whole region due to the comparable small number of high reflectivity measurements from the marker area. With the GMM from the previous section, we already find the possibility to distinguish several areas of the cell in terms of different components representing the geometrical contour of the respective area. Using the xGMM, this can be enhanced to also distinguish different areas based on non-geometrical properties. For instance, in the example of Figure 4.8, an additional component is created to represent the high reflective area although it contains no further geometric information.

### 4.4.1 Binary Reflectivity Modeling

For adding reflectivity to the GMM, we model the reflectivity state as a binary state, $\rho = \{low, high\}$, indicating either low or high reflectivity. This choice is supported by the following reasons:

1. It represents the measurement output of many of currently available (safety) Lidar sensors.

2. It increases the potentially supported sensors since quasi-continuous sensor outputs can easily be integrated by transforming the quasi-continuous state into the binary state by simple thresholding. In contrast, integrating a sensor outputting binary information into a quasi-continuous reflectivity state gets more complex.

3. It allows for more efficient probability estimation and mapping algorithms as well as it avoids calibration issues compared to more high-dimensional state spaces (see also Khan, Member et al. (2016) for a motivation of that topic).

The reflectivity measurement $\zeta^\rho$ is then incorporated into the extended observation vector $\hat{\zeta}_t = \begin{bmatrix}\zeta & \zeta^\rho\end{bmatrix}^T$. Additionally, the reflectivity probability vector $R_\psi$ is defined as:

$$R_\psi = \begin{bmatrix} p(\rho = low) \\ p(\rho = high) \end{bmatrix} = \begin{bmatrix} p(\rho = low) \\ 1 - p(\rho = low) \end{bmatrix}. \tag{4.34}$$

We now define the likelihood of observing $\hat{\zeta}_t$ using Equation (4.33):

$$p(\hat{\zeta}_t \mid \theta^{xGMM}) = \sum_{\psi=1}^{N_{xgmm}} w_\psi \; p(\zeta \mid \theta_\psi^{NDT}) \; p(\zeta^\rho \mid \theta_\psi^\rho). \tag{4.35}$$

Herein, $p(\zeta \mid \theta_\psi^{NDT})$ remains unchanged while $p(\zeta^\rho \mid \theta_\psi^\rho)$ is modeled as follows:

$$p(\zeta^\rho \mid \theta_\psi^\rho) = Z^T B_z^\rho \; R_\psi^t \tag{4.36}$$

where $Z = \begin{bmatrix}\zeta^\rho = low & \zeta^\rho = high\end{bmatrix}^T$ is the reflectivity measurement vector (appearing as one of the two unit vectors depending on the measured reflectivity state) and $B_z^\rho$ is the reflectivity observation matrix:

$$\begin{aligned} B_z^\rho &= \begin{bmatrix} p(\zeta^\rho = low|\rho = low) & p(\zeta^\rho = low|\rho = high) \\ p(\zeta^\rho = high|\rho = low) & p(\zeta^\rho = high|\rho = high) \end{bmatrix} \\ &= \begin{bmatrix} p(\zeta^\rho = low|\rho = low) & 1 - p(\zeta^\rho = high|\rho = high) \\ 1 - p(\zeta^\rho = low|\rho = low) & p(\zeta^\rho = high|\rho = high) \end{bmatrix}. \end{aligned} \tag{4.37}$$

By modeling the likelihood of $\zeta^\rho$ as conditionally independent from $\theta_\psi^{NDT}$, we neglect geometrical influences on the reflectivity observation. As investigated in (Khan, Member et al. 2016), these influences are not minor. More specifically, the distance and angle of incidence of the laser beam carrying the reflectivity information clearly affect the resulting reflectivity measurement. Nevertheless, neglecting these influences can be justified by the following two reasons. First, while in (Khan, Member et al. 2016) the reflectivity measurement is modeled with a quasi-continuous state space, the effects on binary reflectivity measurements are usually much lower. For example, the intensity measurement of a straight-shaped, high reflective object commonly decreases with increasing incidence angle of the laser beam but only falls below the threshold to be classified as high reflective for extreme angles near 90 degrees. Same consideration holds for increasing distances to the measured object. Second, it enables the usage of a comparable simple and thereby computational efficient measurement model which partly accounts for these influences by choosing appropriate observation probabilities. More specifically, the discussed uncertainty of observing a high reflective object as high reflective is considered by setting a clearly non-zero probability of $p(\zeta^\rho = low|\rho = high)$. In contrast, since a low reflective object is very unlikely to produce a high reflective measurement, $p(\zeta^\rho = high|\rho = low)$ can be set (close) to zero. Apart from that, the selected probability values should also include the measurement properties of the used sensor, e.g., the accuracy of the range and reflectivity measurements.

### 4.4.2 Recursive Estimation of the Reflectivity State

With the described binary reflectivity model, we now want to derive a recursive estimator that is able to process continuously incoming reflectivity measurements and estimate the reflectivity state of each component of the mixture model.

For a recursive estimation of $\rho$, we follow the binary Bayes filter described in Subsection 2.1.2 using the log odds $l(r)$:

$$l(\rho) = \log \frac{p(\rho = low)}{1 - p(\rho = low)} \tag{4.38}$$

with the recursive update rule:

$$l_t(\rho) = \log \frac{p(\rho = low \mid \zeta_t^\rho)}{1 - p(\rho = low \mid \zeta_t^\rho)} + l_{t-1}(\rho) - l_0(\rho) \tag{4.39}$$

where $l_0(\rho)$ is based on the prior over $p(\rho = low)$:

$$l_0(\rho) = \log \frac{p_0(\rho = low)}{1 - p_0(\rho = low)}. \tag{4.40}$$

For being able to recursively solve Equation (4.39), we need to define the inverse observation model $p(\rho = low \mid \zeta_t^\rho)$. Applying Bayes theorem, we get:

$$p(\rho = low \mid \zeta_t^\rho) = p(\zeta_t^\rho \mid \rho = low)\frac{p_0(\rho = low)}{p(\zeta_t^\rho)}. \tag{4.41}$$

Rewriting Equation (4.41) in odds form results in:

$$\frac{p(\rho = low \mid \zeta_t^\rho)}{1 - p(\rho = low \mid \zeta_t^\rho)} = \frac{p(\zeta_t^\rho \mid \rho = low)}{1 - p(\zeta_t^\rho \mid \rho = low)} \frac{p_0(\rho = low)}{1 - p_0(\rho = low)} \tag{4.42}$$

and the following log odds form:

$$\log \frac{p(\rho = low \mid \zeta_t^\rho)}{1 - p(\rho = low \mid \zeta_t^\rho)} = \log \frac{p(\zeta_t^\rho \mid \rho = low)}{1 - p(\zeta_t^\rho \mid \rho = low)} + l_0(\rho). \tag{4.43}$$

Inserting Equation (4.43) in the recursive update rule of Equation (4.39), we finally get:

$$l_t(\rho) = \log \frac{p(\zeta_t^\rho \mid \rho = low)}{1 - p(\zeta_t^\rho \mid \rho = low)} + l_{t-1}(\rho). \tag{4.44}$$

So far, we neglected the dynamics of the cell as well as that $\hat{\zeta}_t$ is only assigned to component $\psi$ with probability $\pi_\psi^t$. In the case of $\pi_\psi^t < 1$, $\hat{\zeta}_t$ needs to have a reduced influence on the current reflectivity belief of component $\psi$ since there is a non-zero probability that the measurement was induced by another component of the mixture model. Similarly, with increasing dynamics of the cell, recent observations need to get a higher weight with respect to elder ones depending on the current dynamic estimate $p(q_t^i = o \mid q_{t-1}^i = o)$ of cell $i$. To account for both, we alter Equation (4.44) to realize a dynamic weighting of the respective terms:

$$l_t(\rho) = \pi_\psi^t \log \frac{p(\zeta_t^\rho \mid \rho = low)}{1 - p(\zeta_t^\rho \mid \rho = low)} + p(q_t^i = o \mid q_{t-1}^i = o)\, l_{t-1}(\rho). \tag{4.45}$$

To get a feeling for the effect of the weighting terms in Equation (4.45), let us first assume that $p(q_t^i = o \,|\, q_{t-1}^i = o) = 1$ so that there is no weighting effect on $l_{t-1}(\rho)$. In that case, the cell is estimated to be static resulting in $\rho$ being a static state. If $p(q_t^i = o \mid q_{t-1}^i = o) < 1$, the cell is assumed to be non-static. Hence, $p(q_t^i = o \,|\, q_{t-1}^i = o)$ discounts the prior log odds $l(p(r_\psi^{t-1}))$ in favor of the current measurement to realize a faster adaption. In the extreme case of $p(q_t^i = o \mid q_{t-1}^i = o) = 0$, the cell is assumed to be highly dynamic and $l_{t-1}(\rho)$ fully discarding the influence of previous measurements on the current log odds. In a similar manner, $\pi_\psi$ controls the influence of the current measurement on the log odds.

We can now expand the dynamic weighting SM-IGMM described in Section 4.3 with the reflectivity estimation described in this section. For checking new observations $\hat{\zeta}_t$ on novelty, we need to expand the novelty criterion of Equation (2.32):

$$p(\hat{\zeta}_t \mid \theta^{xGMM}) < \frac{k_{nov}}{(2\pi)^{\frac{D}{2}} det(\Sigma_\psi)^{\frac{1}{2}}} + k_{nov,\rho}\, H_\psi \quad \forall\, \psi. \tag{4.46}$$

where $k_{nov,\rho}$ is a predefined parameter and $H_\psi$ the Shannon entropy (Shannon 1948) of $p(\rho = low)$ defined as:

$$H_\psi = -p(\rho = low) \log_2 p(\rho = low) - (1 - p(\rho = low)) \log_2 (1 - p(\rho = low)) \tag{4.47}$$

which is used to incorporate the uncertainty of $R_\psi$. If novelty is given, a new component is created with an initial value of the reflectivity's log odds:

$$l_0 = \log \frac{p(\zeta_t^\rho \mid \rho = low)}{1 - p(\zeta_t^\rho \mid \rho = low)} + l_0(\rho) \tag{4.48}$$

Otherwise, the components are updated. With respect to the IGMM algorithm, the posterior computation from Equation (2.26) is altered by using the likelihood given in Equation (4.35). The successive update of the normal distribution parameters remains unmodified while for updating the reflectivity belief, Equation (4.45) is used.

The merging procedure of the SM also needs to be adjusted in a way that the merging criterion considers the reflectivity state of merging candidates to prevent merging two components holding significantly different reflectivity beliefs. Hence, we add the following merging criterion which evaluates the similarity of the reflectivity beliefs and compares this to a predefined upper threshold $k_{merge,\rho}$:

$$\| p(\rho_{\psi_1} = low)\Big(1 - p(\rho_{\psi_2} = low)\Big) - p(\rho_{\psi_2} = low)\Big(1 - p(\rho_{\psi_1} = low)\Big) \| < k_{merge,\rho}. \tag{4.49}$$

Note that the left side of Equation (4.49) converges to zero the higher the similarity, i.e., the closer the reflectivity states of $\psi_1$ and $\psi_2$. Additionally, note that merging of the two components is only carried out if both Equations (4.28) and (4.49) hold true.

The splitting procedure as given in Algorithm 4.3 remains mostly identical. The only part that needs to be added is the definition of the initial reflectivity belief of the resulting components $\psi_{1,2}$ which we set equally for both components using the belief of the split component $\psi$, i.e., $R_{\psi_{1,2}} = R_\psi$.

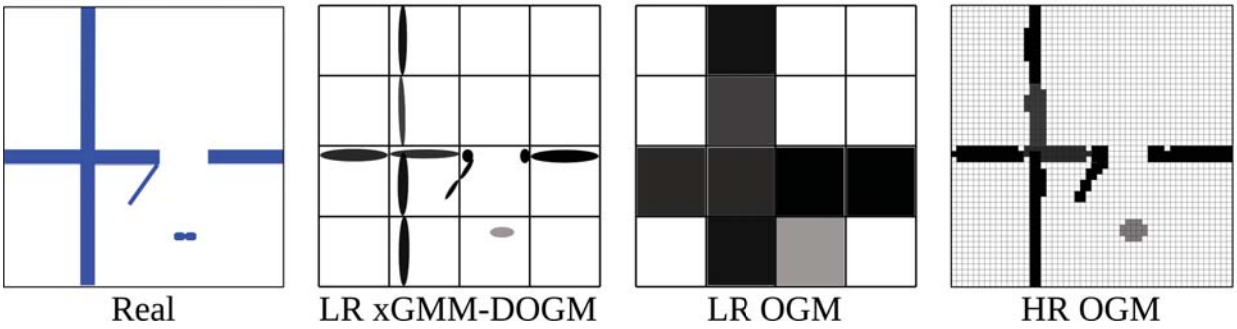



Figure 4.9: Conversion of a LR xGMM-DOGM into a LR/HR OGM. The darker the cell, the higher the probability that it is occupied.

## 4.5 Transformation to High-Resolution Occupancy Grid Maps

As motivated in Section 1.2, one key objective of this work is to have a unified map representation that is usable by both, localization and (geometric) path planning. The main requirement of (geometric) path planning with regard to the map representation is to have an accurate description of free and occupied space in an appropriate format to be processed by the planning algorithm. The xGMM-DOGM provides this information in terms of the DOGM describing the occupancy probability of each cell and the xGMM providing an accurate approximation of the object occupying the cell. Moreover, it contains an estimate about the dynamics of the environment which is also of value for path planning, e.g., to avoid busy areas in which the MR has a higher risk to get stuck. Making the planning algorithm compatible to xGMM-DOGMs and thereby taking advantage of its full potential is therefore the recommended option. However, in order to increase its versatility and compatibility to state-of-the-art path planning implementations, we additionally want to provide a more commonly used map interface in terms of a standard OGM.

Since both map representations base on grid maps, a naive approach of converting a xGMM-DOGM into a OGM applies the same grid resolution and simply assigns each cell in the OGM the occupancy value from the respective cell in the xGMM-DOGM. However, since the xGMM-DOGM is intended to be used with low grid resolutions, the resulting OGM, in general, poorly approximates occupied and free spaces which impedes the path planner to find optimal paths. Figure 4.9 illustrates this issue. Given an LR xGMM-DOGM (Figure 4.9.b) which results from mapping the environment illustrated in Figure 4.9.a, the LR OGM depicted in Figure 4.9.c is obtained by applying the described naive transformation approach.

Instead, a more suitable approach consists in incorporating the geometric information of the object hold by the xGMM and converting the xGMM-DOGM into a HR OGM as illustrated in Figure 4.9.d. In this approach, the conversion is carried out by rendering the

ellipses of the xGMM of each LR cell on an HR grid where each HR cell that is touched by an ellipse is assigned the occupancy value of its overlying LR cell. Note that the reflectivity information hold by the xGMM-DOGM is discarded in this transformation process since it is not of value for common path planning approaches.

## 4.6 Conclusion

In this chapter, we presented the map representation as well as respective update algorithms for long-term map estimation as a base for the C-LT-SLAM. The main challenge consisted in compromising the different and partially contrary requirements in terms of moderate computational demands, high geometric accuracy, handling environmental dynamics, ease of map merging and fusion as well as the ability to incorporate further non-geometric states valuable for sparsely or symmetrically structured environments. By rating recent approaches, we found that none of them could fully fulfill these requirements needed by our target application. However, a combination of some of the approaches seemed promising for compensating the individual drawbacks. More specifically, we found suitable properties by combining DOGM, i.e., a grid-based focusing on modeling the environment dynamics, with the NDT, i.e., a geometric approach approximating the object's contour within each cell. To be able to have a better geometric approximation of strongly non-linear contours as well as to properly integrate the object's reflectivity, we extended the geometric approach by replacing the NDT with a mixture model basing on and extending the well-known GMM. Specially tailored for our target application, we were then able to design an iterative update algorithm in terms of the SM-IGMM to update the model parameters online based on new sensor observations in long-term operation.

In the following chapter, we will see how the xGMM is used by the local LT-SLAM for localization and mapping as well as on the global level to realize cooperative map updating and providing up-to-date global maps.

# 5 Multi-Robot Cooperative Long-Term SLAM and Mutual Localization

*Based on the derived map representation of Chapter 4, in this chapter, the C-LT-SLAM as a key module of the overall navigation system described in Chapter 3 is presented. Similar to the previous two chapters, Section 5.1 starts with an introduction investigating relevant requirements (Subsection 5.1.1) and related work (Subsection 5.1.2). Subsequently, our approach is presented with a local LT-SLAM (Section 5.2) as the core component where cooperative capabilities in terms of mutual localization (Section 5.3) and cooperative map updating (Section 5.4) are build upon.*

## 5.1 Introduction

While previous chapters already mentioned the major concept and related requirements of the C-LT-SLAM in the context of the overall navigation system, we now take a deeper step on the approach level to determine the requirements as the base for developing a suitable approach.

### 5.1.1 Requirement Analysis

In Section 3.3, we already described the structure of the C-LT-SLAM in terms of the local LT-SLAM whose primary task consists in providing localization information for local motion control and the LT-SLAM server enabling cooperative capabilities in terms of mutual localization and cooperative map updating.

For the local LT-SLAM, the major requirement is to provide an online pose estimation with needed accuracy and robustness in dynamic environments. Due to varying accuracy requirements for different applications as well as the potential heterogeneity of the mobile fleet, this needs to be achieved with different sensor configurations. Being able to process data from multiple sensors of different type and varying sensor data quality is therefore of

crucial importance. Regarding the dynamics of the environment, this does not only involve incorporating the developed map representation of Chapter 4 but further deal with uncertainties resulting from the dynamics in order to achieve a long-term sustaining localization system. Furthermore, computational efficiency of the local LT-SLAM is another important requirement since hardware resources on the agents are limited. To enable cooperation, the LT-SLAM needs to be capable of interfacing with the LT-SLAM server in terms of detecting, sharing and fusing map changes as well as incorporating mutual detections into the pose estimation process. The ability of initial map building, i.e., the ability to create a map of the environment without prior knowledge, can be seen as an optional requirement. Since initial map building, especially for large-scale environments, in general poses different demands to the SLAM approach and only needs to be done once, having two separated and more focused approaches for the two tasks (i.e., initial map building and LT-SLAM) is preferable over a unified but less-performant approach. Same holds for global localization.

In Section 3.3, the concept for cooperative map updating, i.e., sharing and fusion of map information, has been presented in terms of a hierarchical approach with the central LT-SLAM server connected to the local LT-SLAM components of the agents as a trade-off between global coherence of the solution, local autonomy of the agents and communicational demands. Although this concept drastically reduces communication bandwith requirements in contrast to a fully centralized approach, an efficient representation of the transmitted map changes is further desirable in order to maximize the number of participating agents supported by the C-LT-SLAM. This includes the transmission of local map updates from the agents to the server as well as global map updates from the server to the agents. Additionally, on the server side, the capability of continuously fusing inputs from a potentially large number of agents is required. Finally, a concept to avoid fusing erroneous data into the local as well as the global map needs to be incorporated.

Mutual or cooperative localization describes approaches that use relative measurements among the robots to improve their localization. The relative measurement is usually generated by a detection module on the robots that is able to detect other team members with on-board sensors. A special issue of interest for these approaches is data association, i.e., if the detection module provides information of which robot in the fleet was detected or if this association is unkown. Data association can get complex depending on the homogeneity of the fleet, its size and the distribution of the robots in the environment. Therefore, most approaches concentrate on the basic capability of mutual localization by assuming a detection module outputting measurements with known associations. Since our target application also involves massive and highly homogeneous fleets and we do not want to limit on detection modules that can clearly identify the robot which it is currently detecting, the desired approach is able to handle unknown data association. Another important requirement deals with the modeling of measurement uncertainty. While this uncertainty is

often neglected or modeled with a simple normal distribution, the desired approach needs to handle more complex uncertainty models to increase its versatility in terms of usable detection modules and sensor types. Finally, the detection measurements including their uncertainty measures and further information need to be communicated among the robots arising the necessity of an efficient data representation.

### 5.1.2 Related work

Related work for the C-LT-SLAM divides in the fields of (single robot) localization and LT-SLAM for dynamic and changing environments as well as cooperative localization and SLAM.

#### Single Robot Long-Term SLAM

As discussed in Section 2.2, LT-SLAM can be seen as an intermediate problem between localization and SLAM. Consequently, existing LT-SLAM approaches commonly expand an approach in one of the two areas to provide the long-term capability. The properties of a LT-SLAM approach basing on a SLAM algorithm are naturally closer to that of SLAM approaches, i.e., improved abilities to initial map building, exploration of new areas and loop closure but also include its drawbacks of higher computational demands as well as less-frequent and less-accurate localization updates. More localization-related LT-SLAM approaches behave conversely.

We already reviewed the work from Biber and Duckett (2009), Churchill and Newman (2012) and Biswas and Veloso (2014) (in the related work of map representations, see Subsection 4.1.2) as examples of experience or episode-based representations of the dynamic environment. These works also provide respective LT-SLAM algorithms which embed their map representations. However, for the same reason given in Subsection 4.1.2, i.e., the non-unified representation, those approaches are not further investigated.

Particle filter-based approaches for 2D robot localization and SLAM can be seen as the currently predominant approaches mainly due to their capabilities to incorporate arbitrary (non-linear) motion and sensor observation models as well as their computationally efficiency when applied to this particular application. We already presented MCL as the particle filter-based solution for localization as well as FastSLAM for SLAM. According to the discussion of SLAM-related and localization-related LT-SLAM approaches, Valencia, Saarinen et al. (2014) present a LT-SLAM based on an extension of MCL while the approach Tipaldi, Meyer-Delius et al. (2013) builds opon FastSLAM.

The latter combines the already described DOGMs with a RBPF similar to the described FastSLAM of Subsection 2.2.5. However, in contrast to FastSLAM, each particle does not hold a complete map estimate but only a set of cells that have been observed by the particle

in combination with a so called lifelong map of the whole environment which is shared among the particles. However, after having visited the entire environment, the particle's set of observed cells may also have grown to the size of the environment. To reduce the resulting computational complexity, a forgetting mechanism is developed that reduces the number of cells which each particle holds in its map estimate. Given the parameters of the transition matrix of the HMM, a stationary distribution and mixing time can be computed. The stationary distribution represents the occupancy state that the cell converges to if it is not further observed. The mixing time is the time needed for this convergence. Once a cell is stationary, it is removed from the particle's cells set in case it has not significantly diverged from the cell in the lifelong map.

In contrast, Valencia, Saarinen et al. (2014) present a LT-SLAM using an extended MCL algorithm and the already described NDT-OGMs as the map representation. The work is an extension of (Saarinen, Andreasson et al. 2013b) which examines MCL on a static NDT-OGM. In this work, the authors demonstrate that the maximum a posteriori (MAP) pose estimate of the MCL remains very accurate when exposed to dynamic objects. Since mapping with a known pose is a comparable simple task, Valencia, Saarinen et al. (2014) exploits this insight to extend the approach for changing environments by using the MAP pose to map the changes into a short-term map assuming that the environment only changes slightly during each visiting interval. The short-term map is added as a separate layer to the static map which the authors refer to as a dual-timescale map. An incoming sensor observation is then processed as follows. First, the particles are updated using MCL. For computing the importance weights, the MCL algorithm is expanded in a way that sensor observation likelihoods are first evaluated on the static map. If this likelihood is below a predefined threshold, the process is repeated on the short-term map. Second, the map is updated. To avoid mapping erroneous data, mapping of the current observation is only executed if the localization uncertainty is low. As an uncertainty measure the trace of the covariance resulting from the posterior pose distribution of the particles is used. If the uncertainty is below a predefined threshold, the mapping process proceeds by updating the short-term NDT-OGM based on the MAP pose using a RSC update for the normal distribution parameters and the occupancy update model as presented in (Saarinen, Andreasson et al. 2013c).

#### Cooperative Methods for Localization and SLAM

Cooperative localization and SLAM approaches can commonly be divided in centralized and decentralized methods. Centralized methods consider the respective problem in a common and therefore high-dimensional state space usually achieving more coherent and accurate solutions at the cost of massive computational demands and high communication requirements. In contrast, decentralized methods can only approximate the global coherent

solution but are practically more relevant due to their computational and communicational efficiency.

**Mutual Localization** In (Fox, Burgard et al. 2000), a decentralized approach for mutual localization is presented. Due to the high dimensionality when including all robot poses in a common state space, an approximated representation by factorization of the individual pose beliefs is applied so that each robot is able to run a local pose estimator of its own pose belief. This estimator is chosen as a MCL basing on odometry and exteroceptive sensor observations to provide basic localization functionality. Additionally, each robot is equipped with a detection module using on-board vision and Lidar sensors to detect and measure other robots. Once a robot makes a detection of another robot, the detection measurement is used to update the localization beliefs of both robots. The approach assumes known associations of the detected robot. For processing the measurement information within the MCL, a detection model for updating the importance weights of the particles is derived. In its general form, this detection model needs to multiply the localization beliefs of the two involved robots. In order to avoid multiplying two sample-based densities, the particle distribution is converted using a QT. Furthermore, the actual observation model is presented which consists in a mixture model incorporating both, the likelihood of true positive (TP) and false positive (FP) detections as well as the uncertainty of the actual relative pose measurement variables in from of a unimodal normal distribution. A similar work is found in (Prorok, Bahr et al. 2012) where a clustering algorithm is proposed to approximate the particle distribution with a set of clusters to decrease the communicational and computational burden.

Roumeliotis and Bekey (2002) describe a KF-based approach for mutual localization. Assuming a mutual detection module outputting relative pose measurements with known association and Gaussian noise, a centralized as well as a decentralized approach is derived. The decentralized approach resolves the lack of global knowledge by letting the robots exchange relevant localization beliefs and Jacobi matrices of the measurement model. Thereby, same as in the previous work, associations of the detections are assumed to be known.

Similar, in (Nerurkar, Roumeliotis et al. 2009), a distributed mutal localization approach is presented using a MAP-based estimator to estimate the full trajectory of each robot. To avoid a centralized approach due to mentioned efficiency reasons, a distributed data allocation scheme is presented enabling each robot to reconstruct the global optimization problem. The resulting non-linear least-squares minimization problems is then solved by a Levenberg-Marquart algorithm.

In terms of centralized approaches, (Howard, Matark et al. 2002) present an approach using maximum likelihood estimation and numerical optimization techniques to infer the

relativ poses of the robot team. Martinelli, Pont et al. (2005) examine a KF-based, centralized approach which also considers the case of observing only relative range or bearing of two robots.

**Cooperative SLAM** An exemplary work for centralized cooperative SLAM is given in (Howard 2006) which extends the single-robot FastSLAM for cooperative exploring and mapping of large-scale unknown environments. The main problem tackled in this work consists in the unknown relative initial poses of the robots needed for mapping into the same global map. Two methods are presented to overcome the problem using a mutual detection module which is assumed to provide data association and minor uncertainty of the detection measurement.

In Kleiner, Sun et al. (2011), a cooperative navigation framework for teams of mobile robots in large industrial domains is described. Although the paper focuses on the description of an adaptive road map planner, a cooperative map updating approach is briefly presented. Each robot localizes itself with a local instance of MCL on a static OGM. Inconsistencies between current observations and the OGMs are communicated to a central server. The server fuses these reported inconsistencies into a global DOGM and provides updates for the robots' OGMs. Further information of the format of the map inconsistencies and the fusion process is not given.

The following three works have already been mentioned in previous related work sections but are now revived with a focus on the current topic.

Arumugam, Enti et al. (2010) demonstrates the speed-up of a FastSLAM implementation running on a cloud-server, mainly as a result of parallelizing the particle processing steps. In their approach, raw sensor data of odometry and 2D scans are communicated to the cloud-server where the whole SLAM algorithm is carried out. However, the approach presents no cooperation among several robots. Moreover, the critical issue of network load and latencies is not investigated.

In (Riazuelo, Civera et al. 2014), a cloud-based, cooperative visual SLAM is presented. In this distributed approach, each robot is equipped with a camera. Comparable to the approach in (Kleiner, Sun et al. 2011), the localization on a known map remains on the robots while the cloud-based server overtakes the 3D map construction and optimization. The server is fed with new raw images from the robots when inconsistencies between the map and current observations appear. In the experiment section, the bandwidth analysis shows that a single robot client induces network traffic of average 1 MB/s which is less than a standard wireless connection usually offers. However, it also makes clear that the number of clients is strongly limited with this approach. Similar to (Riazuelo, Civera et al. 2014), Mohanarajah, Usenko et al. (2015) presents a cloud-based, cooperative visual SLAM

which is able to half the communicational demands by only transmitting key-frames to the cloud-server.

Deutsch, Liu et al. (2016) further reduce the communication demands with their visual graph-based multi-robot SLAM. Each robot runs a single robot SLAM to stay operational also in the case of a network disruption. If the connection to the server is alive, the robots transmit changes of their pose graphs. However, instead of raw image data, only image features are transmitted.

As can be seen from the presented approaches, cooperative SLAM is an active research area, especially using visual sensors. However, most of recent work concentrates on cooperative SLAM (C-SLAM) for initial map building while C-LT-SLAM is not tackled explicitly. To sum up, there exists numerous work tackling single parts of the desired C-LT-SLAM but no integrated approach covering all requirements. This motivates combining and enhancing several state-of-the art approaches. Since the different components of the C-LT-SLAM have strong interdependencies, choosing the most promising approach for each specific component may not lead to a suitable overall system. Instead, the sub-approaches must be carefully chosen to fit the overall system.

By being able to use arbitrary motion and sensor models, particle filters fulfill the requirement of incorporating different sensors in the localization process as well as realize a computational efficient localization solution. Furthermore, with (Tipaldi, Meyer-Delius et al. 2013) and (Valencia, Saarinen et al. 2014), we already find promising particle filter-based LT-SLAM approaches. On the cooperation side, we find the mutual localization system described by Howard, Matark et al. (2002) which also bases on particle filters and seems promising to be enhanced in order to meet not yet resolved requirements. For this reason, particle filters are chosen as the base localization framework of our C-LT-SLAM approach. In the following sections, we will overcome remaining drawbacks of related work by combining different approaches and enhancing them with needed functionalities to meet all requirements of the desired C-LT-SLAM.

## 5.2 Local Long-Term SLAM

As described in the last section, the work from Tipaldi, Meyer-Delius et al. (2013) and Valencia, Saarinen et al. (2014) already provide sound base algorithms to be used for the local LT-SLAM of the C-LT-SLAM. Although, Tipaldi, Meyer-Delius et al. (2013) provide a cell forgetting mechanism, their approach still possesses a higher dependency of the computational demands on the chosen particle number due to the fact that each particle holds and updates its internal set of observed cells compared to (Valencia, Saarinen et al. 2014) where the particles themselves do not hold any map information. When dealing with noisy sensors, having sufficient particles is of crucial importance in order to achieve the

needed robustness against outliers and recover from failures. For this reason, the MCL with MAP pose mapping of (Valencia, Saarinen et al. 2014) is favored when looking solely from that point of view. However, the dual-timescale approach arises several issues. First of all, it assumes the definition of a static map as a priori knowledge without giving information on how to get this map. If it is generated with a standard SLAM, it will not only contain static objects but rather also all semi-static parts. Furthermore, the authors present no strategy for long-term mapping, i.e., how to process the gathered data in the short-term map for succeeding runs, which is crucial for long-term operation. Instead, the short-term map is discarded after each run. Once the static map has significantly diverged from the time it was built, the assumption of only facing incremental changes is violated leading to increased localization inaccuracies and failures. Finally, the authors do not provide a strategy on how to combine the dual-timescale map layers in order to have a unified representation which is usable for path planning.

### 5.2.1 Long-Term SLAM using Dual Confidence Map Layers

Based on theses considerations, the following approach is chosen for the local LT-SLAM. As the core algorithm, we use the MCL with MAP pose mapping of (Valencia, Saarinen et al. 2014) and integrate the forgetting mechanism of Tipaldi, Meyer-Delius et al. (2013) to further reduce computational and memory demands. Since, we are not using plain DOGM but the xGMM-DOGM from Section 4.4, the forgetting mechanism needs to be adjusted in a way that the divergence of the cell parameters is not only evaluated on the HMM parameters but also on the additional parameters of the mixture model. Furthermore, we replace the concept of a dual-timescale with a dual-confidence representation. In this representation, the map $m_t$ consists of a verified layer $m_t^v$ containing only cells with verified information and a non-verified map layer $m_t^{\bar{v}}$:

$$m_t = \left\langle m_t^v, m_t^{\bar{v}} \right\rangle . \tag{5.1}$$

Both map layers hold an instance of the entire map of the environment and are initialized with the same initial map $m_0$:

$$m_0 = m_0^v = m_0^{\bar{v}} \tag{5.2}$$

that was generated with a standard SLAM algorithm during a setup run or constructed from CAD data of the environment.

Similar to (Valencia, Saarinen et al. 2014), a slightly adjusted MCL is used to update the particles. The modification appears in the particle weighting step where now both map layers are used to compute the particle's weight. Equation (2.15) is therefore adjusted to:

$$w_t^{[k]} = w_{t-1}^{[k]}\, p(z_t \mid x_t^{[k]}, m_t^v, m_t^{\bar{v}}). \tag{5.3}$$

The observation model $p(z_t \mid x_t^{[k]}, m_{t-1}^{v}, m_{t-1}^{\bar{v}})$ using the developed map representation from Chapter 4 will be described in Subsection 5.2.2.

Subsequently, the map update step is executed. To avoid mapping with inaccurate localization, the confidence of the current localization belief is computed. Inspired by (Valencia, Saarinen et al. 2014), we define a confidence measure $\tau_t$ based on the trace of the covariance matrix of the a posteriori particle distribution $\Sigma_{x_t}$:

$$\tau_t = exp\Big( - k_{tr} \ \mathrm{Tr}(\Sigma_{x_t})\Big) \tag{5.4}$$

with the predefined gain parameter $k_{tr}$ and $\Sigma_{x_t}$ as the weighted sample covariance:

$$\Sigma_{x_t} = \sum_{k=1}^{N} w^{[k]} \ (x_t^{[k]} - \mu_{x_t})(x_t^{[k]} - \mu_{x_t})^T \tag{5.5}$$

where

$$\mu_{x_t} = \sum_{k=1}^{N} w^{[k]} \ x_t^{[k]}. \tag{5.6}$$

Note that the given formulas in Equation (5.5) and (5.6) compute the full covariance matrix. However, for computing the trace, only the diagonal entries, i.e., the variances of each state, are needed. Computing the off-diagonals can therefore be omitted. Additionally, Equation (5.5) and (5.6) are only valid for continuous non-limited state spaces which does not hold for the angular state in $x_t$. The respective formulas for estimating mean and variance of circular variables are omitted at this point for clarity but can be found in (Fisher 1995).

Given $\tau_t$, we can now decide if the current sensor observation is used for mapping by comparing $\tau_t$ to a predefined threshold $k_\tau$. If $\tau_t$ falls below this threshold, we skip the map update step to minimize the risk of mapping erroneous data. Otherwise, the sensor observation in combination with the MAP pose $x_t^{MAP}$ is used to update $m_t^{\bar{v}}$ by updating its individual cell parameters $\Theta_t^{\bar{v},i}$:

$$p(m_t^{\bar{v}} \mid z_t, x_t^{MAP}, m_{t-1}^{\bar{v}}) = \prod_{i=1}^{N_m} p(\Theta_t^{\bar{v},i} \mid z_t, x_t^{MAP}, \Theta_{t-1}^{\bar{v},i}) \tag{5.7}$$

using the map estimation algorithms given in Chapter 4 and $x_t^{MAP}$ as the pose of the particle $k^\star$ with currently maximum likelihood, i.e., with highest weight $w_t^{[k]}$:

$$k^\star = \arg\max_k \{w_t^{[k]}\}. \tag{5.8}$$

For being able to distinguish between verified and non-verified cells, we add a binary confidence value $\tau_t^{\bar{v},i} = \{\text{confident}, \text{non-confident}\}$ to each cell $i$ of $m_t^{\bar{v}}$. We then model

the probability of $\tau_t^{\bar{v},i}$ depending on the localization accuracy during the cell observation. More detailed, each time $c_i^{\bar{v}}$ is observed, we update $p(\tau_t^{\bar{v},i})$ based on the current localization confidence $\tau_t$ using its log odds form $l(\tau^{\bar{v},i})$ with the recursive update formula:

$$l_t(\tau^{\bar{v},i}) = \log \frac{\tau_t}{1-\tau_t} + l_{t-1}(\tau^{\bar{v},i}). \tag{5.9}$$

$l_t(\tau^{\bar{v},i})$ is then used for the update of $m_t^v$ which only depends on $m_t^{\bar{v}}$ not the observation $z_t$ directly:

$$p(m_t^v \mid m_t^{\bar{v}}, m_{t-1}^v) = \prod_{i=1}^{N_m} p(\Theta_t^{v,i} \mid \Theta_t^{\bar{v},i}, \Theta_{t-1}^{v,i}). \tag{5.10}$$

The update model for the verified cells, $p(\Theta_t^{v,i} \mid \Theta_t^{\bar{v},i}, \Theta_{t-1}^{v,i})$, is defined as follows:

$$\Theta_t^{v,i} = \begin{cases} \Theta_t^{\bar{v},i} & \text{if } l_t(\tau^{\bar{v},i}) > k_v \\ \Theta_{t-1}^{v,i} & \text{otherwise.} \end{cases}. \tag{5.11}$$

With this simple update model, $\Theta_t^{v,i}$ is assigned the respective cell parameters of $\Theta_t^{\bar{v},i}$ in case $l_t(\tau^{\bar{v},i})$ is above a predefined verification threshold $k_v$. Otherwise, it remains unchanged.

The update routine of the presented MCL with MAP pose mapping on dual-confidence maps is given in pseudo-code of Algorithm 5.1. Note that for resampling in line 4.1, we incorporate the proposed algorithm from (Grisetti, Stachniss et al. 2007) which decides if the resampling process is carried out by computing the effective particle number. Moreover, we employ the KLD sampling of (Fox 2003) to dynamically adjust the particle number based on the particle distribution.

### 5.2.2 Observation Likelihood Modeling

We now want to derive the observation model $p(z_t \mid x_t^{[k]}, m_t^v, m_t^{\bar{v}})$ used in Equation (5.3). Since it highly depends on the used map representation, we will derive it for the three presented variations in Chapter 4, i.e., the NDT-DOGM, the GMM-DOGM and the glsxGMM-DOGM. Although the latter is the chosen map representation of this work, presenting the models for all variations improves the comprehension of the differences induced by the different representation. Moreover, it is used to benchmark the different approaches, see Chapter 6.

#### Observation likelihood in NDT-DOGMs

Following Equation (5.3), the observation likelihood, $p(z_t \mid x_t^{[k]}, m_{t-1}^v, m_{t-1}^{\bar{v}})$ is needed for weighting the particles based on the current observation $z_t = \{\zeta_j\}_{j=1}^{N_z}$, the particle's pose $x_t^{[k]}$ and previous map estimate $m_{t-1}$.

**Algorithm 5.1** MCL with MAP pose mapping on dual-confidence maps update

**procedure** UPDATE_MCL_MAP_MAPPING_UPDATE($z_t, X_{t-1}, m_{t-1}$)
  **for all** particles in $X_{t-1}$ **do**
    $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ ▷ Drawing from motion model
    $w_t^{[k]} = w_{t-1}^{[k]}\, p(z_t \mid x_t^{[k]}, m_{t-1}^{v}, m_{t-1}^{\bar{v}})$ ▷ Weight update
  **end for**
  $X_t = \left\langle x_t^{[k]}, w_t^{[k]} \right\rangle$
  $\mu_{x_t} = \frac{1}{N} \sum_{i=1}^{N} w^{[k]}\, x_t^{[k]}$ ▷ A posteriori mean pose
  $\Sigma_{x_t} = \frac{1}{N-1} \sum_{i=1}^{N} w^{[k]}\, (x_t^{[k]} - \mu_{x_t})(x_t^{[k]} - \mu_{x_t})^T$ ▷ A posteriori pose covariance
  $\tau_t = exp\Big( - k_{tr}\ \mathrm{Tr}(\Sigma_{x_t})\Big)$ ▷ A posteriori pose belief confidence
  **if** $\tau_t > k_\tau$ **then**
    $x_t^{MAP} = \max_{w_t^{[k]}} \{x_t^{[k]}\}$ ▷ MAP pose
    **for all** cells $c^{\bar{v},i}$ in $m_{t-1}^{\bar{v}}$ **do** ▷ Update $m_t^{\bar{v}}$
      $\left\{\Theta_t^{\bar{v},i}\right\} = update\_cell\_parameters(z_t, x_t^{MAP}, \Theta_{t-1}^{\bar{v},i})$
    **end for**
    **for all** observed cells **do**
      $l_t(\tau^{\bar{v},i}) = \log \frac{\tau_t}{1-\tau_t} + l_{t-1}(\tau^{\bar{v},i})$ ▷ Update cell confidences
    **end for**
    **for all** cells $c^{v,i}$ in $m_{t-1}^{v,i}$ **do** ▷ Update $m_t^{v}$
      $\Theta_t^{v,i} = \begin{cases} \Theta_t^{\bar{v},i} & \text{if } \Big(l_t(\tau^{\bar{v},i}) > k_v \text{ and } \Theta_t^{\bar{v},i} \text{ is stationary}\Big) \\ \Theta_{t-1}^{v,i} & \text{otherwise.} \end{cases}$
    **end for**
  **end if**
  $X_t = resample(X_t)$
  **return** $X_t, m_t$
**end procedure**

Following Biber and Strasser (2003) as well as Saarinen, Stoyanov et al. (2013), we also apply an NDT on $z_t$, i.e., convert the 2D scan measurement into a set of normal distributions. First, all scan points $\zeta_j$ are transformed from the sensor frame $\mathcal{S}$ into the global map frame $\mathcal{M}$ using $x_t^{[k]}$ as the robot pose given in $\mathcal{M}$. Afterwards, the NDT of the scan is carried out by computing the normal distribution parameters $\theta_z^{NDT,i,j'} = \left\langle \mu_z^{i,j'}, \Sigma_z^{i,j'} \right\rangle$ based on all $\zeta_j$ that fall into the boundaries of cell $i$ using Equation (2.18) and (2.19):

$$z_t = \{\zeta_j\} \rightarrow z'_t = \{\theta_z^{NDT,i,j'}\}. \tag{5.12}$$

The probability of observing $\theta_z^{NDT,i,j'}$ is then computed by multiplying the occupancy $p(q_t^i = o)$ of $c_i$ with the likelihood of $\theta_z^{NDT,i,j'}$ given the cell's NDT:

$$p(\theta_z^{NDT,i,j'} \mid x_t^{[k]}, m_{t-1}) = \sum_{i=1}^{N_m^*} p(q_t^i = o)\ p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,i}). \tag{5.13}$$

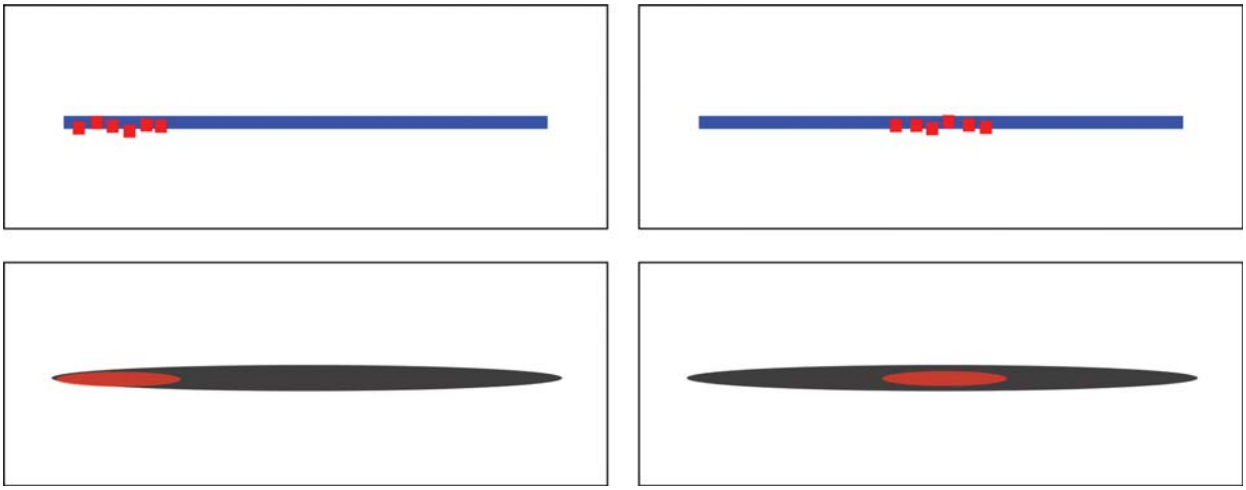


Figure 5.1: Different observations of cell containing a wall: Environment structure (blue line) and resulting scan points (red squares) when observed by the robot from different positions (top left/ top right), map and observation model view (bottom) where drawn ellipse base on cell parameters $\theta_t^{NDT,i}$ (grey) and observed NDT $\theta_z^{NDT,i,j'}$ (red).

In general, the likelihood is computed by summing over the likelihood of each cell in $m_t$. However, in real-world environments, this gets impractical. Since the likelihood of $\theta_z^{NDT,i,j}$ quickly gets close to zero for cells with an increasing distance to $c_i$, it is both reasonable and computational efficient to only evaluate the cells $c_{i^*}$ in the close neighborhood of $c_i$ where $N_m^* \ll N_m$.

In (Stoyanov, Magnusson et al. 2012) and (Saarinen, Andreasson et al. 2013c), the $L_2$-likelihood is used for evaluating the likelihood of $\theta_z^{NDT,i,j'}$:

$$p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,i}) = \exp\left[-\frac{1}{2}\left(\mu_z^{i,j'} - \mu_t^i\right)\left(\Sigma_{z^{i,j'}} + \Sigma_t^i\right)^{-1}\left(\mu_z^{i,j'} - \mu_t^i\right)^T\right]. \tag{5.14}$$

However, this likelihood measure puts a strong weight on the distance between the means of the two normal distributions. With LR grid maps, we often face situations where only a part of the object within the cell is observed. The resulting NDT observations often exhibit a shifted mean while their covariance ellipse perfectly fit into the ellipse of the cell's NDT. An example scenario is illustrated in Figure 5.1. Both measurements in this example observe a part of equal length of the wall contained by the cell. Intuitively, the likelihood of the two measurements should be close to each other. However, due to the significant difference between the means, the observation on the left of Figure 5.1 will have a significantly lowered $L_2$-likelihood. In this case, a more elaborated measure bases on the overlapping region of the two ellipses. However, computing this region is computational intense in contrast to the $L_2$-likelihood which can be computed efficiently. Yang, Baum et al. (2016) investigate the general problem of comparing elliptic shapes using different metrics. For our specific

**Algorithm 5.2** Observation likelihood for NDT-OGM

**procedure** OBSERVATION_LIKELIHOOD_NDT_DOGM($z_t, x_t, m_{t-1}$)
  **for all** $\zeta_j$ in $z_t$ **do**
    $\zeta_j = transform\_to\_map\_frame(\zeta_j)$
  **end for**
  ${z'}_t = extract\_NDTs(z_t)$
  **for all** components in ${z'}_t$ **do**
    $p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,i}) = k(N_{i,j'}) L_{KLD}\left(\theta_z^{i,j'} \mid\mid \theta_t^{NDT,i}\right)$
    $p(\theta_z^{NDT,i,j} \mid x_t^{[k]}, m_{t-1}) = \sum_{i=1}^{N_m^*} p(q_t^i = o)\; p(\theta_z^{NDT,i,j} \mid \theta_t^{NDT,i})$
  **end for**
  $p(z_t \mid x_t^{[k]}, m_{t-1}^{\bar{v}}) = \sum_{j=1'}^{N_z^{NDT,i,j'}} \sum_{i=1}^{N_{m^{\bar{v}}}^*} p(q_t^{\bar{v},i} = o)\; p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,\bar{v},i})$
  $p(z_t \mid x_t^{[k]}, m_{t-1}^{v}) = \sum_{j=1'}^{N_z^{NDT,i,j'}} \sum_{i=1}^{N_{m^{v}}^*} p(q_t^{v,i} = o)\; p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,v,i})$
  $p(z_t \mid x_t^{[k]}, m_{t-1}^{v}, m_{t-1}^{\bar{v}}) = k_v\; p(z_t \mid x_t^{[k]}, m_{t-1}^{v}) + (1 - k_v)\, p(z_t \mid x_t^{[k]}, m_{t-1}^{\bar{v}})$
  **return** $p(z_t \mid x_t^{[k]}, m_{t-1}^{v}, m_{t-1}^{\bar{v}})$
**end procedure**

application, we find the KLD as a suitable trade-off between accuracy and computational complexity. The KLD applied to the comparison of the two NDT is defined as:

$$L_{KLD}\left(\theta_z^{i,j'} \mid\mid \theta_t^{NDT,i}\right) = \frac{1}{2}\left( \mathrm{Tr}\left(\left(\Sigma_t^i\right)^{-1} \Sigma_z^{i,j'}\right) + \left(\mu_z^{i,j'} - \mu_t^i\right)^T \left(\Sigma_t^i\right)^{-1} \left(\mu_z^{i,j'} - \mu_t^i\right) - 2 + \ln\left(\frac{\det\, \Sigma_t^i}{\det\, \Sigma_z^{i,j'}}\right)\right). \quad (5.15)$$

Using $L_{KLD}$, we model the KLD-based observation likelihood as follows:

$$p(\theta_z^{i,j'} \mid \theta_t^{NDT,i}) = k(N_{i,j'}) L_{KLD}\left(\theta_z^{i,j'} \mid\mid \theta_t^{NDT,i}\right) \quad (5.16)$$

where $k(N_{i,j'})$ is a scaling factor which accounts for the number of points $N_{i,j'}$ used to compute $\theta_z^{i,j'}$. The likelihood of $z_t$ can finally be computed by summing the likelihoods of all $\theta_z^{NDT,i,j'}$:

$$p(z_t \mid x_t^{[k]}, m_{t-1}) = \sum_{j'=1}^{N_{NDT,i,j'}} \sum_{i=1}^{N_m^*} p(q_t^i = o)\; p(\theta_z^{NDT,i,j'} \mid \theta_t^{NDT,i}). \quad (5.17)$$

For our dual-confidence approach, Equation (5.17) is now evaluated on both confidence layers:

$$p(z_t \mid x_t^{[k]}, m_{t-1}^{v}, m_{t-1}^{\bar{v}}) = k_v\; p(z_t \mid x_t^{[k]}, m_{t-1}^{v}) + (1 - k_v)\, p(z_t \mid x_t^{[k]}, m_{t-1}^{\bar{v}}) \quad (5.18)$$

where the user-defined parameter $k_v \in [0, 1]$ enables a weighting to regulate the influence of the specific confidence layer on the final result.

### Observation model for GMM-DOGMs

We recall when using GMM-DOGMs, the geometry of the object within the cell is approximated with a set of normal distributions $\theta^{GMM,i} = \left\langle \{w_\psi^i, \theta_\psi^{NDT,i}\}_{\psi=1}^{N_{gmm}^i} \right\rangle$ instead of a single component like in the NDT-OGMs. In accordance with the observation likelihood computation for NDT-OGMs, a GMM $\theta_z^{GMM,i,j'}$ is extracted for each cell based on the current observation $z_t$:

$$z_t = \{\zeta_j\} \rightarrow z'_t = \left\{\theta_z^{GMM,i,j'}\right\}. \tag{5.19}$$

Following the PDF of GMMs, the likelihood of observing $\theta_z^{GMM,i,j'}$ depending on the GMM $\theta_t^{GMM,i}$ of the $i^{th}$ cell is computed with:

$$p\left(\theta_z^{GMM,i,j'} \mid \theta_t^{GMM,i}\right) = \sum_{\psi=1}^{N_{gmm}^i} \sum_{\psi*=1}^{N_{gmm}^{i,j'}} w_\psi^i w_{\psi*}^{i,j'} p\left(\theta_{\psi*}^{NDT,i,j'} \mid \theta_\psi^{NDT,i}\right). \tag{5.20}$$

Since we have already defined $p\left(\theta_{\psi*}^{NDT,i,j'} \mid \theta_\psi^{NDT,i}\right)$ in Equation (5.16), we now have everything to compute $p\left(\theta_z^{GMM,i,j'} \mid \theta_t^{GMM,i}\right)$. Finally, for getting the likelihood of $z_t$, we just need to slightly alter Equation (5.17) to:

$$p(z_t \mid x_t^{[k]}, m_{t-1}) = \sum_{j=1}^{N_z^{NDT,i,j'}} \sum_{i=1}^{N_m^*} p(q_t^i = o)\; p(\theta_z^{GMM,i,j'} \mid \theta_t^{GMM,i}). \tag{5.21}$$

### Observation model for xGMM-DOGMs

With the previous results, we can finally derive the observation likelihood for the xGMM-DOGMs as the map representation used for the C-LT-SLAM. As we will see, the modification with respect to the presented likelihood computation for GMM-DOGMs is minor. Instead of GMMs, we now need to extract xGMMs from $z_t$ and then have to define the likelihood of observing $\theta_z^{xGMM,i,j'}$ depending on the xGMM $\theta_t^{xGMM,i}$ of the $i^{th}$ cell. Again, we find most of the needed information to do this in the definition of the xGMM of Section 4.4 or more specifically in Equation (4.35) which leads us to:

$$p\left(\theta_z^{xGMM,i,j} \mid \theta_t^{xGMM,i}\right) = \sum_{\psi=1}^{N_{xGMM}^i} \sum_{\psi*=1}^{N_{xGMM}^{i,j}} w_\psi^i w_{\psi*}^{i,j} p\left(\theta_{\psi*}^{NDT,i,j} \mid \theta_\psi^{NDT,i}\right) p\left(R_\psi^{i,j} \mid R_{\psi*}^i\right) \tag{5.22}$$

where we just added the reflectivity observation likelihood $p\left(R_\psi^{i,j} \mid R_{\psi*}^i\right)$ to Equation (5.21). Derived from the reflectivity likelihood of a single scan point in Equation (4.36), we define

$$p\left(R_\psi^{i,j} \mid R_{\psi*}^i\right) = R_\psi^{i,j^T}\; B_z^\rho\; R_{\psi*}^i. \tag{5.23}$$

## 5.3 Mutual Localization

Based on the described approach for the local LT-SLAM, the extension to a mutual localization system will be the focus of this section. The main task consists in how to integrate the detection measurements into the localization process as well as how to realize an efficient communication among the agents. With the presented work of Fox, Burgard et al. (2000), we already find a base algorithm for this which meets many of our requirements, i.e., it realizes a distributed approach building upon MCL and employing a detection model that incorporates both, uncertainties of false detections as well as of the measured pose. However, if we recall the requirements of Subsection 5.1.1, the approach does not cover everything needed for our application. More specifically, it relies on known associations which impedes a versatile usage. Moreover, by modeling the pose measurement uncertainty as normal distributed, it further limits possible detection modules usable for our cooperative system. For instance, a Lidar-based detection module which tries to detect a quadratic-shaped robot is not able to output an unambiguous estimate of the detected robot's orientation due to the 4-fold rotational symmetry of the square. This example further indicates that possible outputs of a detection module strongly depend on the used sensor type as well as the unambiguity of the measured property (e.g. the geometric contour of the robot in our example). To overcome this issue, the desired approach should be able to deal with a wide range of possible probability distributions representing the measurement belief.

Following this idea, we need to have a closer look at possible outputs and measurement uncertainties of typical detection systems. First of all, we consider marker-based approaches where each robot is equipped with a unique marker at a highly visible position and a respective sensor that is able to identify and measure the pose of the marker, e.g., a quick response (QR) tag in combination with a vision sensor or a pattern of retro-reflective markers with a Lidar sensor. Depending on the sensor accuracy and the detection algorithm, these approaches commonly provide a unimodal pose belief of the detected robot in its sensor frame $\mathcal{S}$. The uncertainty can then usually be well approximated as normal distributed with the mean pose vector and the respective covariance matrix accounting for position and orientation noise of the measurement process. Additionally, associations can be inferred from the detected marker.

In contrast, configuration-based approaches do not rely on artificial markers but detect other robots based on a observable configuration (shape, color, etc) of the robot or a robot type. Knowledge about the possible configurations of the robots in the fleet is either given as a priori knowledge or can be learned from training data. We already mentioned such an approach in terms of a Lidar-based detection module using the geometric contour to detect and measure other robots. In this approach, the modality of the pose belief highly depends

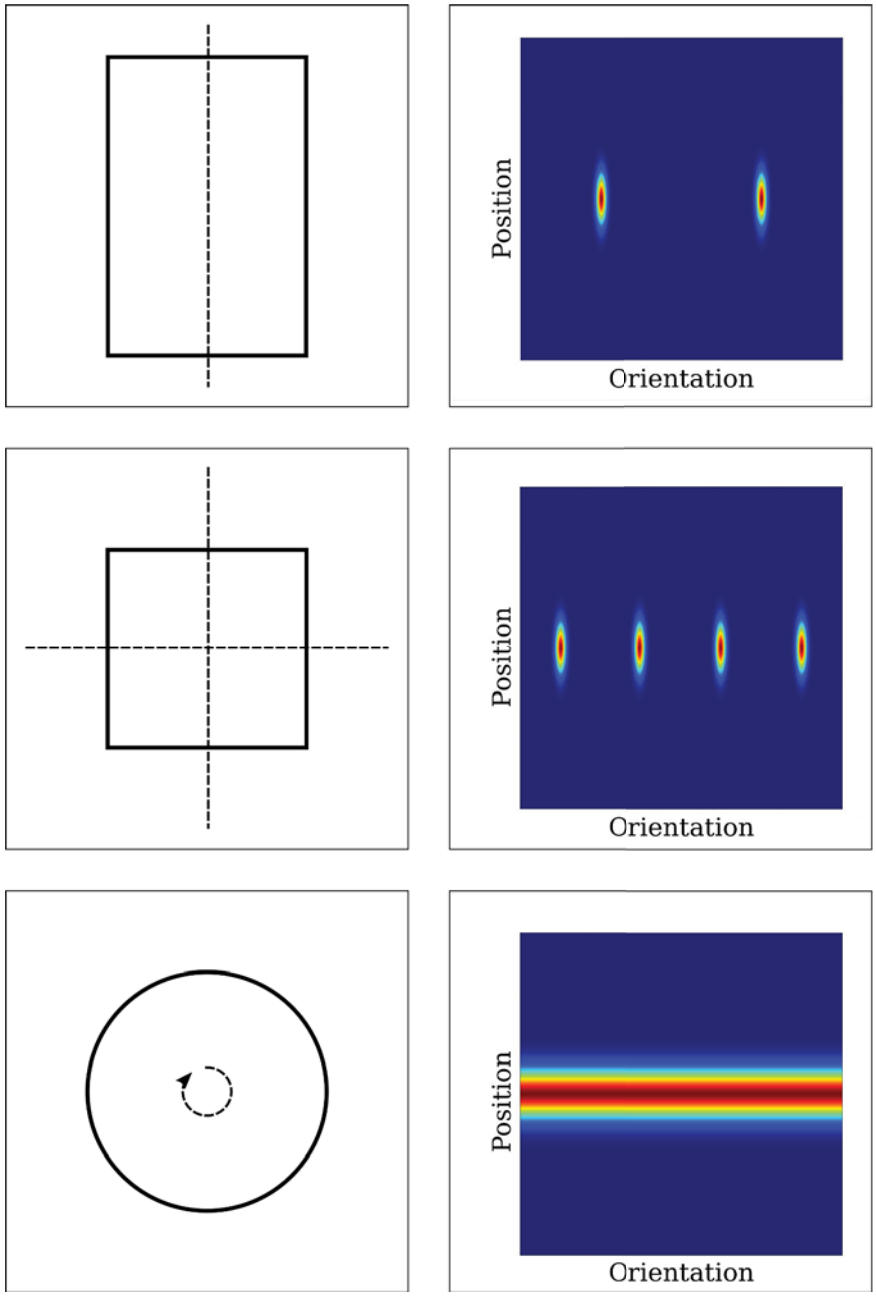



Figure 5.2: Common robot shapes and possible implications for contour-based detection. Left: rectangular (top), squared (middle) and circular-shaped (bottom) robot with rotation symmetry lines. Right: Exemplary GMM modeling the uncertainty of the relative 2D pose of the robot when observed by another robot.

on the geometric shape of the robots that need to be detected. For clearly non-symmetrical shapes, the detection module may be able to output a unimodal belief similar to the previously discussed marker-based approaches. In contrast, symmetric shapes will always lead to an ambiguity mostly due to their manifold rotational symmetry. As an example, see the 2-fold, 4-fold and infinity-fold rotational symmetry of a rectangular, squared and circular-shaped robot in Figure 5.2. Note that so far we considered the geometrical symmetry of

the entire shape. However, a sensor observation will in most cases only partially observe the shape of the object which could lead to further ambiguities. For example, when only observing a corner of a rectangular-shaped robot, the previously 2-fold modality becomes 4-fold since the observed corner may be indistinguishable from the other three. Moreover, while in the so far discussed examples the multi-modality only appears in the orientation belief, one can also construct shapes where it affects the position belief.

These examples demonstrate that a multi-modal measurement belief is necessary to cover the ambiguities of common robot shapes. However, they also show that we commonly find only a small number of modalities where each modality exhibits a clearly single peaked probability distribution. For representing the measurement belief in our mutual localization approach, we therefore chose the following mixture model. The spatial uncertainty of the measurement is modeled with a GMM since it perfectly covers the multi-modalities and has suitable computational properties. Additionally, to deal with classification uncertainties of the deployed detection module, we incorporate the probabilities of a true-positive detection $p_{TP}$ as well as false-positive detection $p_{FP}$, leading to the following measurement model:

$$p\left(z_t^{(\xi,\chi)} \mid \theta_z^{(\xi,\chi)}\right) = p_{TP} \sum_{\psi_z=1}^{N_z} w_{\psi_z} p\left(z_t^{(\xi,\chi)} \mid \mu_{\psi_z}, \Sigma_{\psi_z}\right) + p_{FP} \tag{5.24}$$

where $z_t^{(\xi,\chi)}$ is the relative pose of detected robot $\xi$ observed by robot $\chi$ in its sensor frame $\mathcal{S}_\chi$. $\theta_z^{(\xi,\chi)}$ represents the parameters of the mixture model. The second term of Equation (5.24) does not depend on $z_t^{(\xi,\chi)}$ and thereby adds a uniform distribution over the whole state space which is desired since a FP detection does not give any information about the location of robot $\xi$. $p_{TP}$ and $p_{FP}$ are assumed to be given by the detection module. Otherwise, it can be estimated from experience or learned based on a labeled data set (see also (Fox, Burgard et al. 2000)).

Based on this relative detection measurement model, we now want to investigate its usage to update the localization belief of both detected and detector robot. In the following section, we simplify the problem in terms of assuming known associations in order to be able to focus on the localization update. The subsequent section then extends this concept for the case of unknown associations.

### 5.3.1 Mutual Localization with Known Associations

Following (Fox, Burgard et al. 2000), we define the belief of robot $\chi$ about the pose of robot $\xi$ based on $z_t^{(\xi,\chi)}$ and its own pose belief $x_t^{(\chi)}$ at time $t$:

$$bel_\chi\left(x_t^{(\xi)}\right) = \int p\left(x_t^{(\xi)} \mid z_t^{(\xi,\chi)}, x_t^{(\chi)}\right) \; bel_\chi\left(x_t^{(\chi)}\right) dx_t^{(\chi)}. \tag{5.25}$$

Equation (5.25) is then used to update robot $\chi$'s belief about its own pose with:

$$bel'_{\xi}\left(x_t^{(\xi)}\right) = bel_{\xi}\left(x_t^{(\xi)}\right) bel_{\chi}\left(x_t^{(\xi)}\right). \tag{5.26}$$

Note that due the symmetry, Equation (5.25) and (5.26) can be used in the same way for updating the pose belief of $\chi$ just by flipping the respective terms.

In Equation (5.25), the detection model $p\left(x_t^{(\xi)} \mid z_t^{(\xi,\chi)}, x_t^{(\chi)}\right)$ represents the probability of robot $\xi$ being in pose $x_t^{(\xi)}$ when detected by $\chi$ from pose $x_t^{(\chi)}$ with the relative pose measurement $z_t^{(\xi,\chi)}$. We can infer this model by transforming our relative detection model from Equation 5.24 into the global map frame $\mathcal{M}$ using the transformation $T\left(x_t^{(\chi)}\right)$ based on robot $\chi$'s pose estimate:

$$p\left(x_t^{(\xi)} \mid z_t^{(\xi,\chi)}, x_t^{(\chi)}\right) = p_{TP} \sum_{\psi_z=1}^{N_z} w_{\psi_z} p\left(x_t^{(\xi)} \mid [\mu_{\psi_z}]^{\mathcal{M}}, [\Sigma_{\psi_z}]^{\mathcal{M}}\right) \tag{5.27}$$

where $[\mu_{\psi_z}]^{\mathcal{M}} = T\left(x_t^{(\chi)}\right)\mu_{\psi_z}$ and $[\Sigma_{\psi_z}]^{\mathcal{M}} = T\left(x_t^{(\chi)}\right)^T \Sigma_{\psi_z} T\left(x_t^{(\chi)}\right)$. Equation (5.27) thereby executes a transformation of the (spatial) model parameters of $\theta_z^{(\xi,\chi)}$ into $\mathcal{M}$, or formally: $\left[\theta_z^{(\xi,\chi)}\right]^{\mathcal{S}_\chi} \rightarrow \left[\theta_z^{(\xi,\chi)}\right]^{\mathcal{M}}$.

Realizing the pose update of Equation (5.26) with Equation (5.25) and (5.27) would thereby involve transmitting $\theta_t^z$ and $bel(x_t^{(\chi)})$ from robot $\chi$ through the wireless network to robot $\xi$. However, communicating the sample-based representation of $bel(x_t^{(\chi)})$ is inefficient, especially when using large particle sets commonly needed to deal with noisy sensor data. Moreover, it would involve a multiplication of two sample-based densities, i.e., the pose beliefs of robot $\chi$ and $\xi$ in Equation (5.26), which cannot be carried out straightforwardly. As already discussed, Fox, Burgard et al. (2000) resolves this issue by converting the sample-based pose belief $bel_{\chi}\left(x_t^{(\chi)}\right)$ into a QT representation. This seems appropriate for their application of mutual global localization where the particles can be distributed widely around the whole environment. However, since global localization is not the scope of this work, we seek for a more efficient representation in order to lower computational and communication demands. In our application, the particles are most of the time closely spread around the actual pose building a unimodal density distribution which is also a prerequisite to achieve the demanded localization accuracies. In challenging areas with sparse or symmetric environmental structure, the particles can get more distributed, forming into a few clusters. Based on these properties, we propose a conversion of the pose belief into GMM as a compact and efficient representation to be processed in our mutual localization update routine:

$$bel_{\chi}\left(x_t^{(\chi)}\right) \sim p\left(x_t^{(\chi)} \mid \theta_{\chi}^{GMM}\right) = \sum_{\psi_x=1}^{N_x} w_{\psi_x} p\left(x_t^{(\chi)} \mid \mu_{\psi_x}, \Sigma_{\psi_x}\right). \tag{5.28}$$

$\theta_{\xi}^{GMM}$ can be estimated by using the particle set $X_t^{(\chi)}$ and the EM algorithm described in Subsection 2.1.4. Since this EM does not handle weighted samples, we can either generate unweighted samples from $X_t$ comparable to the resampling step in MCL or use an extended EM that is able to incorporate the weights, see for instance (Gebru, Alameda-Pineda et al. 2016).

Another advantage of using GMMs for both, representing the measurement belief as well as the pose belief of $\chi$, is that their multiplication is straightforward. Using Equation (5.27) - (5.28), we can compute Equation (5.25) with:

$$bel_{\chi}\left(x_t^{(\xi)}\right) = p_{TP} \sum_{\psi_z=1}^{N_z} \sum_{\psi_x=1}^{N_x} w_{z,x}\, p\left(x_t^{(\xi)} \mid \mu_{z,x}, \Sigma_{z,x}\right) + p_{FP} \tag{5.29}$$

where

$$w_{z,x} = w_{\psi_z} w_{\psi_x} \tag{5.30}$$

$$\mu_{z,x} = T\left(\mu_{\psi_x}\right) \mu_{\psi_z} + \mu_{\psi_x} \tag{5.31}$$

$$\Sigma_{z,x} = T\left(\mu_{\psi_x}\right) \Sigma_{\psi_z} \left(T\left(\mu_{\psi_x}\right)\right)^T + \Sigma_{\psi_x}. \tag{5.32}$$

We will denote hereinafter the parameter set from Equation (5.29) as $\theta_{z,x}^{(\xi,\chi)}$.

Finally, the update of $bel'_{\xi}\left(x_t^{(\xi)}\right)$ can be carried out by updating the importance weights $w_t^{[k](\xi)}$ of robot $\xi$ using Equation (5.29):

$$w'^{[k](\xi)}_t = w_t^{[k](\xi)} \left( p_{TP} \sum_{\psi_z=1}^{N_z} \sum_{\psi_x=1}^{N_x} w_{z,x}\, p\left(x_t^{[k],(\xi)} \mid \mu_{z,x}, \Sigma_{z,x}\right) + p_{FP} \right). \tag{5.33}$$

The whole update procedure of a detection measurement with known association is illustrated in Figure 5.3. It starts with the detection module of robot $\chi$ signalizing a new detection with measurement parameters $\theta_z^{(\xi,\chi)}$ which triggers the detection handler of the local LT-SLAM to compute the GMM of the current particle distribution $\theta_{\chi}^{GMM}$ and $\theta_{z,x}^{(\xi,\chi)}$ respectively. The latter is transmitted as a detection message to the detection handler of the LT-SLAM server in the cloud. Since the association is known, the detection handler directly forwards the message to robot $\xi$ which uses the information to update its particle weights according to Equation (5.33). Subsequently, robot $\xi$ communicates its GMM-based pose belief $\theta_{\xi}^{GMM}$ back to robot $\chi$ which then itself is able to use the inversed detection measurement altogether with $\theta_{\xi}^{GMM}$ to update its own pose belief. Same as in (Fox, Burgard et al. 2000), we forbid successive detections until one of the robots has significantly moved to avoid incorporating the same evidence multiple times.

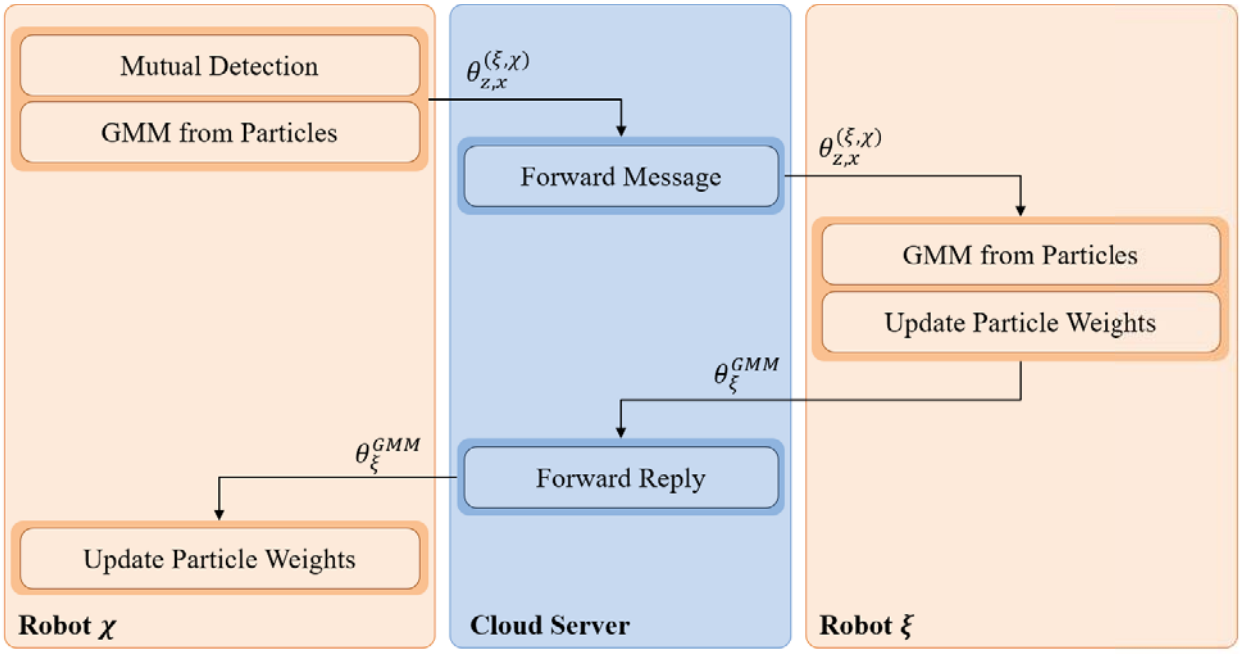



Figure 5.3: Information flow for mutual localization with known associations.

Furthermore, Figure 5.4 illustrates the mutual detection procedure in a simplified 1D example where the pose and measurement belief can be constructed with a single-component GMM.

The derived formulas for mutual detection so far considered two mobile agents being involved in the process. However, this functionality is also useful for the situation of a static agent detecting a mobile agent. Since we assume the static agent's pose to be known, the procedure simplifies in a way that the pose belief of $\chi$ (as the static detector) is static and has minor or no uncertainty and does not need to be updated.

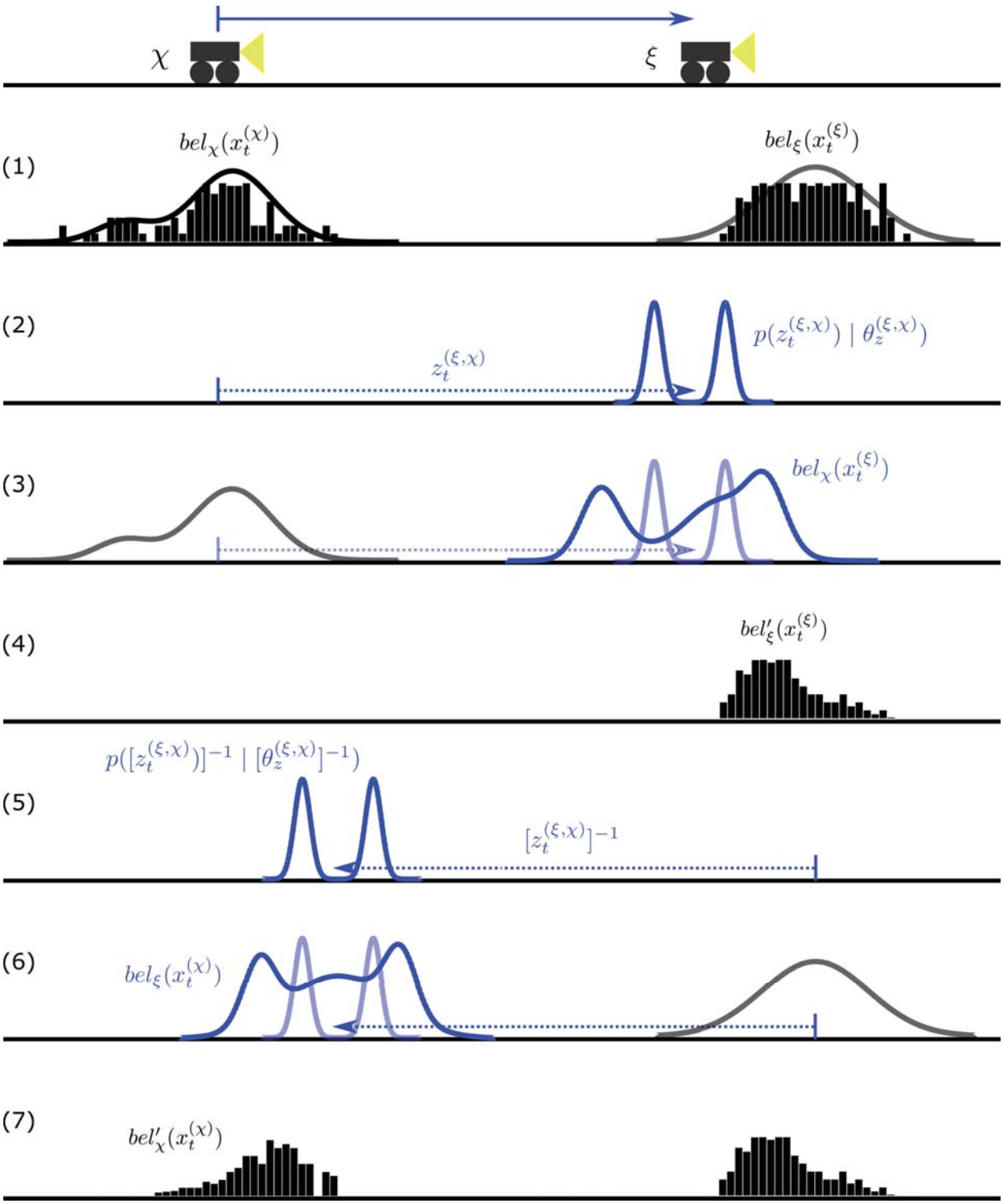



Figure 5.4: Mutual detection procedure in simplified 1D example of robot $\chi$ detecting $\xi$. (1) Sample-based (bars) and GMM (curved line) pose beliefs of $\chi$ as well as $\xi$ prior to the detection, (2) Relative pose measurement belief with dual component GMM, (3) pose belief of $\chi$ about pose of $\xi$ (dark blue) constructed from the measurement belief (light blue) and $\chi$'s own pose belief (grey), (4) updated pose belief of robot $\xi$, (5) Inverted measurement belief, (6) Pose belief of $\xi$ about pose of $\chi$ (dark blue) constructed from the measurement uncertainty (light blue) and $\chi$'s prior pose belief (grey), (5) Updated pose belief of robot $\chi$

.

### 5.3.2 Mutual Localization with Unknown Associations

The presented concept of mutual localization so far bases on the assumption of known associations, i.e., robot $\chi$ knows the identity of detected robot $\xi$. This identification is trivial, e.g., if the detection module bases on unique markers mounted on each robot. In contrast, configuration-based approaches are in general not able to identify the detected robot uniquely but only the type of the robot. Since we do not restrict our mutual localization on marker-based detection modules nor on fleets exhibiting only one robot per type, we also need to provide a solution for the case of unknown association.

Based on the knowledge about the robot types, the problem of finding the detection association becomes finding robot $\xi$ among a subset of robots $\upsilon$, $\upsilon \in [1, N_{type}], N_{type} \leq N_{fleet}$, where $N_{type}$ is the number of robots of the detected type present in the fleet and $N_{fleet}$ the total number of robots. Since we need to deal with uncertainties, we follow a probabilistic association approach where we compute for each robot $\upsilon$, the likelihood $L^{\upsilon=\xi}$ of being detected robot $\xi$. This can be carried out by computing the likelihood of the current pose belief of robot $\upsilon$ (see Equation (5.28)) based on the belief of robot $\chi$ about the pose of $\xi$ (see Equation (5.29)):

$$\begin{aligned} L^{\upsilon=\xi} &= p\left(\theta_{(\upsilon)}^{GMM} \mid \theta_{z,x}^{(\xi,\chi)}\right) \\ &= \sum_{\psi_x^{(\upsilon)}=1}^{N_z^{(\upsilon)}} \sum_{\psi_z^{(\chi)}=1}^{N_z^{(\chi)}} \sum_{\psi_x^{(\chi)}=1}^{N_x^{(\chi)}} w_{\psi_x}^{(\upsilon)} w_{z,x}^{(\chi)}\; p\left(\mu_{\psi_x}^{(\upsilon)} \mid \mu_{z,x}^{(\chi)}, \Sigma_{z,x}^{(\chi)} + \Sigma_{\psi_x}^{(\upsilon)}\right). \end{aligned} \tag{5.34}$$

Consequently, the association probabilities $p_a^i$ are computed with:

$$p_a^{\upsilon=\xi} = \frac{L^{\upsilon=\xi}}{\left(\sum_{\beta=1}^{N_{type}} p_{TP}\; L_z^{\beta=\xi}\right) + p_{FP}} \tag{5.35}$$

where we sum in the denominator over all $L^{\upsilon=\xi}$ and add the possibility of $z_t^{(\xi,\chi)}$ being a false detection. We then update the pose belief of each robot $\upsilon$ by using Equation (5.33) and additionally accounting for its association probability:

$$w'^{[k](\upsilon)}_t = w_t^{[k](\upsilon)} \left( p_a^{\upsilon=\xi} \sum_{\psi_z=1}^{N_z} \sum_{\psi_x=1}^{N_x} w_{z,x}\; p\left(x_t^{[k],(\xi)} \mid \mu_{z,x}, \Sigma_{z,x}\right) + (1 - p_a^{\upsilon=\xi}) \right). \tag{5.36}$$

In the same way, the particles of $\chi$ are weighted using the $\theta_{(\upsilon)}^{GMM}$ of all robots $\upsilon$ with $p_a^{\upsilon=\xi} > 0$.

Figure 5.5 illustrates the procedure of mutual localization with unknown associations. In contrast to the known associations case, the server requests the GMM-based pose beliefs $\theta_{\upsilon}^{GMM}$ from all robots of the detected type. It then computes the association probabilities $p_a^{\upsilon=\xi}$ and transmits them to all robots $\upsilon$ together with $\theta_{\upsilon}^{GMM}$ in order to be able to update

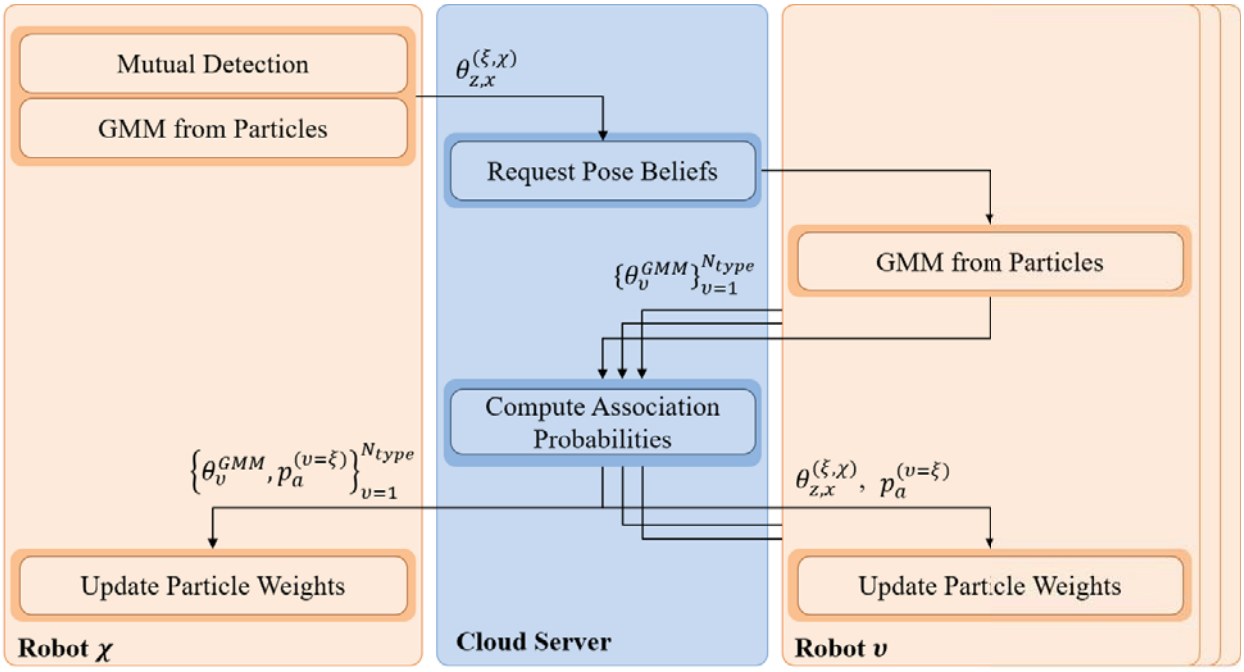



Figure 5.5: Information flow for mutual localization with unknown associations.

their pose beliefs. Detector robot $\chi$ also receives the set of $p_a^{v=\xi}$- for all $v$ in combination with their pose beliefs $\theta_v^{GMM}$ to update its own pose belief.

## 5.4 Cooperative Map Updating

With cooperative map updating, we aim at using the detected map changes of each agent to keep the global map up-to-date. In Chapter 4, we already presented our base concept in terms of a hierarchical approach where each agent (in terms of its local LT-SLAM) transmits the detected local changes of the map to a central LT-SLAM server which is responsible for the fusion of incoming map changes and distribution of map updates. This section now provides further details on the concept and procedure behind this. We will see that by carefully considering potential enablers and requirements of cooperative map updating from the beginning when designing the overall navigation system, map representation and localization concept, we now can reap the rewards by choosing a comparable simple approach to achieve the desired functionality. More specifically, we can rely on the following conditions:

1. All agents localize with respect to the same global map frame.
2. All agents map into an individual instance of the same global grid map.
3. Map changes can be represented as a set of diverged cells of this global grid map.
4. Different map instances can easily be compared by investigating their respective cells.

Our cooperative map updating basically consists of three parts. First, the agents collect and transmit detected map changes to the server referred to as map upstreaming. Second, the server fuses all upstreams into the global map and third, map downstreaming which includes the preparation and transmission of agent-individual map updates on the server side as well as the fusion of the received map changes in the local map of the agents.

### 5.4.1 Map Upstreaming

This part starts with the process of extracting detected changes on the agent side. When reviewing the approach of the local LT-SLAM from Section 5.2, we find that it already holds information about recently observed changes in from of the cells of its unverified map layer. However, since we need to minimize the chance to transmit erroneous data to the server, simply transmitting the recently updated cells of this layer is no option. Instead, we wait until it gets verified, i.e., when the cell is written to the verified layer. Moreover, this filters out cells exhibiting only minor changes. By collecting the cells that are written to the verified map layer in the measurement update step, we find access to detailed and trustworthy information about map changes in a highly compact representation, i.e., as a set of cells that exhibit significant changes in at least one of their cell parameters. The map upstream process can thereby simply be defined as follows. The set of cells written into the verified map layer since the last upstream are collected until they are transmitted to the server using a period upstream cycle. The frequency of the upstream cycle can be chosen in favor of a suitable network load.

### 5.4.2 Map Fusion

Once the upstream is received at the server side, it needs to be fused into the global map. As described above, the upstreamed cells have a direct relation to the global grid map given by the cells' indices. Map association, commonly involving rather complex transformation or cell matching strategies, is thereby superseded. Instead, the association process reduces to picking the respective cell in the global with corresponding index. The remaining part deals with finding an appropriate strategy on how to fuse the parameters of two cells. For this issue, we apply a simple but effective approach in terms of a "winner-takes-all" strategy basing on the timestamp $t_z^i$ of the latest observation of the cells. More specifically, we fuse two cells by simply comparing their timestamps and write the parameters of the cell with the more recent timestamp into the merged cell. This might sound inappropriate at first glance, e.g., when the "winner" cell bases its parameter estimation only on a single recent observation while the "loser" cell contains profound knowledge on the base of long-term observations of the specific object within the cell. Additionally, a proper fusion of each cell parameter incorporating the information of both cells seems necessary in order to not

loose valuable information. However, if we have a closer look, the given example is unlikely to happen due to the fact that the merged cells are constantly shared with the agents which incorporate these cells into their local maps. So instead of strongly diverging, the two cells of our previous example would have already been merged in previous time steps. As a result, the "winner" cell already contains most of the knowledge of the "loser" cell and has incorporated his recent observation upon this. With this insight, we see that our fusion strategy makes highly sense. Nevertheless, we can still construct situations where the strategy is suboptimal. For instance, two agents observing a rectangular object within a cell from opposing positions, e.g., each one observes just one side of the rectangle. In this case, instead of integrating the components of both sides into the merged cell, we end up with a merged cell only containing one side, namely the one that has been observed most recently. However, these special situations are rare and in most cases quickly resolved after some movement of the involved agents. Moreover, an actual fusion of the single cell parameters would increase the computational complexity and would require more information about internal states of the respective parameter estimation algorithms (e.g. internal states of the EM for estimating the transition matrix parameters of the HMM in DOGM). While the computational complexity issue can be seen as minor due to the massive resources of the cloud server, the latter issue is problematic since it increases the data volume of each cell to be communicated from agent to server and vice versa.

### 5.4.3 Map Downstreaming

To provide the agents the latest changes of the global map, the LT-SLAM server collects modified cells from its fusion process. Like in the upstream process, these cells are periodically transmitted to the agents or can directly be requested by the agent. To further decrease the data volume of the downstream, cells can be filtered by their relevance for the respective agents. The filtering bases on a region of interest considering the current pose and global path of the agent. Only cells within the region of interest are transmitted. On the agent side, the received cells are fused into the verified map layer. Since these cells may also have been updated internally, we employ the same fusion strategy as on the LT-SLAM server. At startup of an agent, the map downstream contains the entire global map used by the agent to initialize its map layers $m_0^v$ and $m_0^{\bar{v}}$.

## 5.5 Conclusion

This chapter presented the C-LT-SLAM which embeds in the overall cooperation navigation architecture presented in Chapter 3 for providing high-frequency, robust and accurate localization of the mobile agents as well as up-to-date maps of the environment to be used

for path planning. More specifically, the approaches for the local LT-SLAM as well as cooperative map updating and mutual localization were derived.

For the local LT-SLAM, we combined two promising approaches both building upon well-known PFs. The main novelty consisted in the way uncertainty of the mapping process was handled in the LT-SLAM. Using a two-layered approach, we distinguished between verified and non-verified map information. MCL is then performed on both layers in order to update the pose belief with new sensor observations. Consequently, mapping is carried out using the MAP pose estimate and the developed map representation from Chapter 4. Within the mapping process, we additionally track the confidence of the new cell information based on localization uncertainty and thereby are able to assign it to the respective layer. Separating into two layers is not only valuable for updating the pose belief but also for deciding which cells are passed to the LT-SLAM server in the next upstream cycle as trustful map changes. For cooperative map updating, the LT-SLAM server receives the set of newly verified map cells from each agent. In a simple but effective, timestamp-based approach, the cell sets are then merged into a consistent global map. For providing agent-individual map updates to its respective agent, an agent handler per registered agent on the LT-SLAM server tracks the changes of the global map. Furthermore, we showed how mutual localization could be integrated in the C-LT-SLAM. To support a variety of different detection modules, a versatile mixture model for representing the measurement belief of the mutual detections was developed. By additionally converting the agent's particle-based pose belief into a GMM, we achieve a compact and yet accurate representation of the information thats needs to be communicated between server and agents. Update routines for both, the detected and detector robot were given also in the case of unknown associations.

# 6 Evaluation

*This chapter evaluates the developed concepts and approaches of the previous chapters. Evaluations are thereby carried out in simulation as well as in real-world experiments. We start with evaluating the environment representation and long-term map estimation, followed by benchmarking the local LT-SLAM and the C-LT-SLAM under different environmental conditions. Finally, the overall navigation system is tested to demonstrate the superior capabilities with respect to single-robot navigation systems.*

The overall goal of this chapter aims at evaluating the performance and capabilities of the developed navigation system. Due to the scope of this work, the experiments will first focus on evaluating the C-LT-SLAM with its single components and finally their integration into the overall cooperative navigation system. For more detailed evaluations of other modules than the C-LT-SLAM (e.g., cooperative global path planning), the reader may be referred to respective publications listed in Section 1.5.

## 6.1 Implementation

Proper evaluations of localization and mapping algorithms strongly depend on the availability of ground truth data, i.e., the actual pose of the robot for localization as well as precise knowledge of the actual environment for mapping which is both rarely available in most real-world scenarios, at least not with suitable accuracy. In contrast, simulation environments usually provide detailed ground truth information and additionally the flexibility to simulate arbitrary environments and different environmental conditions. On the downside, a simulation can only approximate the real-world up to a certain level. As an example and most relevant for this work, the quality of the simulated sensor observations strongly affects the significance of the experiment results for the real-world. For this reason, the strategy for evaluating the approaches of this work is to use a combination of simulated and real-world experiments and thereby being able to have both, detailed performance measures under varying conditions and scenarios as well as verifying real-world usability. Furthermore, since our approach contains several sub-approaches on different levels, we follow a bottom-up strategy by beginning with testing low-level components and afterwards their interaction

Figure 6.1: MRs of Fraunhofer IPA: rob@work 3 (left) operating in industry-like test fields and Care-O-bot 4 (right) as a retail assistance in a public electronics market.

on higher levels. More specifically, we start with investigating the long-term mapping and parameter estimation on cell level, followed by testing the local LT-SLAM as a standalone localization system (i.e., without connection to the server) and finally C-LT-SLAM which combines the latter into a cooperative system.

### Real-world experiments

Real-world experiments are carried out on real mobile robots in industrial or industry-like environments. The available mobile robots for the evaluation of this work are the rob@work 3[1] and Care-O-bot 4[2] (depicted in Figure 6.1) as well as the already presented STR (see Figure 1.1). All robots are equipped with Sick S300 safety Lidar sensors at leg height in different configurations. The close to circular-shaped Care-O-bot 4 is equipped with three S300 giving a full 360 degree field of view, same field of view is realized with two S300 on the rectangular-shaped rob@work 3, an AGV for automotive manufacturing while the STR is only equipped with a single scanner at the front resulting in a 180 degree view. From a computational hardware perspective, all robots are equipped with industrial PCs (IPCs) of different kinds where the navigation software is deployed. While the detailed specifications differ, the IPCs are all located in the mid-range price sector of nowadays com-

[1] https://www.care-o-bot.de/en/rob-work.html
[2] https://www.care-o-bot.de/en/care-o-bot-4.html

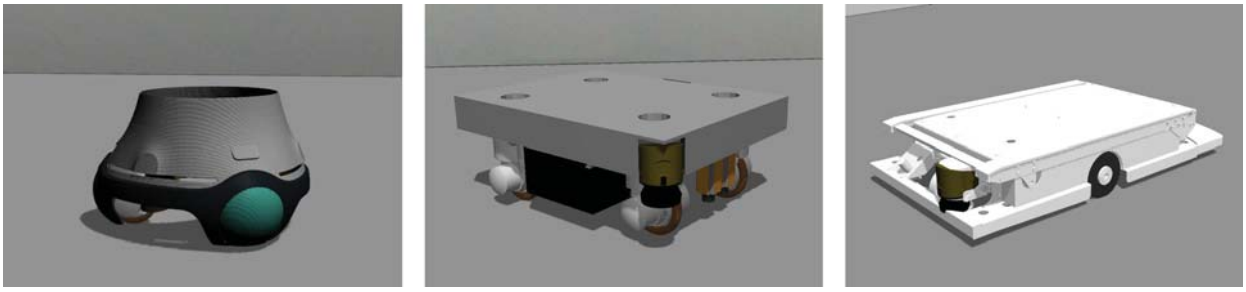


Figure 6.2: Simulated MRs in Gazebo used for simulation experiments: Base platform of Care-O-bot 4 (left), base platform of rob@work 3 (middle), STR (right).

mercial IPC. Additionally, all IPCs running on the Robot Operating System (ROS)[3] which makes them easy to run the same software modules and tools and evaluating data given by the respective on-board sensors. Looking at test environments, industry-like test fields at Fraunhofer IPA are used for real-world experiments. Additionally, an actual automotive production environment was available for testing and evaluation in terms of both, data sets gathered during experiment runs as well on-site testing on real hardware.

### Simulation experiments

For our simulation experiments, we use the robot simulation software Gazebo[4] which provides a high-quality physics engine as well as programmatic and graphical interfaces to simulate robots in user-defined indoor and outdoor environments with detailed configuration possibilities of robotic and environmental properties. Moreover, it can easily interface with ROS enabling the possibility to run the same navigation software in simulated as on real robots. For our simulation experiments, we are able to rely on implementations of precise models of the previously presented rob@work 3, Care-O-bot 4 and STR incorporating among others their chassis and sensors, see Figure 6.2. Additionally, we implemented a logistics environment with parameterizable conditions in terms of its dynamics. Besides, we use a self-implemented grid cell simulator which we use for detailed experiments on cell level. This cell simulator has the advantage of easily setting up as well as controlling possible cell configurations and evolutions in addition with simulating sensor observations over long-time periods in order to examine the long-term behavior and numerical stability of the mapping algorithms.

### Implementation aspects

The presented approaches and algorithms have been fully implemented in C++ and using ROS which thereby serves as the software framework including numerous libraries and tools

[3] http://wiki.ros.org/
[4] http://gazebosim.org/

needed for robotic software development. The most basic components of a ROS system consist in the ROS master and one or more ROS nodes. The master is the central coordinating instance for the processes and communication within the ROS network. Single computation units are called nodes within the ROS system. Most robotic software systems consist of numerous nodes. The communication among the nodes takes place through publisher and subscriber of so called topics realizing a many-to-many, one-way communication or through services realizing one-to-one, two-way communication. Nodes can be spatially distributed on different hardware throughout the network making multi-robot systems in principle applicable with ROS. However, a node (temporarily) losing connection to the ROS master will immediately crash. This circumstance makes the usage of wireless networking among the nodes nearly impractical. Since wireless communication is a central property needed for mobile multi-robot systems, the usage of ROS for these systems is not straightforward. A way out are so called ROS multi-master systems which enable the communication among different ROS masters. In that way, each robot (or another network component) runs its own master managing the local nodes whereas the communication among the robots is managed by the multi-master system. For implementing our cooperative navigation system, we use the ROS multi-master system rocon[5] which exhibits suitable properties in terms of handling network disruptions and ease of configuration.

## 6.2 Experiments

### 6.2.1 Long-Term Map Estimation

The developed environment representation and map estimation of Chapter 4 contains on top level a two layered representation with a metric and roadmap/object layer. Since the basic concept of this dual layered approach has already been evaluated successfully in different related works, we focus on the new approach of the metric layer in terms of the xGMM-DOGM. The goal of this experiment aims at testing the approach for long-term mapping given different dynamics of the environment as the two key requirements for our target application. Furthermore, we benchmark the approach against the related approaches of NDT-OGMs and NDT-DOGMs (as described in Chapter 4) to demonstrate the different capabilities.

For being able to reconstruct different scenarios with cell level of detail and perform long-term simulations, we use the cell simulator described in Section 6.1. The experiment setup is depicted in Figure 6.3. In the first scenario of the experiment, a wall with different dynamic configurations is simulated. During the experiment, the wall changes its position to one of the three configurations with a given frequency. By altering this frequency in different

[5] http://wiki.ros.org/rocon

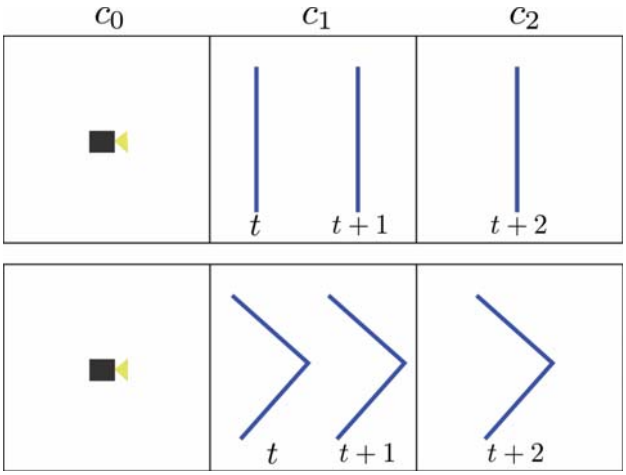



Figure 6.3: Setup of map estimation experiment. Wall (top) and corner (bottom) scenario of a simple environment modeled with three grid cells. The robot and its sensor are located in cell $c_0$ (left) and observes cell $c_1$ (middle) as well as cell $c_2$ (right) in different, time-depending configurations of the object in the cell.

tests, a semi-static wall with different changing rates can be simulated. In the second scenario, same is carried out for a corner-shaped object instead of the wall. Depending on the configuration of the change frequency, the following tests are carried out:

- Static wall/corner: no position change of object during experiment
- Low-frequently changing wall/corner: low position change frequency
- High-frequently changing wall/corner: high position change frequency

During the experiment, scans are generated by the cell simulator. This is carried out by raytracing a set of beams from the sensor position through the cells until an object is hit. Subsequently, noise is added to the computed distance to the object. We apply Gaussian noise (with zero mean and 0.03 $m$ standard deviation). Additionally, we simulate measurement outliers resulting in real world, e.g., from glass surfaces by sampling from a uniform distribution within some distance around the object. To improve comparability among the approaches as well as to demonstrate the effects of the different geometric descriptions used by the respective approach to approximate the object contour, no high-reflective objects are simulated within these experiments. Mapping and localization including reflectivity data will be tested in a separate experiment, see Section 6.2.2.

To evaluate the performance, we need to consider the actual usage of the map within our navigation system in terms of localization and path planning. While in principal both rely on accurate maps, an appropriate quality measure differs for the two components.

For path planning, geometrical accuracy of the map is most important. More specifically, accuracy is rated in terms of correctly classifying free and occupied space. To measure this accuracy, we transform the estimated LR xGMM-DOGM map into an HR OGM as

described in Section 4.5 and compare the HR cells contained by each LR cell to the respective ground truth cells. The ground truth is generated by applying the same HR grid and declare each cell which is (partly) covered by the known ground truth object as occupied and the remaining cells as free. For each LR cell, we can then compute the well-known F-measure by comparing the corresponding HR grid cells. The F-measure is defined as:

$$F = 2 \, \frac{precision \cdot recall}{precision + recall} \tag{6.1}$$

where:

$$precision = \frac{TP}{TP + FP} \tag{6.2}$$

$$recall = \frac{TP}{TP + FN}. \tag{6.3}$$

In this context, a TP is a correctly declared occupied cell, an FP is a free cell that is declared as occupied and a false negative (FN) is an occupied cell that is declared as free.

In contrast, to measure the map quality for localization, we evaluate the likelihood of the scan observations with respect to the estimated map (in the following referred to as localization likelihood) by using the observation likelihood model given in Section 5.2 and the known ground truth pose of the robot. Since the actual pose of the robot should generate the highest observation likelihood, this measure implies a direct feedback on the map quality with respect to localization. However, a suitable map does not only need to indicate high likelihoods for high accurate pose estimates but also low likelihoods for inaccurate poses. Therefore, we additionally evaluate simulated scan observations from a drifted robot pose, i.e., a pose within a close radius around the ground truth pose. This measure is referred to as drifted localization likelihood.

To test the long-term mapping capability, we simulate $10^8$ scan points. Assuming that a single observation of the cell contains 50 scan points, the sensor outputs observations with 20 Hz and the agent observes the specific cell for 1 hour accumulated over the day, this simulates a whole year of operation which should be sufficient to test long-term stability of the given approach.

Table 6.1 summarizes the results of the tests from the wall scenario. Within these tests, we use two different configurations of the NDT-OGM. As discussed in Subsection 4.1.2, this approach relies on a predefined parameter that controls the adaption rate of the NDT to recent observations. Setting a value close to 1 realizes a low adaption (referred to as NDT-OGM low) suitable for slowly changing areas whereas decreasing the value increases the adaption rate suitable for highly changing areas (referred to as NDT-OGM high). Since the dynamics of each area is usually a priori unknown, our NDT-DOGM and also xGMM-DOGM perform an online estimation of this parameter. We can clearly observe the effect of

this parameter when looking at the results of the *low/high-frequently changing* test cases. As expected, in the *static* and *low-frequency* test cases, the low adaptive NDT-OGM performs better than its high adaptive counterpart whereas in the *high-frequency* test cases, this behavior is reversed. The NDT-DOGM is able to perform in all three test cases comparable to the respective NDT-OGM configured suitable adaption rate for the specific test case. This demonstrates the superior ability of online estimating the adaption rate with respect to a predefined, fixed rate.

A wall can be well approximated with a single NDT which should give the GMM no additional edge compared to the NDT approaches in these test cases. However, the NDT-xGMM still significantly outperforms the NDT approaches in every single wall test case. If we have a closer look at the reason, we discover an additional advantage of the xGMM-NDT that has not been considered much so far which is the ability of filtering out outliers. If an outlier occurs, the xGMM-DOGM creates a new component for this outlier which usually disappears after few successive observations since it is not supported by them. The already existing component representing the actual line remains mostly unaffected and thereby keeps up its accuracy of the real object. In contrast, NDT approaches incorporate each observation and thereby also the outliers when updating the NDT parameters leading to bulky NDT ellipses with low accuracy. This effect is also discoverable if we have a look at the localization likelihoods of the NDT approaches. While the actual localization likelihood is comparable good (in some cases even exceed the xGMM-NDT score), the drifted localization likelihoods are also quite high. This also results from bulky ellipses and impairs the possibility to infer an accurate pose estimate from observations. We found an extreme case of this when looking at the NDT-DOGM high in the *static* and *staticlow-frequently changing wall* test cases where the drifted localization likelihood is even higher than the actual localization likelihood.

The motivation of the *corner* test case is to further investigate the influences of non-linear object contours on the mapping accuracy. As can be taken from the results in Table 6.2, the xGMM-DOGM approach can further increase its superior position in the test cases of this scenario. This further proofs the effectiveness of using more than one NDT for approximating non-linear objects' contours. Besides this, the test cases of this scenario further emphasize the described observations of the *wall* test cases.

While the so far presented results already indicated superior capabilities of the xGMM-DOGM, we only considered mean values of the whole test case. For proving a suitable behavior for long-term mapping, we also have to proof that the approach is neither degenerating over time nor temporarily failing. Therefore, in Figure 6.4, we find the accuracy measures plotted over the number of observations of the entire *high-frequently changing corner* test case. To improve the readability of the figure, we still averaged over sequences of 50 observations. We clearly see that all quality measures remain constant over time supporting its long-term stability.

Table 6.1: Results from *wall* test cases

| **Map type** | **Mean F-measure** | **Mean Loc. likelihood** | **Mean drifted Loc. likelihood** |
|---|---|---|---|
| | **Static wall** | | |
| NDT-OGM low | 0.481 | 0.603 | 0.096 |
| NDT-OGM high | 0.144 | 0.660 | 0.652 |
| NDT-DOGM | 0.481 | 0.603 | 0.096 |
| xGMM-DOGM | 0.934 | 0.543 | 0.091 |
| | **Low-frequently changing wall** | | |
| NDT-OGM low | 0.569 | 0.553 | 0.245 |
| NDT-OGM high | 0.410 | 0.476 | 0.551 |
| NDT-DOGM | 0.569 | 0.552 | 0.244 |
| xGMM-DOGM | 0.790 | 0.527 | 0.064 |
| | **High-frequently changing wall** | | |
| NDT-OGM low | 0.437 | 0.492 | 0.420 |
| NDT-OGM high | 0.490 | 0.642 | 0.664 |
| NDT-DOGM | 0.477 | 0.613 | 0.476 |
| xGMM-DOGM | 0.661 | 0.558 | 0.120 |

Table 6.2: Results from *corner* test cases

| **Map type** | **Mean F-measure** | **Mean Loc. likelihood** | **Mean drifted Loc. likelihood** |
|---|---|---|---|
| | **Static corner** | | |
| NDT-OGM low | 0.385 | 0.473 | 0.252 |
| NDT-OGM high | 0.218 | 0.563 | 0.379 |
| NDT-DOGM | 0.384 | 0.472 | 0.091 |
| xGMM-DOGM | 0.934 | 0.544 | 0.094 |
| | **Low-frequently changing corner** | | |
| NDT-OGM low | 0.542 | 0.448 | 0.244 |
| NDT-OGM high | 0.461 | 0.553 | 0.467 |
| NDT-DOGM | 0.542 | 0.448 | 0.246 |
| xGMM-DOGM | 0.688 | 0.536 | 0.138 |
| | **High-frequently changing corner** | | |
| NDT-OGM low | 0.467 | 0.456 | 0.335 |
| NDT-OGM high | 0.547 | 0.608 | 0.452 |
| NDT-DOGM | 0.518 | 0.506 | 0.316 |
| xGMM-DOGM | 0.698 | 0.526 | 0.087 |

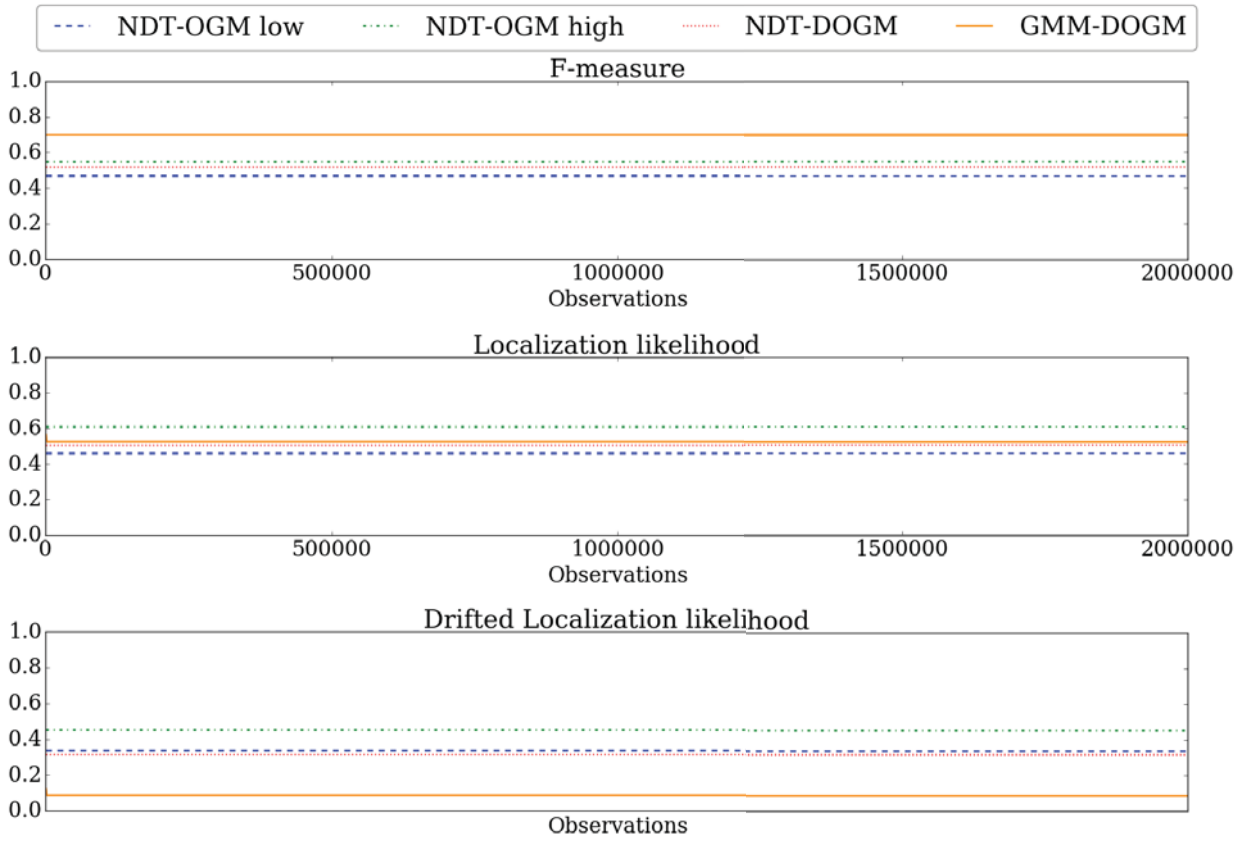



Figure 6.4: F-measure, localization likelihood and drifted localization likelihood over number of processed observations from *high-frequently changing corner* test case. To improve readability of the plot figure, clutter is removed by plotting mean values of sequences with 1000 observations each.

To sum up, the map representation and map estimation experiment demonstrated the suitable properties of our xGMM-DOGM approach in terms of mapping accuracy for both path planning and localization in long-term mapping scenarios. Moreover, it clearly outperforms the NDT approaches. This comes at cost of increased computational demands which appeared to be in the range of factor 2-5 compared to the NDT approaches depending on the actual test case. However, we will demonstrate in following experiments that by using xGMM-DOGM, we can further increase grid cell sizes which compensates for the increased computation times on cell level.

### 6.2.2 Local Long-Term SLAM

The local LT-SLAM is the core component of the C-LT-SLAM. Its task is to provide accurate, robust and high-frequency localization information, also under difficult environmental conditions as well as in the (temporary) absence of connectivity to the cloud-server. Without a local LT-SLAM fulfilling these requirements, the C-LT-SLAM is not able to work. Due to this circumstance, extensive evaluations of the local LT-SLAM described in

Section 5.2 need to be carried out. Furthermore, the aim of these experiments is to demonstrate the capabilities but also the limitations of the LT-SLAM in the case of not being connected to the server. For realizing detailed evaluations of the robustness and accuracy of the local LT-SLAM when exposed to environments with varying dynamic properties. The first experiment is carried out in a simulated warehouse environment. Afterwards, the results are verified in a real-world experiment with test data gathered in an automotive production environment.

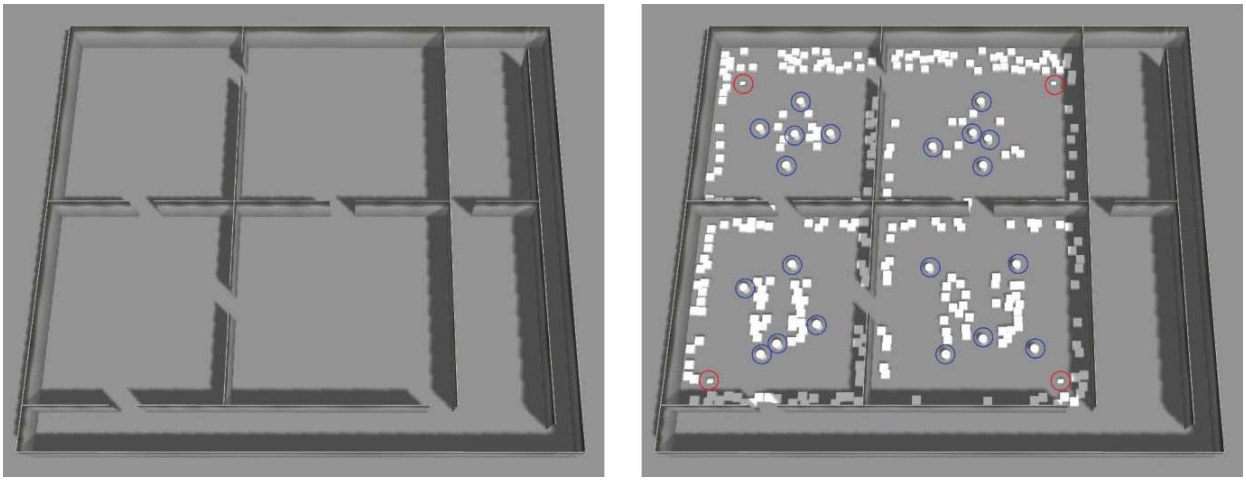


Figure 6.5: Simulated warehouse used for localization benchmarks. Left: Base setup only containing static walls, right: populated setup with four robots (highlighted with red circles), further dynamic objects (pillars high-lighted with blue circles) and semi-static boxes.

### Warehouse Simulation

The following experiment aims at simulating different environmental conditions using the previously described robot simulation environment in order to provide a detailed measure of the influence of these conditions on the localization performance of the local LT-SLAM. Therefore, a warehouse environment is modeled, see Figure 6.5 for a visualization of the simulated environment. The warehouse consists of four major rooms partly surrounded by corridors covering a total area of 2500 $m^2$. To set up different scenarios, it can be populated with an arbitrary number of boxes and pillars where each can be configured as static (not moving during the experiment), semi-static (changing the position with a specified frequency) or dynamic (continuously moving with specified velocity) opening up the possibility to emulate various types of dynamics that may occur in a warehouse. Obviously, testing all configurations including different number of static, semi-static and dynamic objects, realistic changing frequencies of each semi-static object as well as velocity of each dynamic object is impractical. Instead, we reduce configuration variants by using the following configuration parameters:

- Object ratio: Proportion of the number of semi-static objects with respect to the sum of semi-static and (non-occluded) static objects. Neglecting minor fluctuations, this ratio is constant throughout all areas of the warehouse.
- Change frequency: Frequency of pose changes of semi-static objects.
- Dynamic Objects: Presence of dynamic objects (apart from the robot fleet). If enabled, each room is populated with five dynamic objects.

Based on these parameters, different test cases are generated (see Table 6.3) simulating typical environmental conditions. In all test cases, four mobile agents of type rob@work 3

Table 6.3: Environment configuration for local LT-SLAM test cases.

| **Test case** | **Object Ratio** | **Changing Frequency** | **Dynamic Objects** |
|---|---|---|---|
| Fully static | 0.0 | - | none |
| Static and dynamic objects | 0.0 | - | present |
| Mixed static, semi-static and dynamic | 0.5 | low | present |
| Fully non-static, slowly changing | 1.0 | low | present |
| Fully non-static, moderately changing | 1.0 | medium | present |
| Fully non-static, fast changing | 1.0 | high | present |

operate within the environment, one in each of the four rooms. Every ten minutes, the agents switch the room. Although this experiment aims at testing the local LT-SLAM without any cooperation, we record data from all four robots. This enables the possibility to evaluate the performance of the cooperative system (see Subsection 6.2.3) on the same data set and get a direct feedback on its effectiveness compared to the non-cooperative set-up. Each data set contains the sensor data of several hours of operation in the specific environment configuration. Obviously, this duration cannot be declared as a long-term operation. However, as described in Section 1.2, the major challenge for long-term localization systems is to handle the changes of the environment without a degeneration of the localization performance. Since the test duration is sufficient to expose the robots to typical changes of the environment, successfully handling those changes is the major proof for long-term stability.

As accuracy measure, we evaluate the mean position error (MPE) as the position error of the localization estimate averaged over the test duration. Additionally, the standard deviation of the MPE is computed. For rating the localization robustness, a second measure accounts for localization failures during the test. Thereby, a localization failure is defined by a position or orientation error of the localization estimate exceeding a certain threshold from which it cannot recover. In real-world applications, these failures usually lead to a standstill of the mobile robot and require human intervention. Hence, these kinds of failures

drastically reduce the autonomy of the robot and are thereby more severe compared to a constant but minor localization inaccuracy. In our tests, we choose the thresholds of 1.25 m for position error and 45 degree for orientation error and a period of five seconds given for recovery. If the robot does not succeed to recover from the error within this time period, the failure counter is incremented and the robot's pose estimate is reset with the ground truth pose to be able to continue the test.

In order to assess the resulting performance of the local LT-SLAM developed within this work (referred to as MCL-Dual-Conf-xGMM-DOGM), we benchmark its results against relevant state-of-the-art approaches as given in Subsection 5.1.2. More specifically, we additionally test implementations of an MCL-NDT-OGM comparable to Saarinen, Andreasson et al. (2013b), of an MCL on dual timescale NDT-OGMs (referred to as MCL-Dual-TS-NDT-OGM) as presented in (Valencia, Saarinen et al. 2014) and of an RBPF-DOGM similar to the approach presented in (Tipaldi, Meyer-Delius et al. 2013). In order to make each approach using the exact same input information, we do not simulate any high reflective objects in the rooms' areas which could be processed solely by our MCL-Dual-Conf-xGMM-DOGM. Instead, the processing of reflectivity observations will be evaluated in a separate experiment followed by this experiment. Additionally, comparability is provided by giving each approach the same initial map based on the initial state of the environment in form of a OGM which is constructed from ground truth data prior to the test runs. As the computational hardware where the implemented approaches are executed, a Intel Xeon dual core CPU @ 3.3 GHz is used.

For setting the baseline, we start with the evaluation of the *fully static* test case where we primarily aim at investigating the dependency of the localization performance on different grid cell sizes of the underlying grid map of the different approaches. Results are given in Figure 6.6 where we find the highest evaluated resolution with 0.1 m/cell. Higher resolutions than this are neglected since they did not produce reasonable results on the tested hardware due to their extensive computational demands. As expected, we see in Figure 6.6 that the RBPF-DOGM as the only approach without holding a geometric model of the object contour within the cell, is highly dependent on high grid resolutions. The remaining approaches can drastically lower their dependency on the grid resolution. From 0.25 m to 1.0 m grid cell sizes, the resolution has only marginal impacts on their localization accuracies. With even lower resolutions, the accuracies slowly decreases.

As expected, in static environments, a LT-SLAM has no advantage compared to localization approaches relying on a static map like the MCL-NDT-OGM which achieves the best results when looking at relevant grid resolutions. In contrast, the mapping step of the LT-SLAM approaches induces some noise on the map estimate which may lead to inaccuracies of the localization. Especially for low grid sizes, this effect becomes visible. However, with its ability to estimate the cell dynamics and to approximate non-linear structures,

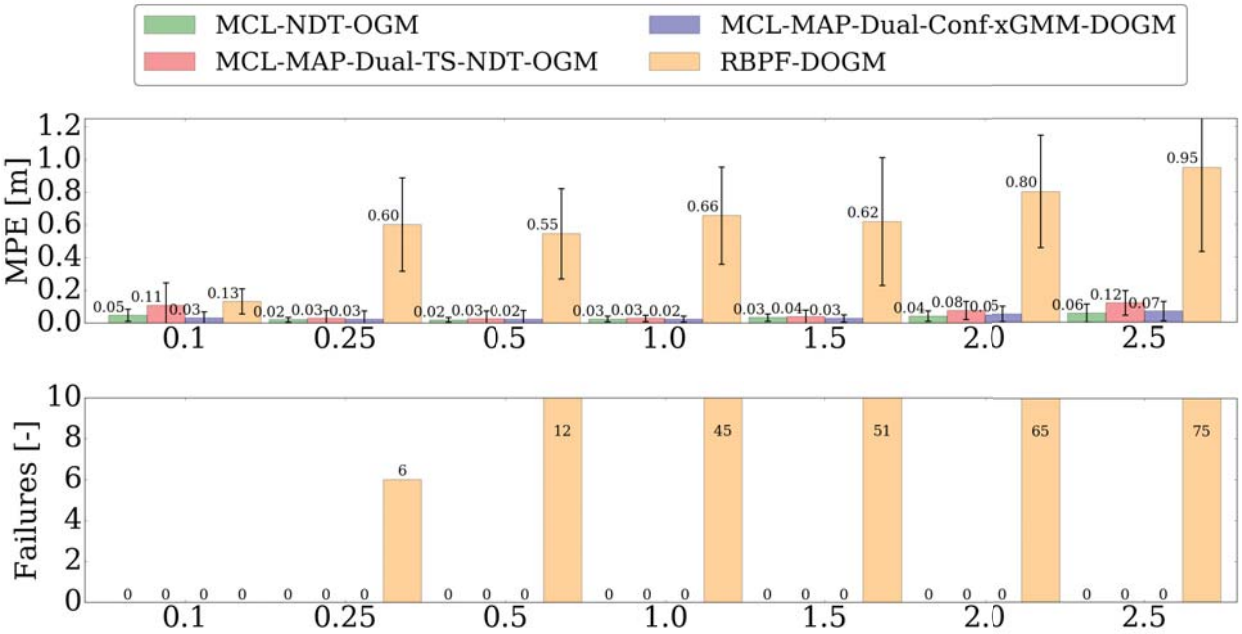



Figure 6.6: Overview of results for *fully static* test case: MPE with standard deviation (top), number of localization failures (bottom) over different cell sizes of the underlying grid map.

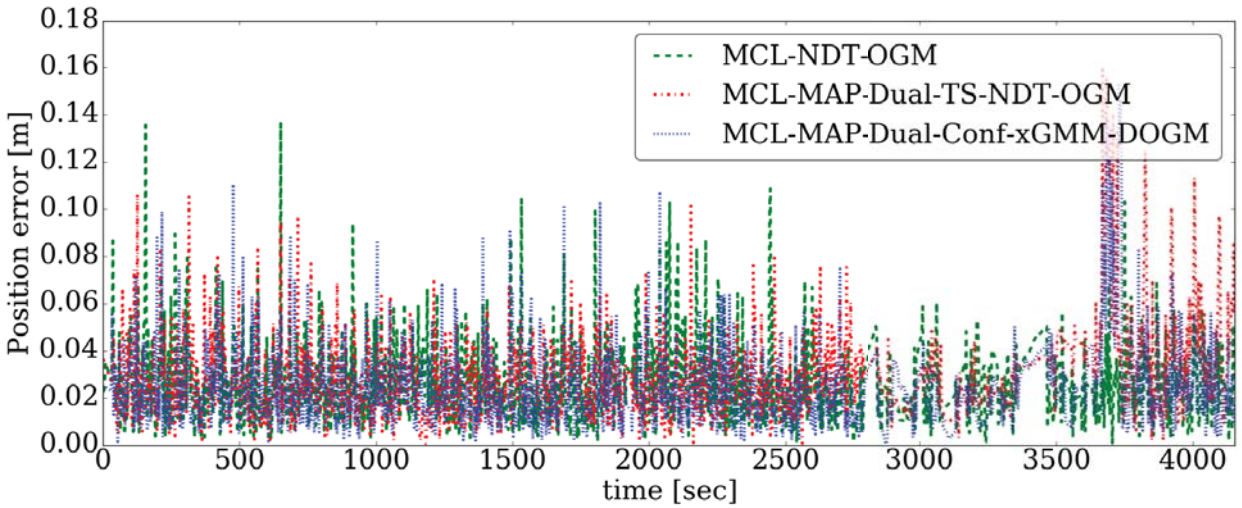



Figure 6.7: Position error for grid size of 1.0 m over time in *fully static* test case.

our MCL-Dual-Conf-xGMM-DOGM can keep this noise at a marginal level and thereby outperforms the MCL-NDT-OGM.

Figure 6.7 depicts the position errors over time when using a grid size of 1.0 m. The RBPF-DOGM is excluded from this figure to focus on the remaining approaches which succeed this test run without failures. Although we can clearly identify noise on all localization estimates, none of the approaches exhibits major localization errors or drifts which would indicate problems with respect to its long-term stability.

The key insights from the *fully static* test case are as follows. The RBPF-DOGM does not present a usable approach for our application and is therefore neglected for the further test cases to focus on the remaining ones that successfully handled the test case. Furthermore, we saw that the influence of the grid resolution on the localization performance is negligible up to a resolution of 1.0 m and subsequently increases slowly. Therefore, instead of testing the full range of grid cell sizes from 0.1 m–2.5 m, we concentrate on the particular cell sizes of 1.0 m and 2.0 m to demonstrate resulting localization accuracies from different cell sizes. Following this, Figure 6.8 depicts the results for all test cases of Table 6.3.

Starting with the results of the *static and dynamic* test case, we see that our tests confirm the result of (Saarinen, Andreasson et al. 2013b) in the way that adding few dynamic objects to a otherwise static environment does not have any significant influence on the resulting localization performance of the MCL-NDT-OGM. Same holds for the LT-SLAM approaches which achieve comparable results as in the *fully static* test case.

The *mixed static, semi-static and dynamic* now represents the first test case where the localization approaches are exposed to semi-static objects. However, with an object-ratio of 0.5, this test case additionally exhibits many static objects, i.e., areas where the map does not change. For a grid resolution of 1.0 m, all three approaches do not exhibit any localization failures but a slightly increased MPE. In contrast, the results using a grid size of 2.0 m are completely different. In this configuration, the MCL-NDT-OGM commits 15 localization failures. This drastic difference can be explained by the fact that with this high cell sizes, only few cells are occupied with only static objects while the major part of cells rather contain a combination of static and non-static objects. The resulting NDT observations now poorly matches to the initial map. The MCL-Dual-TS-NDT-OGM can partly overcome this problem by mapping those inconsistencies in its short-term NDT-OGM and thereby is able to avoid localization failures. However, its localization accuracy significantly decreases due to the reason that a single NDT poorly approximates several objects within the cell. By using a GMM, the MCL-Dual-Conf-xGMM-DOGM is largely unsusceptible to this issue and thereby only exhibits a minor decrease of accuracy.

The following *fully non-static* test cases intensify this issue due to the complete absence of static objects leading to completely altered environments after some time period depending on the applied change frequency. These test cases clearly demonstrate the necessity of an LT-SLAM for these kind of environments. Both tested LT-SLAM approaches can handle the *fully non-static, slowly changing* with only marginal differences in the resulting accuracy compared to the previously discussed *mixed static, semi-static and dynamic* test case. This demonstrates the effectiveness of the LT-SLAM approaches which are able to keep the MPE at a low level despite the fact that after some time the initial map does not contain any valid object position. However, with increasing changing frequency, the localization performance of the LT-SLAM approaches clearly decreases. While in the *moderately changing* test case,

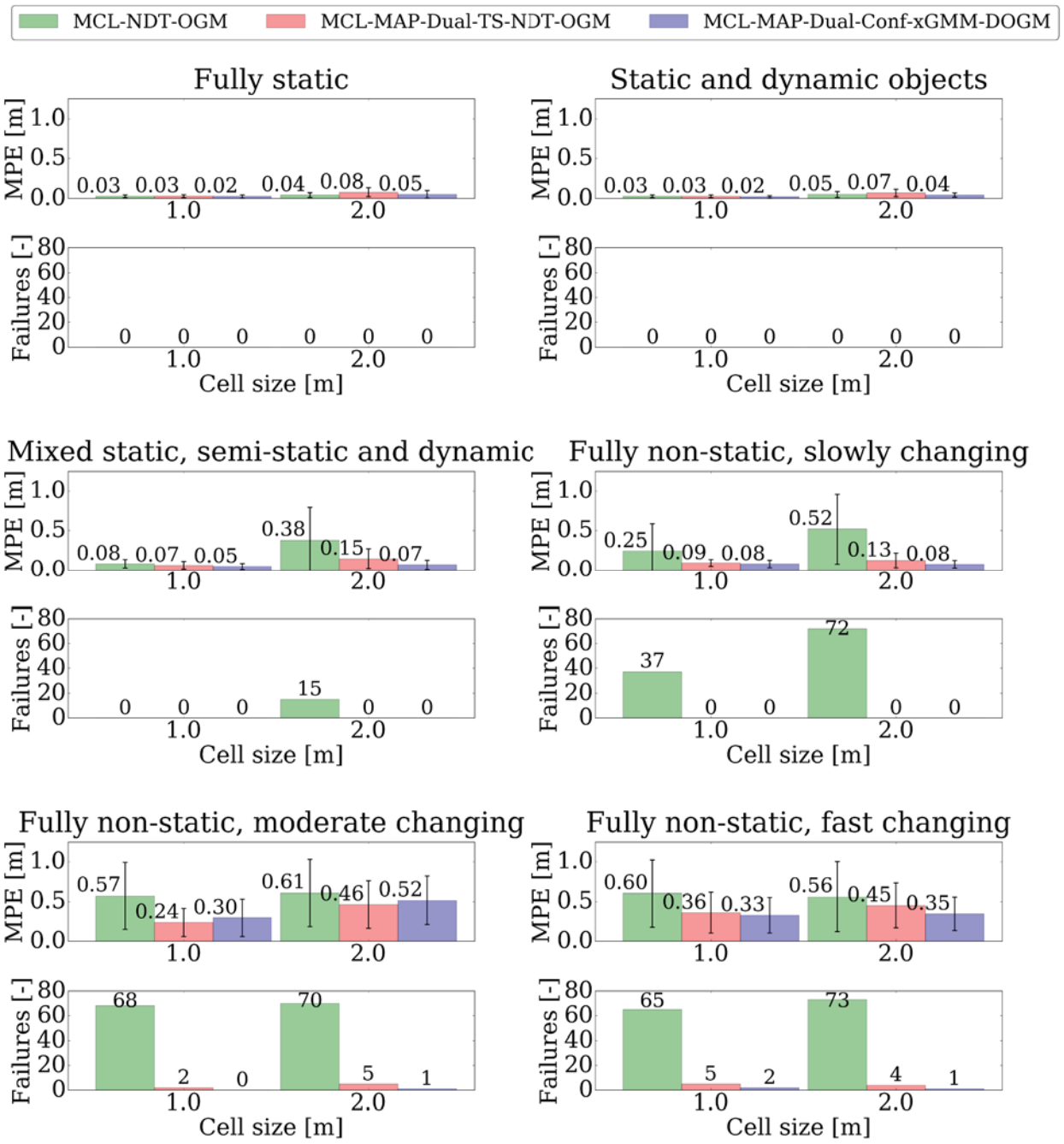



Figure 6.8: Overview of results for different test cases: MPE with standard deviation (top), number of localization failures (bottom) for different cell sizes of the underlying grid map.

our MCL-Dual-Conf-xGMM-DOGM (with 1.0 m grid resolution) is still able to finish the test without a failure, none of the approaches is able to handle the *fast changing* test case. This can be clearly traced back to the high degree of non-observed changes by the single MR and, as a result, being exposed to highly diverged areas when traveling through the environment. As discussed in Section 1.2, from the perspective of a single MR, those highly diverged areas can be either the result of high frequency changing rates of the semi-static

objects or low revisiting frequencies by the MR making them not a rarity in industrial environments.

To sum up, we demonstrated the necessity, potential but also limitations of LT-SLAM approaches when exposed to different dynamics of the environment. Our MCL-Dual-Conf-xGMM-DOGM clearly outperformed other benchmarked approaches. However, its performance is also limited due to operating on local sensor data and thereby it cannot overcome the problems of limited environment knowledge in strongly changing environments.

#### Reflectivity processing experiment

In the previous experiments, we omitted reflectivity data in order to focus on pure geometric maps used for localization and SLAM. As demonstrated in the previous experiment, a pure geometric description of the environment is sufficient in environments with dense geometric structure like in the room areas of the simulated warehouse. However, as described in Section 5.2, some environments only exhibit sparse or symmetric geometrical structures. As an extreme example, we found the described corridor area of Section 4.4 where a sole geometric approach cannot infer any information about the position of the robot alongside the corridor. Instead, it will be exposed to the growing error of accumulated odometry information when traveling along the corridor. This circumstance motivated the extension from the pure geometric describing GMM to xGMMs in order to make usage of reflectivity measurements in the SLAM process. We now want to evaluate the effectiveness of our approach by measuring the position error when traveling along the right corridor of the simulated warehouse (see Figure 6.5). The corridor is equipped with 0.05 m-thick retro-reflective markers on both sides of the corridor with distances of 4 m to one another. The marker positions are a priori unknown to the LT-SLAM. Like in the previous experiment, the LT-SLAM is given an initial OGM of the environment. For realizing a direct comparison, we test our MCL-Dual-Conf-xGMM-DOGM with the reflectivity processing enabled and disabled. Figure 6.9 depicts the results of this experiment in terms of the resulting position error over the time traveling within the corridor area. As expected, with disabled reflectivity processing unit, the error continuously grows leading to a final error of about 4.0 m which results from a mean odometry error of about 0.01 m per traveled meter which is a common value for nowadays mobile robots' odometry quality. In contrast, with the usage of reflectivity measurements, the error remains stable throughout the test runs with an overall MPE of 0.19 m.

#### Real-world automotive production environment

The final experiment of the local LT-SLAM aims at verifying the applicability of the developed LT-SLAM approach in real-world. For this purpose, we use a data set of a eight-

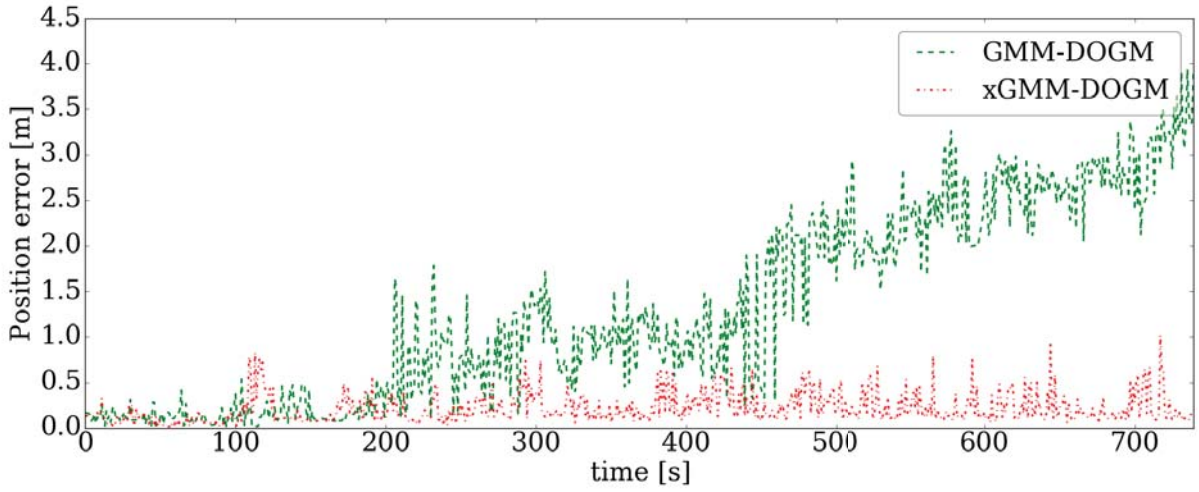



Figure 6.9: Position error over time when traveling in corridor area with reflection measurement processing enabled (xGMM-DOGM) and disabled (GMM-DOGM).

week operation of the previously presented STR operating in the automotive production scene. Due to the absence of ground truth data, we cannot evaluate the localization performance like we did in the previous experiments. Instead, we evaluate the convergence of the map by comparing the initially given map with the resulting map after finishing the test set. A proper convergence, i.e., minor modifications of static elements, is only possible if no localization drifts or failures occur. Hence, it gives us an indirect feedback on the localization performance. Figure 6.10 shows the results of this map comparison. Here, we find the initial map drawn as a standard OGM by using the map conversion described in Section 4.5 with black and white pixels representing occupied and free cells respectively. Additionally, the difference between the initial and the final map is drawn as blue and red pixels with different opacities indicating the amplitude of the difference. Blue pixels depict areas that have changed their states from occupied (or unknown) to free whereas red pixels changed from free to occupied cells. We see that the area in which the STR was traveling exhibits only few static elements in terms of the surrounding walls and a sparse grid of steel girders. These are only slightly altered in the final map giving no indication for localization drifts. The rest of the area consists of rather slowly changing semi-static objects. Thereby, this environment represents the *fully non-static, slowly changing* test case of the warehouse simulation experiment, only that the density of semi-static objects is significantly sparser. Although we are not able to rate the correctness of each changed pixel, the final map does not contain any artifacts that indicate a misbehavior. This rather qualitative evaluation is supported by extensive, several-day test runs where the LT-SLAM was applied as the localization system within a (non-cooperative) navigation system and produced no observable localization errors or failures.

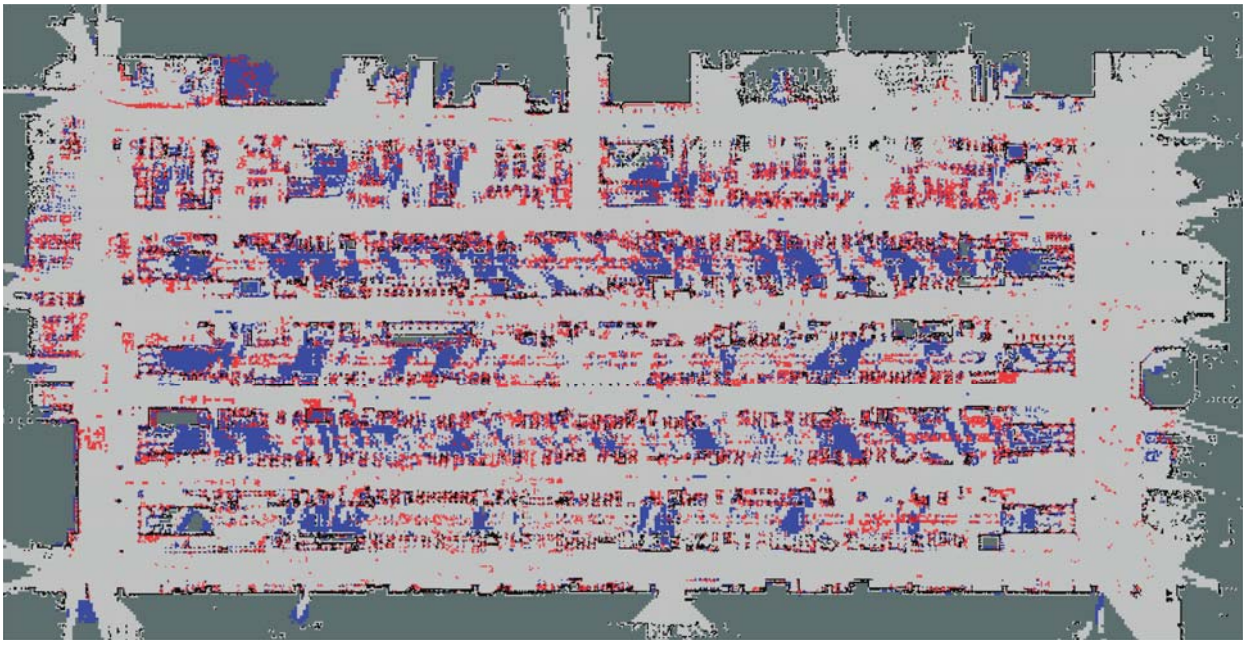


Figure 6.10: Comparison of initial and final map after four-month operation in an automotive production scene. Black pixels depict unchanged occupied cells from initial map, white pixels unchanged free cells and grey cells unchanged unknown cells. Blue pixels represent cells that have changed from occupied (or unknown) to free whereas red pixels from free to occupied cells.

### 6.2.3 Cooperative Map Updating

In Section 6.2.2, we demonstrated that the local LT-SLAM used as a stand-alone localization module without server connection achieved suitable results when applied to environments with a limited degree of environmental changes. In environments exhibiting higher degrees of changes, the localization performance declines. While in the *fully non-static, slowly changing* test case, a marginal decrease of localization accuracy was observable, the decrease got significant in the respective *moderately changing* test case. Finally, in the *fast changing* test case, it could not keep up with the changes of the environment resulting in several localization failures. The latter two test cases are therefore of main interest to evaluate the effectiveness when enabling the agents to share their detected map changes and leverage map updates provided by the LT-SLAM server. Since the data sets used for the benchmarks of Section 6.2.2 already contain data from all four simulated robots, we can reuse them to run the respective test cases another time but enable cooperative map updating functionality and thereby achieve a direct measure on the influence of the server-based map updates on the localization performance of the robots.

Using the ROS framework with the multi-master extension, we set up the C-LT-SLAM nodes, i.e., the LT-SLAM server node on the cloud server and an LT-SLAM node for each robot running on a separate computational unit all being fed with their recorded data of the specific test case. The initialization of the C-LT-SLAM module is carried out as follows.

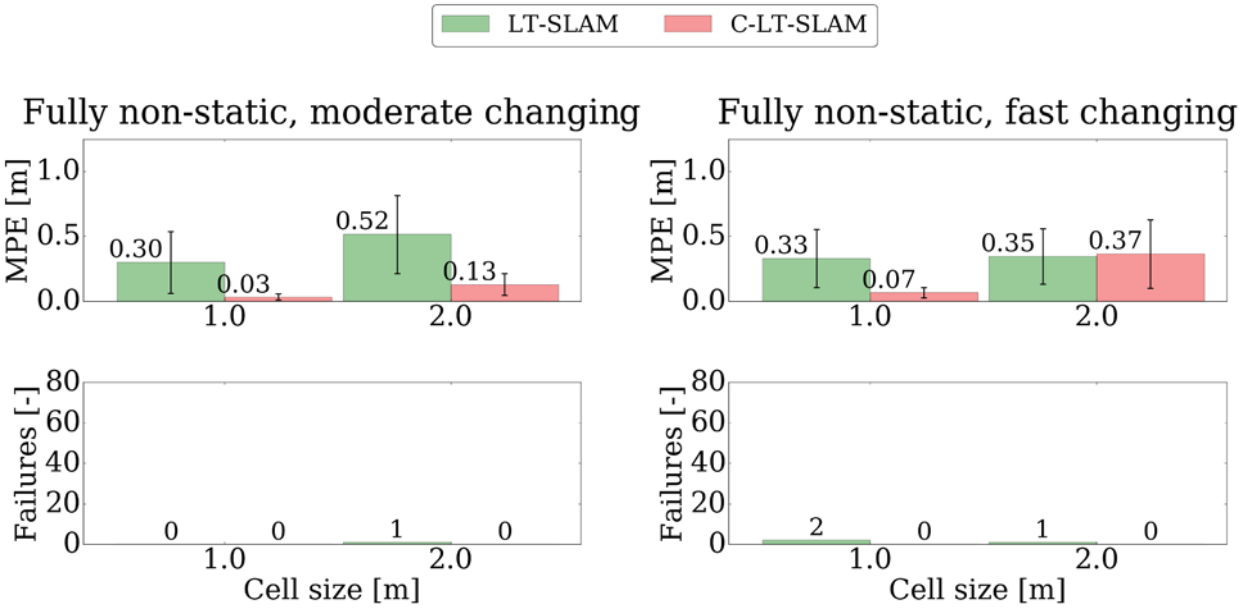


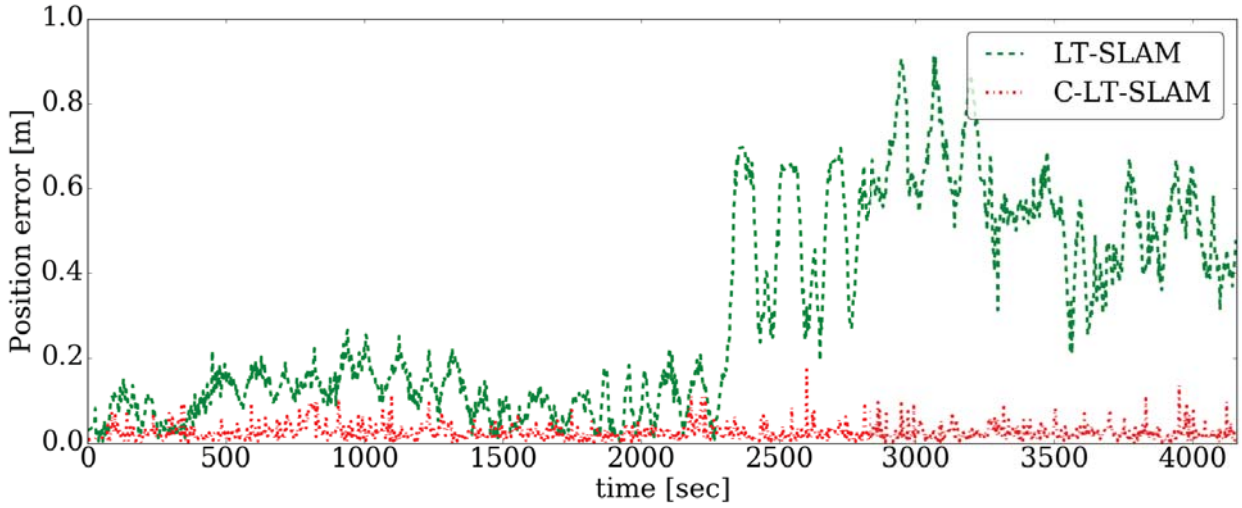

Figure 6.11: Overview of results of cooperative map updating for *fully non-static, moderately changing* and *fully-static, fast changing* test cases: MPE with standard deviation (top), number of localization failures (bottom) for different cell sizes of the underlying grid map.

At start up, each robot registers at the LT-SLAM server and receives the initial map which, same as in the local LT-SLAM tests, represents the initial state of the environment. After initialization, the periodic map upstream and map downstream process is initiated. For our tests, we choose a period of 10 seconds for both cycles as an appropriate value to compromise sufficient update rates with communication loads.

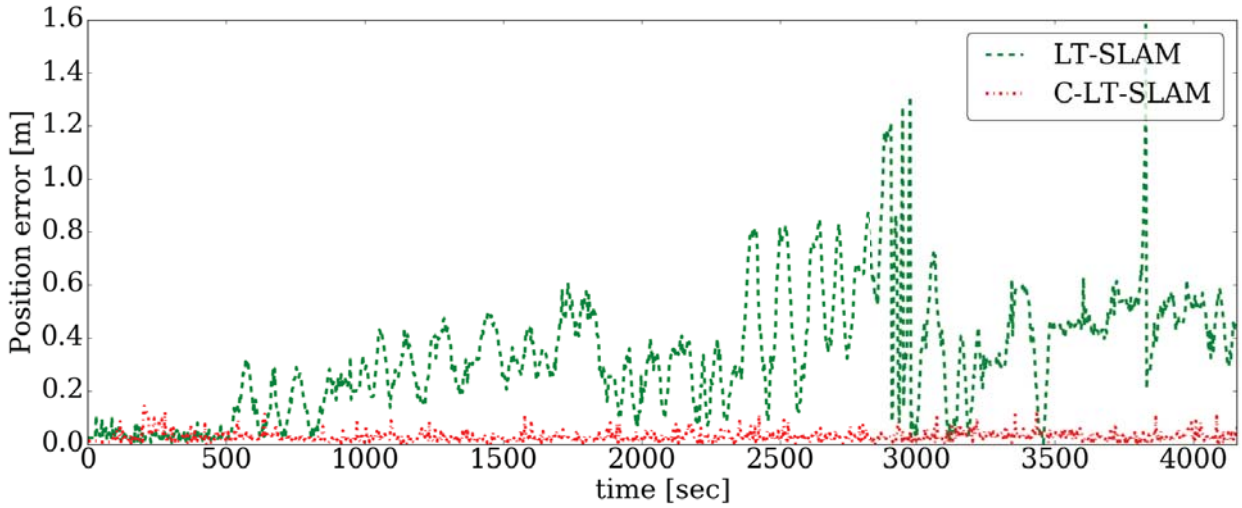
### Influence on Localization Performance

The localization performance is measured in the same way like for benchmarking the local LT-SLAM approaches in Section 6.2.2. The results are given in Figure 6.11 where C-LT-SLAM refers to our cooperative system using the MCL-MAP-Dual-Conf-xGMM-DOGM as the local LT-SLAM approach whereas LT-SLAM just refers to the non-cooperative MCL-MAP-Dual-Conf-xGMM-DOGM. These results have already been presented in Section 6.2.2 and are only recited here again for better comparison. From Figure 6.11, we see that when using the C-LT-SLAM in the *fully non-static, moderately changing* test case, the localization error can drastically be decreased achieving accuracies close to that of the LT-SLAM on the *fully static* test case (see Section 6.2.2). Figure 6.12a additionally depicts the position error over time for that specific test case demonstrating how the C-LT-SLAM is able to keep the MPE constantly low while the MPE of the non-cooperative LT-SLAM slowly diverges. Same holds for the *fast changing* test case where the C-LT-SLAM is the only tested approach that is able to handle the dynamics without any localization failure and only a slight decrease of

the localization accuracy. Figure 6.12b additionally shows the position error over time from this test case where the C-LT-SLAM again demonstrates its superior performance in terms of showing no drift of the position error over time emphasizing its long-term stability. In contrast, the position error of the non-cooperative LT-SLAM continuously grows resulting in two failures (occurring at about 3000 s and 3750 s) where it needed a reset.




(a) *Fully non-static, moderately changing* test case




(b) *Fully non-static, fast changing* test case.

Figure 6.12: Position error for grid size of 1.0 m over time in different test cases.

While the C-LT-SLAM coped perfectly with these two test cases, its major weak point was identified in a test run when preparing the test cases. Due to a navigation failure, a collision between two robots occurred and led to a significant orientation error of the LT-SLAM's pose estimate of one robot. In that particular case, the LT-SLAM could not recover from this localization error but instead overestimated its localization confidence resulting in a drifted local map that was also merged into the global map of the LT-SLAM server. While this should never happen in the first place, succeeding robots entering this area are

usually able to fix this map error by updating the map with a correct pose estimation. However, in this particular case, due to an adverse configuration of the remaining area, the succeeding robots could not recover the map but instead inferred to same erroneous pose estimate and thereby further reinforced the drifted map estimate of the area. While the cause of this failure is rather irrelevant for real-world applications, this example clearly points out the susceptibility of the current C-LT-SLAM to these kind of errors and will therefore be further discussed in the outlook of this work, see Section 7.2.

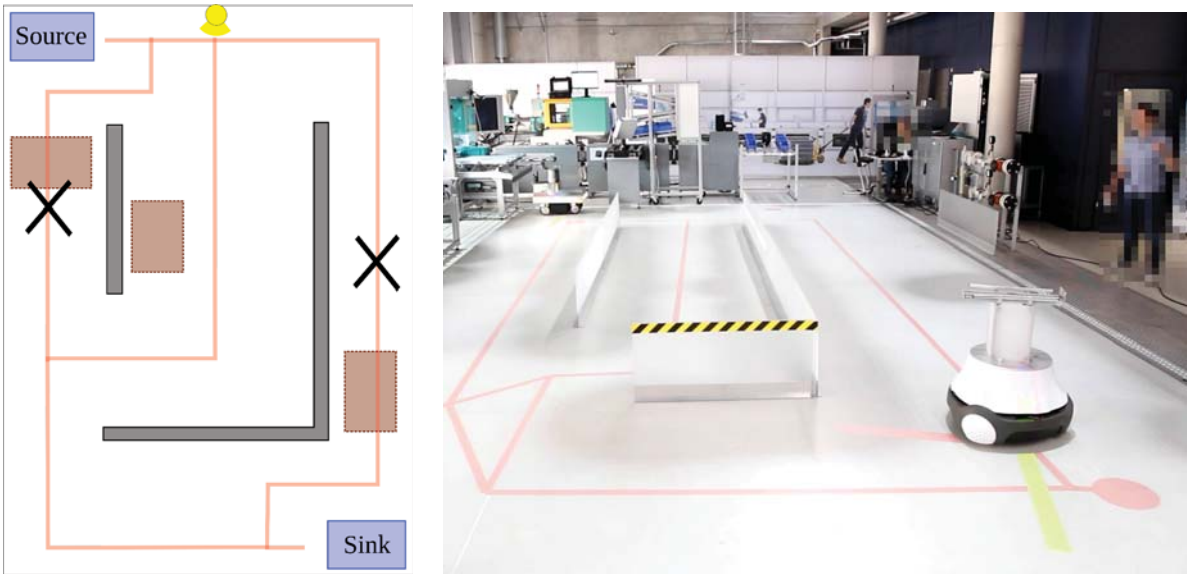
#### Influence on Path Planning Performance

While the effectiveness of cooperative map updating on improving localization robustness as well as accuracy has been clearly demonstrated, we now want to investigate its influence on path planning and resulting travel times to fulfill navigation tasks.

In our cooperative navigation system, the cooperative global path planner interfaces with the LT-SLAM server in order to compute optimal paths for the fleet based on the latest map information. In comparison to a non-cooperative navigation system where mapping and path planning is based on solely local sensor information, the resulting paths are optimized on both, latest map information and the navigation tasks of all robots in the fleet. In order to be able to measure the direct influence of cooperative mapping on the path planning performance, we disable cooperative path planning functionalities for global and local path planning within this experiment.

The experiment setup used within this experiment is depicted in Figure 6.13 where we find two mobile robots, i.e., one rob@work 3 and one Care-O-bot 4 base platform, operating in a shared workspace in a real-world, industrial-like test field at Fraunhofer IPA with approximate size of 15 m x 10 m. Additionally, a statically mounted laser scanner is integrated into the environment. In the base setting, the environment exhibits three possible routes between source and sink. During the experiment, these routes are blocked and unblocked several times by manually placing and removing cardboard boxes on the routes. In this way, the shortest route from source to sink and vice versa is non-static but varies depending on the current configuration of the boxes.

For evaluating the influence of cooperative map updating on path planning, we run the experiment in two configurations. In the first run, cooperative map updating is disabled meaning that path planner operates solely on locally gathered map information. In contrast, in the second run, cooperative map updating is performed by the two mobile robots as well as the static sensor and the path planners of the mobile robots are continuously provided with resulting global map updates. In order to guarantee equal conditions for both runs, replacements of the boxes during each test run are manually executed based on a strict schedule about the exact position of all boxes during each time interval. By disabling cooperative path planning, conflicts between the two robots can hardly be resolved by their




(a) Layout (b) Real-world environment

Figure 6.13: Setup of path planning experiment. (a) Schematic layout of environment: Configuration of static walls (grey) realize three possible routes (red) from source to sink and vice versa. Exemplary configuration of boxes (brown) lead to blockage (black crosses) of first and third route. (b) Real-world environment at start configuration without boxes.

local path planning modules. Since investigating these kinds of conflicts is out of scope for this experiment, navigation goals are carefully chosen in order to avoid conflicts.

For both test runs, we measure the total path length as well as the total travel time for both robots needed to sequentially accomplish 25 navigation tasks. Results are given in Table 6.4.

Table 6.4: Comparison of path planning performance when solely relying on local map information of local LT-SLAM in contrast to cooperatively gathered map information using the C-LT-SLAM: Total path length and travel time of both robots.

| | **Total path length [m]** | **Total travel time [s]** |
|---|---|---|
| LT-SLAM | 446.4 | 2524.1 |
| C-LT-SLAM | 408.1 | 1993.8 |

By planning on the cooperatively gathered map information using the C-LT-SLAM, the robots are able to decrease their traveled paths by almost 10 %. Moreover, the total travel time can even be decreased by about 30 %. The higher decrease of travel times with respect to path lengths can be explained by the turning maneuvers the robots have to execute once they recognize that their desired route is blocked. In the narrow environment of this experiment, these maneuvers are heavily time consuming while the additionally taken path length is moderate.

Still, this experiment demonstrates that having access to up-to-date map information of the whole environment has a significant impact on the path planning behavior since it helps avoiding blocked routes but predictively choosing optimal routes based on the current environment configuration in dynamic environments.

### Bandwidth Analysis

Enabling cooperative functionalities create the burden of an increased network traffic. In order to rate the usability of our approach for real-world industrial applications, we now want to investigate the produced load of the cooperative map updating system on the network bandwidth.

Since the dimensions of the warehouse environment are closer to our target application then the real-world test field from the previous experiment, we use the former experiment to analyze resulting network loads. For each test run of this experiment, we therefore additionally measure the network traffic and used bandwidth resulting from exchanging map information between server and agents. For rating the results with respect to real time usability, we need to look at commonly available (wireless) data rates. Wireless local area network (WLAN) standards are defined in IEEE 802.11 (IEEE Standard 2016). While the initial IEEE 802.11 standard only allowed data rates of 1-2 Mbit/s, we nowadays commonly find rates up to 54 Mbit/s with the in 2003 introduced extension IEEE 802.11g. With the further extension to IEEE 802.11n, data rates up to 600 Mbit/s become possible. Since the bandwidth of the WLAN is usually loaded with further applications, we make a conservative assumption of having a maximum data rate of 5 Mbit/s reserved for the cooperative map updating functionality.

Results of the bandwidth load for the *fully non-static, fast changing* test case are given in Table 6.5 where we find the total bandwidth load of 0.153 Mbit/s produced during this test case by four participating mobile agents. With respect to the assumed maximum bandwidth of 5 Mbit/s, cooperative map updating used about 3 % of the available bandwidth, a single agent less than 1 % respectively. For giving an estimation of the maximum number of agents that theoretically can be added without hitting bandwidth limitations, we can extrapolate this number assuming a roughly linear dependency of the needed bandwidth with respect to the number of agents. This is a clearly pessimistic assumption since the additional needed bandwidth per agent rather decreases with increasing number of agents due to the increasing spatial density of agents and resulting overlapping regions of their fields of views. With both, the conservative assumption of the available bandwidth and needed bandwidth per agent, the presented approach still allows more than 100 agents participating in map knowledge sharing on a single wireless access point. Apart from increasing the available bandwidth, this number can be further increased by applying the mentioned filtering of downstream information (as described in Section 5.4) and by reducing the upstream and

Table 6.5: Produced average data rates and bandwidth load per agent and in total for cooperative map updating in *fully non-static, fast changing* test case.

| | **Upstream data rate [Mbit/s]** | **Downstream data rate [Mbit/s]** | **Bandwidth load [Mbit/s]** |
|---|---|---|---|
| Per agent | 0.013 | 0.025 | 0.038 |
| Total | 0.052 | 0.100 | 0.153 |

downstream frequencies. However, in a large-scale warehouse or production environment equipped with several well-distributed access points, we will rarely find more than 100 agents in the vicinity of a single access point even if the fleet contains up to 1000 mobile and stationary agents. Moreover, a high density of agents in a particular area results in highly redundant sensor observations decreasing the average data volume of the transmitted map changes and thereby relaxes the bandwidth load.

### 6.2.4 Mutual Localization

Our mutual localization concept from Section 5.3 aims at supporting the LT-SLAM of the robots, especially when operating in difficult environments where only sparse static or semi-static features are available for map-based localization. In order to emphasize the necessity of additional localization information in these areas, the effectiveness of mutual localization is evaluated using the following experiment layout. As the base environment, we use once again the simulated warehouse. However, in contrast to previous experiments, we do not spawn any additional objects but leave the rooms empty. Moreover, we simulate a heterogeneous fleet of mobile robots where some of the robots are solely equipped with noisy, low-cost sensors. This is realized by simulating three Care-O-bot 4 with an unchanged sensor setup and one rob@work 3 whose sensor setup is downgraded by using a simulated laser scanner with a maximum range of 5 m and by increasing odometry noise by factor 10. These properties resemble typical setups using low-cost laser scanners as well as low-cost drive units. During this experiment, the robots navigate to arbitrary goals within the upper right room of the empty warehouse simulating an application where the robots operate in a shared workspace in proximity to each other. The experiment setup is also depicted in Figure 6.14.

To provide the mutual detection measurements, a scan-based detection module is running on each robot. Suitable for our target application, we use a learning-based approach described in (Dietrich 2018). Without any prior knowledge, the detection module is trained to detect, classify and estimate the relative pose of robots in its sensor field of view. Training data is generated by using ground truth information about the robots, e.g., their types and pose sequences in the data set. While in simulation this information is provided by

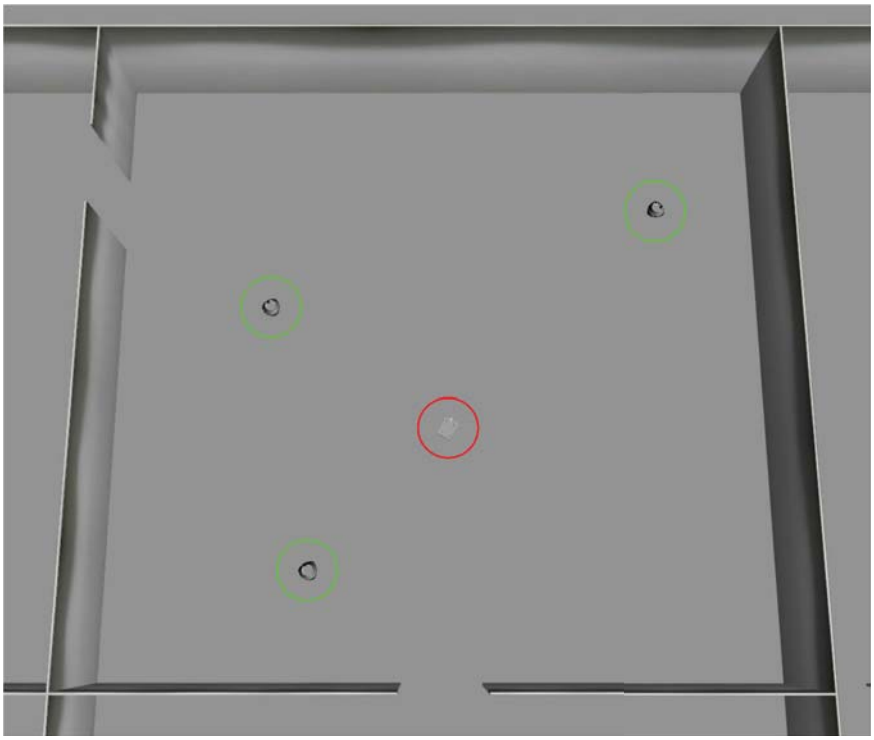


Figure 6.14: Test setup of mutual localization experiment: Simulated heterogeneous fleet of three Care-O-bot 4 (green circles) and one rob@work 3 (red circle) operating in an single empty room of warehouse environment.

the simulation, for real-world tests, the pose estimate of the LT-SLAM modules running on the robots is used. We therefore only train the detection module in highly-structured environments where the LT-SLAM has a high localization accuracy. In simulation environment, this mutual detection module running on the Care-O-bot 4 outputs relative pose measurements within a range of 5 m with mean position error of about 10 cm. The TP rate lies at approximately 50 % while about 10 % of the measurements are FPs.

For evaluating the impact of our mutual localization approach on the localization accuracy, we focus on the rob@work 3 since it imposes the hardest challenge for localization when solely basing on local sensors due to its low-cost (and low-range) sensor setup and thereby has the highest potential to benefit from localization information of the network. Similar to previous experiments, we record a data set incorporating all relevant sensor and ground truth information of all robots within the described experiment setup. Afterwards, the localization systems can be evaluated with mutual localization being activated (w mutual localization) and deactivated (w/o mutual localization) respectively. Results are given in Figure 6.15 in terms of the resulting position errors of these two configurations. We see that by enabling mutual localization, the MPE can be decreased by more than 50 % to approximately 35 cm within this experiment. Moreover, when looking at the development of the position error over time, the mutual localization is able to keep the error below about 1 m and thereby produces no localization failures during the whole experiment. In contrast,

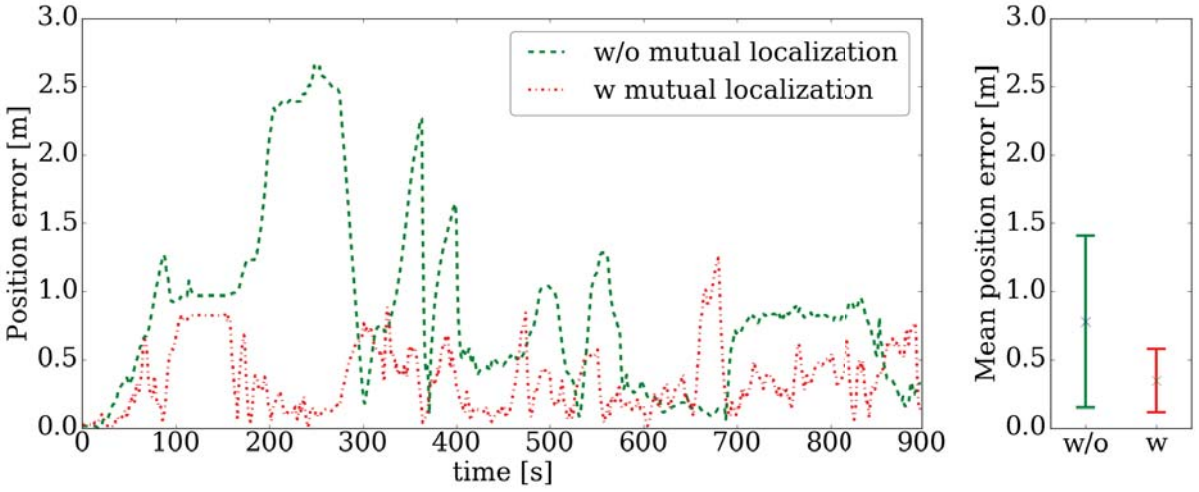



Figure 6.15: Comparing localization accuracy with/without processing mutual localization information: position error over time (left), mean position error and standard deviation (right).

without mutual localization, the position error of the LT-SLAM exhibits several peaks with a maximum error of approximately 2.5 m. These peaks are the result of several periods during the experiment where the rob@work 3 is out of sight of any walls and thereby exposed to its noisy odometry data. Most of the peaks are followed by a sharp decrease of the error when regaining sight of the static walls which at least demonstrates the ability of the LT-SLAM to handle periods of high uncertainty.

The network load caused by mutual localization in this experiment is of dimensions lower than the one caused by cooperative map updating. As a result, mutual localization does not add any relevant contribution to the total network load in our cooperative localization system. Hence, a detailed network analysis as given for cooperative map updating is skipped.

# 7 Conclusion

*This chapter summarizes the work and results of this thesis as well as discusses potential extensions and future work.*

## 7.1 Summary

In this thesis, we tackled the limitations of current navigation systems for mobile robots (MRs) in industrial applications like logistics or production in order to meet the growing demand for flexible and versatile fleets of MRs. A major issue was identified in terms of the lacking autonomy resulting from nowadays solutions which needs to be overcome with extended navigation capabilities mainly in terms of infrastructure-less localization and predictive path planning in order to cope with the dynamics of the environment and the strong requirements of industrial applications. Since those capabilities strongly rely on knowledge about the current state of the dynamic environment, maintaining and providing an up-to-date map of the environment usable by both localization and path planning were deduced as the core requirements leveraging the desired navigation capabilities. In general, LT-SLAM approaches tackle the tightly coupled problem of localization and map updating in dynamic and changing environments. However, current approaches suffer from limited knowledge about the current state of the environment when applied to strongly changing and large-scale environments. Therefore, applying concepts from networked and cloud robotics was chosen as the key to overcome these limitations and as the overall objective of this thesis.

Following this, we first presented the concept of the overall cooperative navigation system for coordinating and navigating heterogeneous fleets of MRs. The developed architecture compromised on exploiting the usage of global coherent solutions without imposing too strong requirements on wireless communication which would limit its usability in real-world applications. More specifically, the architecture realizes extended navigation capabilities through knowledge sharing and remote computations on the cloud server but keeps basic navigation functionalities on the local hardware of the MRs. Thereby, the MRs become temporal independent of the server in order to keep their operational capability despite network disruptions, limited bandwidths and communication latencies but also profit from

cooperative capabilities provided by the cloud server crucial for the long-term stability and efficiency of the navigation system.

For the particular part of long-term localization and mapping, we introduced a C-LT-SLAM consisting of a local LT-SLAM deployed on the MRs for providing high-frequency localization updates as well as updates of the local environment and an LT-SLAM server running on the cloud server enabling cooperative map updating, i.e., using the observations of the whole fleet to keep the global map up-to-date, and mutual localization, i.e., process mutual detections and relative pose measurements of the MRs to strengthen their localization.

A central issue for the C-LT-SLAM consisted in finding a suitable map representation that was able to fulfill the manifold and partially contrary requirements for our target application which mainly were defined as high geometric accuracy, properly handling the dynamics of the environment, ease of map fusion, modeling additional non-metric properties (like the reflectivity of objects to support localization in areas with sparse metric features) and appropriate computational demands, also when applied to environments of large-scale industrial sites. We tackled this by first combining two recent approaches in this field in terms of DOGM and NDT-OGM. This combination allowed to have a suitable model of the dynamics within each grid cell using an HMM whose parameters are estimated online and an NDT-based geometric model of the object within each cell to allow for lower grid sizes without loosing geometric accuracy. Subsequently, this concept was further extended by replacing the NDT with a mixture model, referred to as xGMM, to further increase the geometric accuracy, especially for highly non-linear object contours, as well as to be able to properly incorporate further non-metric states. More specifically, a binary reflectivity state was integrated to distinguish high reflective from low reflective components of the mixture model. In the experiment section, we demonstrated that this map representations yielded superior performance compared to related approaches mostly in terms of higher accuracy, robustness against outliers and coping with different kinds of environmental dynamics.

Based on this map representation, the C-LT-SLAM was derived. A main focus herein lied on the local LT-SLAM which served as the base to build upon the cooperative functionalities. A suitable approach was found in combining and extending recent promising PF-based approaches from Tipaldi, Meyer-Delius et al. (2013) and Valencia, Saarinen et al. (2014). The introduced novelty mainly consisted in classifying map information as verified and non-verified in terms of holding two map layers and integrating these layers in the localization and mapping algorithms. Additionally, a suitable observation model was introduced to compute matching scores for current scan observations based on the current estimate of the xGMM-DOGM needed to weight the particles. In our warehouse simulation, this LT-SLAM approach was already able to handle fully non-static environments with proper localization accuracies without applying any cooperative functionalities. How-

ever, as expected, the good performance was limited to slowly changing environments which supported the necessity of cooperation for environments with faster changing rates.

To enable cooperation, a concept for cooperative map updating was presented. By carefully considering respective requirements when developing the map representation as well as the local LT-SLAM, we showed that this task can be realized with comparatively simple but effective methodologies. Once sensor observations indicate a modification of the environment, respective altered map cells are upstreamed by the local LT-SLAM of the agents to the LT-SLAM server in the cloud. To avoid transmitting erroneous data, only those cells that are declared as verified are used. On the server side, the LT-SLAM server fuses these cells into the global map by a simple time stamp-based merging approach and provides the agents with map updates.

Furthermore, the concept of mutual localization was integrated into the PF-based local LT-SLAM. By basing on a versatile mixture model for representing the measurement belief of the relative pose measurements, the developed approach is able to work with a variety of potentially applicable detection modules. Additionally, the approach accounts for the case of unknown associations commonly needed when used with marker-less detection modules.

The effectiveness of the additional cooperative functionalities was extensively demonstrated with several experiments. First of all, we showed that environments with higher changing rates in which the local LT-SLAM without cooperative map updating failed can now be handled with high localization accuracy. Moreover, a real-world experiments using two mobile robots demonstrated the superior navigation behavior when basing path planning on cooperatively gathered map information. Additionally, mutual localization was evaluated in an environment with low density of static or semi-static objects and with a heterogeneous fleet of robots. This experiment demonstrated that the localization of a robot equipped with noisy, low-cost sensors can be stabilized by processing of the mutual detection measurements and is thereby able to operate in environments where localization would tend to fail without the cooperation. Finally, the network traffic was analyzed in order to rate the real-world applicability of our C-LT-SLAM. Although making the conservative assumption of only having access to data rates of maximal 5 Mbit/s, the measured network traffic was marginal, theoretically allowing to scale our approach to fleets of several hundreds of robots.

## 7.2 Outlook

While the superior performance of the presented C-LT-SLAM approach compared to state-of-the art approaches has been clearly demonstrated, further investigations and improvements are needed to further proof its real-world applicability, especially for greater fleet sizes than investigated in the experiments section.

Theoretically, the ability to scale the number of agents has been carefully considered in the developed architecture and selected approaches. More specifically, when looking at the C-LT-SLAM, increasing the fleet sizes does not significantly affect the local LT-SLAM. In contrast, the LT-SLAM server's computational demands increase approximately linearly with the number of agents. However, since computational resources are massively available in the cloud, these demands can be easily handled. Instead, the stability of the wireless communication needs to be further investigated. In the experiment section, we derived a number of more than 100 agents per access point before saturating a bandwidth of 5 Mbit/s by extrapolating the measured bandwidth load from the experiments with four agents. While this number bases on sound assumptions, a real-world verification is still receivable.

From an algorithmic perspective, a major issue remains in terms of recovery from failures, both on local as well as on cloud level. While putting effort in dealing with noisy sensor data and outlier measurements clearly increased the robustness, a sequence of adverse events may still lead to localization errors of the LT-SLAM. In Subsection 6.2.3, we gave an example of such a situation caused by a collision of two robots. Since achieving robustness against each possible occurring event causing such an error is rather impracticable, a significant improvement consists in providing awareness of potential failures and recovery routines to resolve them.

On the local LT-SLAM, error detection routines could be the base to prevent mapping with an erroneous pose estimate as well as recovering the localization. An approach for detecting localization errors may base on evaluating inconsistencies between sensor observations and the current map estimate. Since an inconsistency is not necessarily the result of an localization error but can also occur due to map changes, the detection is in general not trivial, especially when dealing with strongly changing environments and noisy sensors. However, when investigating the structure of the inconsistency, the cause may be assessed to some extent.

For instance, let us consider an inconsistency as a result from an orientation error in an unchanged map. In this simple case, all objects in the current sensor observation will be displaced with respect to the map. More specifically, the displacements can be described with a unique transformation matrix which results from the robot's orientation error. Since an area is very unlikely to change in a way that all objects are simply rotated or translated by some amount with respect to the robot frame, this structure clearly indicates a localization error. In contrast, the displacement of only a part of the objects by arbitrary amounts typically indicates an actual map change. As a matter of fact, typical situations are more complex than the described ones, e.g, when dealing with map changes, combined position and orientation errors as well as erroneous data at the same time. Still, the basic idea of classifying observation inconsistencies yield huge potential to infer localization errors, espe-

cially when looking at recent advances of pattern recognition and situation interpretation using machine learning techniques.

A question that remains is what to do with the information of a possible localization error. First of all, the simple awareness can be used to impede erroneous mapping as well as to report it as an error feedback. However, autonomy is only achieved if the robots can resolve the error. To recover the pose estimate, knowledge about the localization error can be used. If the actual error can be assessed, e.g., by the transformation matrix of the dislocated objects, the particles can simply be sampled around the inferred pose from the localization error estimate. Otherwise, if the detection approach does not provide any kind of this information, global localization techniques can be applied.

For the LT-SLAM server, similar considerations can be carried out to prevent fusing erroneous data from failing agents into the global map as well as to be able to recover the map once erroneous data has been fused into the global map before. Similar to the concept of detecting localization errors in the local LT-SLAM, the structure of the map upstreams may be investigated to rate its conformity. Training an algorithm to be able to distinguish map upstreams containing typically occurring map changes from those resulting from typical localization or mapping failures could be the key to prevent the LT-SLAM server fusing erroneous data.

# Bibliography


Abbenseth 2016 — Abbenseth, Jannik, 2016. *Cooperative Multi-Robot Global Path Planning.* Technische Universität Darmstadt, Master thesis

Abbenseth, Lopez et al. 2017 — Abbenseth, Jannik; Lopez, Felipe Garcia; Henkel, Christian; Dörr, Stefan, 2017. Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments, *In Proceedings of the Symposium on Applied Computing - SAC '17.* ACM, pp. 283–288, ISBN 9781450344869, DOI:10.1145/3019612.3019710

Abraham, Ge et al. 2009 — Abraham, Aswin Thomas; Ge, Shuzhi Sam; Tao, Pey Yuen, 2009. A topological approach of path planning for autonomous robot navigation in dynamic environments, *In Robots and Systems.* IEEE Press, pp. 4907–4912, ISBN 978-1-4244-3803-7, DOI:10.1109/IROS.2009.5354771

Andreasson, Bouguerra et al. 2015 — Andreasson, Henrik; Bouguerra, Abdelbaki; Cirillo, Marcello; Dimitrov, Dimitar Nikolaev; Driankov, Dimiter; Karlsson, Lars; Lilienthal, Achim J.; Pecora, Federico; Saarinen, Jari Pekka; Sherikov, Aleksander; Stoyanov, Todor, 2015. Autonomous Transport Vehicles: Where We Are and What Is Missing. *IEEE Robotics & Automation Magazine* **vol. 22** (1), pp. 64–75, DOI:10.1109/MRA.2014.2381357

Arumugam, Enti et al. 2010 — Arumugam, Rajesh; Enti, Vikas Reddy; Bingbing, Liu; Xiaojun, Wu; Baskaran, Krishnamoorthy; Kong, Foong Foo; Kumar, A. Senthil; Meng,

Kang Dee; Kit, Goh Wai, 2010. DAvinCi: A cloud computing framework for service robots, *In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 3084–3089, ISBN 978-1-4244-5038-1, DOI:10.1109/ROBOT.2010.5509469

Atanasov, Le Ny et al. 2015 — Atanasov, Nikolay; Le Ny, Jerome; Daniilidis, Kostas; Pappas, George J., 2015. Decentralized active information acquisition: Theory and application to multi-robot SLAM, *In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 4775–4782, ISBN 978-1-4799-6923-4, DOI:10.1109/ICRA.2015.7139863

Beinschob and Reinke 2015 — Beinschob, Patric; Reinke, Christoph, 2015. Graph SLAM based mapping for AGV localization in large-scale warehouses, *In Procceedings of the 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP).* New York City, NY: IEEE, pp. 245–248, ISBN 978-1-4673-8200-7, DOI:10.1109/ICCP.2015.7312637

Biber and Duckett 2009 — Biber, Peter; Duckett, Tom, 2009. Experimental analysis of sample-based maps for long-term SLAM. *International Journal of Robotics Research* **vol. 28** (1), pp. 20–33, DOI:10.1177/0278364908096286

Biber and Strasser 2003 — Biber, Peter; Strasser, Wolfgang, 2003. The normal distributions transform: a new approach to laser scan matching, *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003).* New York City, NY: IEEE, vol. 3, pp. 2743–2748,

ISBN 0-7803-7860-1,
DOI:10.1109/IROS.2003.1249285

Biswas and Veloso 2014 — Biswas, Jit; Veloso, Marco, 2014. Episodic Non-markov Localization: Reasoning About Short-Term and Long-Term Features, *In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 3969–3974

Bohg, Morales et al. 2014 — Bohg, Jeannette; Morales, Antonio; Asfour, Tamim; Kragic, Danica, 2014. Data-Driven Grasp Synthesis—A Survey. *IEEE Transactions on Robotics* **vol. 30** (2), pp. 289–309, DOI:10.1109/TRO.2013.2289018

Breitenstein, Reichlin et al. 2011 — Breitenstein, M. D.; Reichlin, F.; Leibe, B.; Koller-Meier, E.; Van Gool, L., 2011. Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **vol. 33** (9), pp. 1820–1833, DOI:10.1109/TPAMI.2010.232

Bubeck, Gruhler et al. 2017 — Bubeck, Alexander; Gruhler, Matthias; Reiser, Ulrich; Weißhardt, Florian, 2017. Vom fahrerlosen Transportsystem zur intelligenten mobilen Automatisierungsplattform. In Vogel-Heuser, Birgit; Baurnhansl, Thomas; ten Hompel, Michael (eds.): *Handbuch Industrie 4.0 Bd. 4 : Allgemeine Grundlagen*. Springer Vieweg, pp. S. 83–95, ISBN 978-3-662-53254-6 5, DOI:10.1007/978-3-662-53254-65

Burgard, Moors et al. 2000 — Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S., 2000. Collaborative multi-robot exploration, *In Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, vol. 1, pp. 476–481, ISBN 0-7803-5886-4, DOI:10.1109/ROBOT.2000.844100

Cardarelli, Sabattini et al. 2015 — Cardarelli, Elena; Sabattini, Lorenzo; Secchi, Cristian; Fantuzzi, Cesare, 2015. Cloud robotics paradigm for enhanced navigation of autonomous vehicles in real world industrial applications, *In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 4518–4523, ISBN 978-1-4799-9994-1, DOI:10.1109/IROS.2015.7354019

Cardarelli, Digani et al. 2017 — Cardarelli, Elena; Digani, Valerio; Sabattini, Lorenzo; Secchi, Cristian; Fantuzzi, Cesare, 2017. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics* **vol. 45**, pp. 1–13, DOI:10.1016/j.mechatronics.2017.04.005

Churchill and Newman 2012 — Churchill, Winston; Newman, Paul, 2012. Practice makes perfect? Managing and leveraging visual experiences for lifelong navigation, *In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 4525–4532, ISBN 9781467314039, DOI:10.1109/ICRA.2012.6224596

Connette, Meister et al. 2007 — Connette, Christian P.; Meister, Oliver; Hagele, Martin; Trommer, Gert F., 2007. Decomposition of line segments into corner and statistical grown line features in an EKF-SLAM framework, *In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 3884–3891, ISBN 978-1-4244-0911-2, DOI:10.1109/IROS.2007.4399404

Cox 1991 — Cox, Ingemar J., 1991. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation* **vol. 7** (2), pp. 193–204, DOI:10.1109/70.75902

Curran, Furey et al. 2011 — Curran, Kevin; Furey, Eoghan; Lunney, Tom; Santos, Jose; Woods, Derek; Mc Caughey, Aiden, 2011. An Evaluation of Indoor Location Determination Technologies. *Journal of Location Based Services* **vol. 5** (2), pp. 1748–9725

D'Andrea 2012 — D'Andrea, Raffaello, 2012. Guest Editorial: A Revolution in the Warehouse: A Retrospective on Kiva Systems and the Grand Challenges Ahead. *IEEE Transactions on Automation Science and Engineering* **vol. 9** (4), pp. 638–639, DOI:10.1109/TASE.2012.2214676

Dellaert, Fox et al. 1999 — Dellaert, Frank; Fox, Dieter; Burgard, Wolfram; Thrun, Sebastian, 1999. Monte Carlo localization for mobile robots, *In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, vol. 2, pp. 1322–1328, ISBN 0-7803-5180-0, DOI:10.1109/ROBOT.1999.772544

Dempster and Laird 1977 — Dempster, Arthur P.; Laird, Nan M., 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society* , pp. 1–38

Deutsch, Liu et al. 2016 — Deutsch, Isaac; Liu, Ming; Siegwart, Roland, 2016. A framework for multi-robot pose graph SLAM, *In Proceedings of the 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR).* New York City, NY: IEEE, pp. 567–572, ISBN 978-1-4673-8959-4, DOI:10.1109/RCAR.2016.7784092

Dietrich 2018 — Dietrich, Robin, 2018. *Deep Learning Based Mutual Robot Detection and Relative Position Estimation.* University Stuttgart, Master thesis

Dietrich and Dörr 2019 — Dietrich, Robin; Dörr, Stefan, 2019. Deep Learning-Based Mutual Detection and Collaborative Localization for Mobile Robot Fleets

Using Solely 2D LIDAR Sensors, *In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019).* New York City, NY: IEEE, pp. 6706–6713, ISBN 978-1-7281-4004-9, DOI:10.1109/IROS40897.2019.8967574

Dörr, Barsch et al. 2016 — Dörr, Stefan; Barsch, Paul; Garcia Lopez, Felipe; Gruhler, Matthias, 2016. Cooperative Longterm SLAM for Navigating Mobile Robots in Industrial Applications, *In Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI).* New York City, NY: IEEE, pp. 297–303, DOI:10.1109/MFI.2016.7849504

Durrant-Whyte 1994 — Durrant-Whyte, Hugh, 1994. Where am I? A tutorial on mobile vehicle localization. *Industrial Robot: An International Journal* **vol. 21** (2), pp. 11–16

Einhorn and Gross 2015 — Einhorn, Erik; Gross, Horst Michael, 2015. Generic NDT mapping in dynamic environments and its application for lifelong SLAM. *Robotics and Autonomous Systems* **vol. 69** (1), pp. 28–39, DOI:10.1016/j.robot.2014.08.008

Elfes 1989 — Elfes, Alberto, 1989. Using occupancy grids for mobile robot perception and navigation. *Computer* **vol. 22** (6), pp. 46–57, DOI:10.1109/2.30720

Endres, Hess et al. 2012 — Endres, Felix; Hess, Jurgen; Engelhard, Nikolas; Sturm, Jurgen; Cremers, Daniel; Burgard, Wolfram, 2012. An evaluation of the RGB-D SLAM system, *In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 1691–1696, ISBN 978-1-4673-1405-3, DOI:10.1109/ICRA.2012.6225199

Engel and Heinen 2010 — Engel, Paulo Martins; Heinen, Milton Roberto, 2010. Incremental Learning of Multivariate Gaussian Mixture Models. In *Brazilian Symposium on Artificial Intelligence*. Springer, pp. 82–91, ISBN 978-3-642-16137-7, DOI:10.1007/978-3-642-16138-4_9

Engmann 2016 — Engmann, Falk, 2016. *Medial Axis graph - Generating and updating roadmaps for path planning in mobile robotics*. University Stuttgart, Master thesis

Fisher 1995 — Fisher, Nicholas I., 1995. *Statistical analysis of circular data*. Cambridge: Cambridge University Press. ISBN 9780511564345, DOI:10.1017/CBO9780511564345

Forsberg, Larsson et al. 1995 — Forsberg, Johan; Larsson, Ulf; Wernersson, Ake, 1995. Mobile robot navigation using the range-weighted Hough transform. *IEEE Robotics & Automation Magazine* **vol. 2** (1), pp. 18–26, DOI:10.1109/100.388295

Fox 2003 — Fox, Dieter, 2003. Adapting the Sample Size in Particle Filters Through KLD-Sampling. *The International Journal of Robotics Research* **vol. 22** (12), pp. 985–1003, DOI:10.1177/0278364903022012001

Fox, Burgard et al. 1997 — Fox, Dieter; Burgard, Wolfram; Thrun, Sebastian, 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* **vol. 4** (1), pp. 23–33, DOI:10.1109/100.580977

Fox, Burgard et al. 2000 — Fox, Dieter; Burgard, Wolfram; Kruppa, Hannes; Thrun, Sebastian, 2000. A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots* **vol. 8** (3), pp. 325–344, DOI:10.1023/A:1008937911390

Frese and Beyerer 2011 — Frese, Christian; Beyerer, Jürgen, 2011. A comparison of motion planning algorithms for

cooperative collision avoidance of multiple cognitive automobiles, *In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium.* pp. 1156–1162, ISBN 9781457708909, DOI:10.1109/IVS.2011.5940489

Garcia Lopez 2018 — Garcia Lopez, Felipe, 2018. *Predictive and Cooperative Online Motion Planning: A Contribution to Networked Mobile Robot Navigation in Industrial Applications.* Fraunhofer Verlag. Stuttgarter Beiträge zur Produktionsforschung, 93. University Stuttgart, Phd thesis, ISBN 978-3-8396-1531-7

Garcia Lopez, Abbenseth et al. 2017 — Garcia Lopez, Felipe; Abbenseth, Jannik; Henkel, Christian; Dörr, Stefan, 2017. A predictive online path planning and optimization approach for cooperative mobile service robot navigation in industrial applications, *In Proceedings of the 2017 European Conference on Mobile Robots (ECMR).* New York City, NY: IEEE, pp. 107–112, ISBN 9781538610961, DOI:10.1109/ECMR.2017.8098677

Gebru, Alameda-Pineda et al. 2016 — Gebru, Israel Dejene; Alameda-Pineda, Xavier; Forbes, Florence; Horaud, Radu, 2016. EM Algorithms for Weighted-Data Clustering with Application to Audio-Visual Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **vol. 38** (12), pp. 2402–2415, DOI:10.1109/TPAMI.2016.2522425

Gerkey and Mataric 2002 — Gerkey, B.P.; Mataric, M.J., 2002. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation* **vol. 18** (5), pp. 758–768, DOI:10.1109/TRA.2002.803462

Gregoire, Bonnabel et al. 2013 — Gregoire, Jean; Bonnabel, Silvère; de La Fortelle, Arnaud, 2013. *Optimal cooperative motion planning*

*for vehicles at intersections*, available at: http://arxiv.org/abs/1310.7729, 1310.7729

Grisetti, Stachniss et al. 2007 — Grisetti, Giorgio; Stachniss, Cyrill; Burgard, Wolfram, 2007. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics* **vol. 23** (1), pp. 34–46, DOI:10.1109/TRO.2006.889486

Haegele 2017 — Haegele, Martin, 2017. *World Robotics 2017 – Service Robots*, Tech. Rep., IFR Statistical Department

Hahnel, Burgard et al. 2003 — Hahnel, Daniel; Burgard, Wolfram; Fox, Dieter; Thrun, Sebastian, 2003. An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements, *In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, vol. 1, pp. 206–211, ISBN 0-7803-7860-1, DOI:10.1109/IROS.2003.1250629

Haug 2012 — Haug, Anton J., 2012. *Bayesian estimation and tracking : a practical guide.*. Hoboken, New Jersey: John Wiley & Sons. ISBN 9780470621707, DOI:10.1002/9781118287798

Hermann, Pentek et al. 2016 — Hermann, Mario; Pentek, Tobiass; Otto, Boris, 2016. Design Principles for Industrie 4.0 Scenarios, *In 2016 49th Hawaii International Conference on System Sciences (HICSS).* New York City, NY: IEEE, pp. 3928–3937, ISBN 978-0-7695-5670-3, DOI:10.1109/HICSS.2016.488

Hess, Kohler et al. 2016 — Hess, Wolfgang; Kohler, Damon; Rapp, Holger; Andor, Daniel, 2016. Real-time loop closure in 2D LIDAR SLAM, *In Proceeding of the 2016 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE,

pp. 1271–1278, ISBN 978-1-4673-8026-3, DOI:10.1109/ICRA.2016.7487258

Howard 2006 — Howard, Andrew, 2006. Multi-robot Simultaneous Localization and Mapping using Particle Filters. *The International Journal of Robotics Research* **vol. 25** (12), pp. 1243–1256, DOI:10.1177/0278364906072250

Howard, Matark et al. 2002 — Howard, Andrew; Matark, Maja J.; Sukhatme, Gaurav S., 2002. Localization for mobile robot teams using maximum likelihood estimation, *In Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*. New York City, NY: IEEE, vol. 1, pp. 434–439, ISBN 0-7803-7398-7, DOI:10.1109/IRDS.2002.1041428

IEEE Standard 2016 — IEEE Standard, 2016. *IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*. New York, NY, USA: IEEE. ISBN 9781504436458

ISO 8373 2012 — ISO 8373, 2012. *ISO 8373 - Robotis and robotic devices - Vocabulary*

Jungheinrich AG 2018 — Jungheinrich AG, 2018. *EZS 350a - automatisierte Produktionsversorgung mit Jungheinrich*, available at: https://www.jungheinrich.ch/unternehmen/news/presse/artikel/nI/3285-ezs-350a-automatisierte-produktionsversorgung-mit-jungheinrich/, accessed on: 2018-06-12

Kalman 1960 — Kalman, Rudolph Emil, 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* **vol. 82** (1), pp. 35–45

Kehoe, Matsukawa et al. 2013 — Kehoe, Ben; Matsukawa, Akihiro; Candido, Sal; Kuffner, James; Goldberg, Ken, 2013. Cloud-based robot grasping with the google object recognition engine, *In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 4263–4270, ISBN 978-1-4673-5643-5, DOI:10.1109/ICRA.2013.6631180

Kehoe, Patil et al. 2015 — Kehoe, Ben; Patil, Sachin; Abbeel, Pieter; Goldberg, Ken, 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering* **vol. 12** (2), pp. 398–409, DOI:10.1109/TASE.2014.2376492

Khan, Member et al. 2016 — Khan, Sheraz; Member, Student; Wollherr, Dirk; Member, Senior; Buss, Martin, 2016. Modeling Laser Intensities For Simultaneous Localization and Mapping. *IEEE Robotics and Automation Letters* **vol. 1** (2), pp. 692 – 699, DOI:10.1109/LRA.2016.2516592

Khan, Balch et al. 2005 — Khan, Zia; Balch, Tucker; Dellaert, Frank, 2005. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **vol. 27** (11), pp. 1805–1819, DOI:10.1109/TPAMI.2005.223

Kleiner, Sun et al. 2011 — Kleiner, Alexander; Sun, Dali; Meyer-Delius, Daniel, 2011. ARMO: Adaptive road map optimization for large robot teams, *In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 3276–3282, ISBN 978-1-61284-456-5, DOI:10.1109/IROS.2011.6094734

Konolige, Grisetti et al. 2010 — Konolige, K; Grisetti, G; Kuemmerle, Rainer; Burgard, W; Limketkai, B; Vincent, R, 2010.

Efficient Sparse Pose Adjustment for 2D mapping, *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 22–29, ISBN 978-1-4244-6674-0, DOI:10.1109/IROS.2010.5649043

Kraetzschmar, Pagès Gassull et al. 2004

Kraetzschmar, Gerhard K.; Pagès Gassull, Guillem; Uhl, Klaus, 2004. Probabilistic quadtrees for variable-resolution mapping of large environments. *IFAC Proceedings Volumes* **vol. 37** (8), pp. 675–680, DOI:10.1016/S1474-6670(17)32056-6

Kuffner 2010

Kuffner, James, 2010. Cloud-enabled humanoid robots, *In Proceedings of the10th IEEE-RAS International Conference on Humanoid Robots (Humanoids).* New York City, NY: IEEE

Kuipers and Byun 1991

Kuipers, Benjamin; Byun, Yung-Tai, 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems* **vol. 8** (1-2), pp. 47–63, DOI:10.1016/0921-8890(91)90014-C

Latombe 1991

Latombe, Jean-Claude, 1991. *Robot Motion Planning*. Boston, MA: Springer US. ISBN 978-0-7923-9206-4, DOI:10.1007/978-1-4615-4022-9

LaValle 2006

LaValle, Steven M, 2006. *Planning algorithms*. Cambridge, MA: Cambridge University Press. ISBN 9780511546877, DOI:10.1017/CBO9780511546877

Leofante, Le Moal et al. 2015

Leofante, Francesco; Le Moal, Gwenole; Garcia, Gaetan; Rabaté, Patrice, 2015. Improving Monte Carlo Localization using Reflective Markers: An Experimental Analysis, *In Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV@IROS'15).* Nantes, Frances: IEEE

Lowe 2004 — Lowe, David G, 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **vol. 60** (2), pp. 91–110, DOI:10.1023/B:VISI.0000029664.99615.94

Lu, Hershberger et al. 2014 — Lu, David V; Hershberger, Dave; Smart, William D, 2014. Layered costmaps for context-sensitive navigation, *In Proceedings of the 2014 IEEE International Conference on Intelligent Robots and Systems (IROS).* pp. 709–715, ISBN 9781479969340, DOI:10.1109/IROS.2014.6942636

Lu and Milios 1997 — Lu, F.; Milios, E., 1997. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots* **vol. 4** (4), pp. 333–349, DOI:10.1023/A:1008854305733

Luna and Bekris 2011 — Luna, Ryan; Bekris, Kostas E., 2011. Efficient and complete centralized multi-robot path planning, *In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 3268–3275, ISBN 978-1-61284-456-5, DOI:10.1109/IROS.2011.6095085

Martinelli, Pont et al. 2005 — Martinelli, Agostino; Pont, Frederic; Siegwart, Roland, 2005. Multi-Robot Localization Using Relative Observations, *In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA).* pp. 2808–2813

Mell and Grance 2009 — Mell, Peter; Grance, Timothy, 2009. *The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology*, Tech. Rep., National Institute of Standards and Technology„ DOI:10.6028/NIST.SP.800-145

Meyer-Delius 2011 — Meyer-Delius, Daniel, 2011. *Probabilistic Modeling of Dynamic Environments for Mobile Robots*. Universität Freiburg, Ph.D. thesis

Meyer-Delius, Beinhofer et al. 2012 — Meyer-Delius, Daniel; Beinhofer, Maximilian; Burgard, Wolfram, 2012. Occupancy Grid Models for Robot Mapping in Changing Environments, *In Procceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence.* AAAI, pp. 2024–2030, ISBN 9781577355687

Mohanarajah, Usenko et al. 2015 — Mohanarajah, Gajamohan; Usenko, Vladyslav; Singh, Mayank; D'Andrea, Raffaello; Waibel, Markus, 2015. Cloud-Based Collaborative 3D Mapping in Real-Time With Low-Cost Robots. *IEEE Transactions on Automation Science and Engineering* **vol. 12** (2), pp. 423–431, DOI:10.1109/TASE.2015.2408456

Mongillo and Deneve 2008 — Mongillo, Gianluigi; Deneve, Sophie, 2008. Online Learning with Hidden Markov Models. *Neural Computation* **vol. 20** (7), pp. 1706–1716, DOI:10.1162/neco.2008.10-06-351

Montemerlo, Thrun et al. 2002 — Montemerlo, Michael; Thrun, Sebastian; Whittaker, William, 2002. Conditional particle filters for simultaneous mobile robot localization and people-tracking. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA)* , pp. 695–701, DOI:10.1109/ROBOT.2002.1013439

Montemerlo, Thrun et al. 2003 — Montemerlo, Michael; Thrun, Sebastian; Koller, Daphne; Wegbreit, Ben, 2003. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges, *In Proceedings of the 2003 International Conference on Artificial Intelligence (IJCAI).* pp. 1151–1156

Murphy 2000a — Murphy, Kevin P., 2000. Bayesian map learning in dynamic environments, *In Proceedings of the 12th International Conference on Neural Information.* Cambridge, MA: MIT Press, pp. 1015–1021

Murphy 2000b — Murphy, Robin R., 2000. Marsupial and shape-shifting robots for urban search and rescue. *IEEE Intelligent Systems* **vol. 15** (2), pp. 14–19, DOI:10.1109/5254.850822

Nerurkar, Roumeliotis et al. 2009 — Nerurkar, Esha D.; Roumeliotis, Stergios I.; Martinelli, Agostino, 2009. Distributed maximum a posteriori estimation for multi-robot cooperative localization, *In Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 1402–1409, ISBN 978-1-4244-2788-8, DOI:10.1109/ROBOT.2009.5152398

Parker 2003 — Parker, Lynne E., 2003. The effect of heterogeneity in teams of 100+, *In Multi-Robot Systems Volume II: From Swarms to Intelligent Automata.* Kluwer, pp. 205–2015

Pavek, Smith et al. 2008 — Pavek, Kerry; Smith, Robert; Keith, Arthur; Ray, Higgins; Hansen, Eric; Ragon, Mark; Jones, Jeffery; Wade, Robert; English, Woody; Novak, Brian; Bruemmer, David; Barbera, Anthony, 2008. *Autonomy Levels for Unmanned Systems (ALFUS) Framework Volume I: Terminology Version 2.0*, available at: https://www.nist.gov/sites/default/files/documents/el/isd/ks/NISTSP{_}1011-I-2-0.pdf

Prorok, Bahr et al. 2012 — Prorok, Amanda; Bahr, Alexander; Martinoli, Alcherio, 2012. Low-cost collaborative localization for large-scale multi-robot systems, *In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 4236–4241,

ISBN 978-1-4673-1405-3,
DOI:10.1109/ICRA.2012.6225016

Riazuelo, Civera et al. 2014 — Riazuelo, Luis; Civera, Javier; Montiel, J. M. Martinez, 2014. C2TAM: A Cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems* **vol. 62** (4), pp. 401–413, DOI:10.1016/j.robot.2013.11.007

RoboEarth 2018 — RoboEarth, 2018. *RoboEarth | A World Wide Web for Robots*, available at: http://roboearth.ethz.ch/, accessed on: 2018-04-30

Roumeliotis and Bekey 2002 — Roumeliotis, Stergios I.; Bekey, George A., 2002. Distributed multirobot localization. *IEEE Transactions on Robotics and Automation* **vol. 18** (5), pp. 781–795, DOI:10.1109/TRA.2002.803461

Rubin 1987 — Rubin, Donald B., 1987. The calculation of posterior distributions by data augmentation: Comment: A noniterative sampling/importance resampling alternative to the data augmentation. *Journal of the American Statistical Association* **vol. 82** (398), pp. 543–546

Saarinen, Andreasson et al. 2013a — Saarinen, Jari; Andreasson, Henrik; Stoyanov, Todor; Ala-Luhtala, Juha; Lilienthal, Achim J., 2013. Normal Distributions Transform Occupancy Maps: Application to large-scale online 3D mapping, *In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*. New York City, NY: IEEE, pp. 2233–2238, ISBN 978-1-4673-5643-5, DOI:10.1109/ICRA.2013.6630878

Saarinen, Andreasson et al. 2013b — Saarinen, Jari; Andreasson, Henrik; Stoyanov, Todor; Lilienthal, Achim J., 2013. Normal distributions transform Monte-Carlo localization (NDT-MCL), *In 2013 IEEE/RSJ International*

*Conference on Intelligent Robots and Systems.* New York City, NY: IEEE, pp. 382–389, ISBN 978-1-4673-6358-7, DOI:10.1109/IROS.2013.6696380

Saarinen, Stoyanov et al. 2013 — Saarinen, Jari; Stoyanov, Todor; Andreasson, Henrik; Lilienthal, Achim J., 2013. Fast 3D mapping in highly dynamic environments using normal distributions transform occupancy maps, *In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* New York City, NY: IEEE, pp. 4694–4701, ISBN 978-1-4673-6358-7, DOI:10.1109/IROS.2013.6697032

Saarinen, Andreasson et al. 2013c — Saarinen, Jari P; Andreasson, Henrik; Stoyanov, Todor; Lilienthal, Achim J, 2013. 3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments. *The International Journal of Robotics Research* **vol. 32** (14), pp. 1627–1644, DOI:10.1177/0278364913499415

Sabattini, Cardarelli et al. 2015 — Sabattini, Lorenzo; Cardarelli, Elena; Digani, Valerio; Secchi, Cristian; Fantuzzi, Cesare; Fuerstenberg, Kay, 2015. Advanced sensing and control techniques for multi AGV systems in shared industrial environments, *In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA).* New York City, NY: IEEE, pp. 1–7, ISBN 978-1-4673-7929-8, DOI:10.1109/ETFA.2015.7301488

Schulz, Burgard et al. 2001 — Schulz, Dirk; Burgard, Worlfram; Fox, Dieter; Cremers, Armin B., 2001. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association, *In Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, vol. 2, pp. 1665–1670,

ISBN 0-7803-6576-3,
DOI:10.1109/ROBOT.2001.932850

Se, Lowe et al. 2002 — Se, Stephen; Lowe, David; Little, Jim, 2002. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research* **vol. 21** (8), pp. 735–758, DOI:10.1177/027836402761412467

Shannon 1948 — Shannon, Claude E., 1948. A mathematical theory of communication (parts I and II). *Bell System Tech. J.* **vol. 27**, pp. 379–423

Sharon, Stern et al. 2015 — Sharon, Guni; Stern, Roni; Felner, Ariel; Sturtevant, Nathan R., 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* **vol. 219**, pp. 40–66, DOI:10.1016/j.artint.2014.11.006

Siciliano and Khatib 2016 — Siciliano, Bruno; Khatib, Oussama, 2016. *Springer Handbook of Robotics*. Berlin: Springer. ISBN 978-3-540-30301-5

Siegwart and Nourbakhsh 2004 — Siegwart, Roland; Nourbakhsh, Illah R., 2004. *Introduction to Autonomous Mobile Robots*. Cambridge, MA: The MIT Press. ISBN 0-262-19502-X

Simmons, Singh et al. 2001 — Simmons, Reid; Singh, Sanjiv; Hershberger, David; Ramos, Josue; Smith, Trey, 2001. First Results in the Coordination of Heterogeneous Robots for Large-Scale Assembly. In *Experimental Robotics VII*. Berlin, Heidelberg: Springer, pp. 323–332, DOI:10.1007/3-540-45118-8_33

Smith and Cheeseman 1986 — Smith, Randall C.; Cheeseman, Peter, 1986. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research* **vol. 5** (4), pp. 56–68, DOI:10.1177/027836498600500404

SSI Schäfer GmbH 2018 — SSI Schäfer GmbH, 2018. *AGV WEASEL*, available at: https://www.ssi-schaefer.com/en-de/products/conveying-transport/automated-guided-vehicles/fahrerloses-transportsystem-weasel-1918, accessed on: 2018-06-12

Stoyanov, Magnusson et al. 2012 — Stoyanov, Todor; Magnusson, Martin; Andreasson, Henrik; Lilienthal, Achim J, 2012. Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The International Journal of Robotics Research* **vol. 31** (12), pp. 1377–1393, DOI:10.1177/0278364912460895

Thrun and Bücken 1996 — Thrun, Sebastian; Bücken, Arno, 1996. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation, *In Proceedings of the National Conference on Artificial Intelligence.* vol. 13, pp. 944–950, ISBN 0-262-51091-X, DOI:10.1.1.153.3743

Thrun and Montemerlo 2006 — Thrun, Sebastian; Montemerlo, Michael, 2006. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research* **vol. 25** (5-6), pp. 403–429, DOI:10.1177/0278364906065387

Thrun, Fox et al. 2001 — Thrun, Sebastian; Fox, Dieter; Burgard, Wolfram; Dellaert, Frank, 2001. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence* **vol. 128** (1-2), pp. 99–141, DOI:10.1016/S0004-3702(01)00069-8

Thrun, Fox et al. 2005 — Thrun, Sebastian; Fox, Dieter; Burgard, Wolfram, 2005. *Probabilistic robotics*. Cambridge, MA: MIT Press. ISBN 9780262201629

Tipaldi, Meyer-Delius et al. 2013 — Tipaldi, Gian Diego; Meyer-Delius, Daniel; Burgard, Wolfram, 2013. Lifelong localization in

changing environments. *The International Journal of Robotics Research* **vol. 32** (14), pp. 1662–1678, DOI:10.1177/0278364913502830

Ueda, Nakano et al. 1998 — Ueda, Naonari; Nakano, Ryohei; Ghahramani, Zoubin; Hinton, Geoffrey E., 1998. Split and merge EM algorithm for improving Gaussian mixture density estimates, *In Neural Networks for Signal Processing VIII. Proceedings of the 1998 IEEE Signal Processing Society Workshop (Cat. No.98TH8378).* pp. 274–283, ISBN 0-7803-5060-X, DOI:10.1109/NNSP.1998.710657

Ullrich 2014 — Ullrich, Gunter, 2014. *Fahrerlose Transportsysteme: Eine Fibel - mit Praxisanwendungen - zur Technik - für die Planung*. Wiesbaden: Springer Fachmedien. ISBN 978-3-8348-2592-6, DOI:10.1007/978-3-8348-2592-6

Valencia, Saarinen et al. 2014 — Valencia, Rafael; Saarinen, Jari; Andreasson, Henrik; Vallve, Joan; Andrade-Cetto, Juan; Lilienthal, Achim J., 2014. Localization in highly dynamic environments using dual-timescale NDT-MCL, *In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA).* New York City, NY: IEEE, pp. 3956–3962, ISBN 978-1-4799-3685-4, DOI:10.1109/ICRA.2014.6907433

Vogel-Heuser, Bauernhansl et al. 2014 — Vogel-Heuser, Birgit; Bauernhansl, Thomas; ten Hompel, Michael, 2014. *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Wiesbaden: Springer Fachmedien. ISBN 978-3-658-04681-1, DOI:10.1007/978-3-658-04682-8

Wan, Tang et al. 2016 — Wan, Jiafu; Tang, Shenglong; Yan, Hehua; Li, Di; Wang, Shiyong; Vasilakos, Athanasios V., 2016. Cloud Robotics: Current Status and Open Issues. *IEEE Access* , pp. 1–1, DOI:10.1109/ACCESS.2016.2574979

Yang, Baum et al. 2016 — Yang, Shishan; Baum, Marcus; Granstrom, Karl, 2016. Metrics for performance evaluation of elliptic extended object tracking methods, *In Proceedings of the 2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI).* New York City, NY: IEEE, pp. 523–528, ISBN 978-1-4673-9708-7, DOI:10.1109/MFI.2016.7849541

Zhang, Chen et al. 2003 — Zhang, Zhihua; Chen, Chibiao; Sun, Jian; Chan, Kap Luk, 2003. EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition* **vol. 36**, pp. 1973–1983, DOI:10.1016/S0031-3203(03)00059-1

In the course of changeable and highly flexible production, automated guided vehicles (AGVs) represent the key component for the realization of highly flexible and scalable intralogistics. In order to meet the high industrial requirements in terms of availability and precision, current solutions for navigation of the AGV usually require additional infrastructure of the environment or are limited to highly structured, predominantly static environments with low dynamics. Thus, they cannot meet the increasing demand for AGVs that operate flexibly and efficiently even in changing and dynamic industrial environments. In the course of this problem, this thesis investigates the use of networking and cloud computing technologies for the specific use case of navigation of mobile robots in an industrial context. The focus of the work within this topic is on the cooperative generation and updating of environment maps, which is a basic requirement for robust and precise localization and efficient path planning of the robots in these environments.

ISBN 978-3-8396-1645-1

9 783839 616451

FRAUNHOFER VERLAG