Environment and Task Modeling of Long-Term-Autonomous Service Robots

Umwelt- und Verhaltensmodellierung langzeitautonomer Serviceroboter

Von der Fakultät für Maschinenbau
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte

Dissertation

von

Marvin Stüde, M. Sc.

2024

1. Referent: Prof. Dr.-Ing. Tobias Ortmaier
2. Referent: Prof. Dr.-Ing. Thomas Seel

Tag der Promotion: 15. Februar 2024

Vorwort und Danksagung

Diese Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mechatronische Systeme der Gottfried Wilhelm Leibniz Universität Hannover entstanden.

Zunächst möchte ich mich bei meinem Doktorvater Professor Tobias Ortmaier für die Möglichkeit bedanken die Promotion unter seiner fachlichen Betreuung durchzuführen. Auch Professor Thomas Seel möchte ich für die Anfertigung des Koreferats und den netten Austausch im Zuge des Promotionsverfahrens danken. Meinen Dank möchte ich außerdem Professor Wurz für die Übernahme des Prüfungsvorsitzes aussprechen.

Ein herzliches Dankeschön richte ich auch an das Dekanat der Fakultät Maschinenbau, insbesondere Sarah Engelmann. Nur durch dein Vorhaben einen Guidingroboter zu entwickeln, war es mir möglich eine tollen Roboter selber zu bauen und im Bereich der Servicerobotik zu forschen. Dabei hatte ich eine breite Unterstützung und möchte an alle beteiligten Studierenden meinen Dank ausrichten, die mich im Rahmen von Abschlussarbeiten, Hiwi-Jobs und als Teil des Design-Teams der Hochschule Hannover unterstützt haben. Besonderer Dank geht dabei an die Studenten (und teilweise mittlerweile langjährigen Kollegen), die gleichzeitig einen erheblichen Teil zu dieser Dissertation beigetragen haben. Namentlich bedanke ich mich bei Timo Lerche, Alexander Petersen, Konrad Westermann, Simon Ehlers und Tim-Lukas Habich.

Ein besonderer Dank geht auch an meine Kolleginnen und Kollegen am imes, die mich während meiner fünfjährigen Institutszeit begleitet haben. Ich habe die Zusammenarbeit, Diskussionen, das gemeinsame Kaffeetrinken und die Nach-Feierabend-Aktivitäten immer sehr genossen und denke gerne daran zurück. Ich wünsche euch sehr, dass sich bald wieder jemand findet, der die Fahne für *Autonome Systeme* in der Robotikgruppe hochhält ;).

Abschließend gilt mein größter Dank meiner Familie und Freunden für den ständigen Rückhalt. Dabei danke ich insbesondere meinen Eltern, Joachim und Angela, die mich immer unterstützt haben und mir überhaupt erst ermöglicht haben zu studieren. Besonders möchte ich auch meiner Partnerin Laura für die ständige Motivation und jahrelange Rücksichtnahme danken. Ohne euch hätte ich es nicht geschafft.

Vielen Dank!

Hannover, Februar 2024 *Marvin Stüde*

Kurzfassung

Der Einsatz von Servicerobotern kann die Effizienz, Produktivität und Sicherheit in Bereichen wie dem Gesundheitswesen, Gastgewerbe oder der Logistik erheblich verbessern. Die Integration der Roboter in komplexe von Menschen genutzte Umgebungen stellt jedoch eine wesentliche Herausforderung für die langfristige Autonomie der Systeme dar. Eine wichtige Lösungsstrategie dahingehend liegt in der Langzeitmodellierung der Umgebungscharakteristika, um eine proaktive Navigation und Aufgabendurchführung zu ermöglichen. So könnten Roboter beispielsweise das Wissen über die langfristige Variabilität menschlicher Aktivitäten dafür nutzen, um stark frequentierte Bereiche während der Navigation zu meiden oder ihre Dienste zu verbessern.

In dieser Arbeit werden Methoden zur Verbesserung der Kartierung, Lokalisierung und Aufgabendurchführung von Servicerobotern im Kontext der Langzeitautonomie vorgestellt, welche diesen Zusammenhang aufgreifen. Hierfür werden insbesondere multimodale Sensorinformationen und (langfristige) Umgebungsmodellierung genutzt. Die gelernte Umgebungsdynamik wird aktiv verwendet, um die Aufgabenerfüllung von Servicerobotern zu verbessern.

Als erster Beitrag wird ein neuer autonomer Langzeit-Serviceroboter vorgestellt, der sowohl innerhalb als auch außerhalb von Gebäuden eingesetzt werden kann. Die multimodalen Sensorinformationen des Roboters bilden die Grundlage für nachfolgende Methoden zur Kartierung und Modellierung menschlicher Aktivitätsmuster.

Es wird gezeigt, dass die Nutzung multimodaler Daten zur Lokalisierung sowie Kartierung die langfristige Robustheit und Qualität der Karten verbessert. Dies gilt insbesondere für Umgebungen unterschiedlicher Charakteristik, wie bspw. gemischte Innen- und Außenbereiche oder klein- und großflächige Gebiete.

Ein weiterer Beitrag ist ein auf Langzeitbeobachtungen menschlicher Aktivitätsmuster basierendes Regressionsmodell, welches eine räumlich-zeitliche Vorhersage dieser Aktivitäten ermöglicht. Das Auftreten von Personen kann somit über lange Zeiträume prädiziert werden und es wird eine proaktive Navigationsplanung ermöglicht.

Die Modellvorhersagen werden anschließend verwendet, um das Verhalten des Roboters umgebungsspezifisch anzupassen. Es wird ein reaktives Aufgabensteuerungssystem eingeführt, das im Falle von Fehlern aktiv Wiederherstellungs- und Suchverhalten auslöst, um die Langzeitautonomie zu verbessern. Softwarefehler können somit gezielt gelöst und potenzielle menschliche Helfer effizient gefunden werden.

Schlagwörter: Langzeitautonomie, simultane Lokalisierung und Kartierung, Umgebungsmodellierung, symbiotische Autonomie, Aufgabensteuerung

Abstract

Utilizing service robots in real-world tasks can significantly improve efficiency, productivity, and safety in various fields such as healthcare, hospitality, and transportation. However, integrating these robots into complex, human-populated environments for continuous use is a significant challenge. A key potential for addressing this challenge lies in long-term modeling capabilities to navigate, understand, and proactively exploit these environments for increased safety and better task performance. For example, robots may use this long-term knowledge of human activity to avoid crowded spaces when navigating or improve their human-centric services.

This thesis proposes comprehensive approaches to improve the mapping, localization, and task fulfillment capabilities of service robots by leveraging multi-modal sensor information and (long-term) environment modeling. Learned environmental dynamics are actively exploited to improve the task performance of service robots.

As a first contribution, a new long-term-autonomous service robot is presented, designed for both inside and outside buildings. The multi-modal sensor information provided by the robot forms the basis for subsequent methods to model human-centric environments and human activity.

It is shown that utilizing multi-modal data for localization and mapping improves long-term robustness and map quality. This especially applies to environments of varying types, i.e., mixed indoor and outdoor or small-scale and large-scale areas.

Another essential contribution is a regression model for spatio-temporal prediction of human activity. The model is based on long-term observations of humans by a mobile robot. It is demonstrated that the proposed model can effectively represent the distribution of detected people resulting from moving robots and enables proactive navigation planning.

Such model predictions are then used to adapt the robot's behavior by synthesizing a modular task control model. A reactive executive system based on behavior trees is introduced, which actively triggers recovery behaviors in the event of faults to improve the long-term autonomy. By explicitly addressing failures of robot software components and more advanced problems, it is shown that errors can be solved and potential human helpers can be found efficiently.

Keywords: long-term autonomy, simultaneous localization and mapping, environment modeling, symbiotic autonomy, task control

Contents

Nomenclature

Symbols used only in single sections are described exclusively in the text.

General conventions

a	scalar or random variable
$\boldsymbol{a}$	vector or vector-valued random variable
$\boldsymbol{A}$	matrix or matrix-valued random variable
$a_{1:n}$	sequence containing exactly n elements, valued as a
$\mathbb{A}$	set

Latin Letters

A_{sbt}	tuple describing a stochastic behavior tree action
c_i	count of people detections in a spatio-temporal grid cell
C_{sbt}	tuple describing a stochastic behavior tree condition
D	robot field of view (detection area)
D_i	detection area at $\boldsymbol{x}_i$
$E_{1:p}$	sequence of p edges
$\exp$	exponential function
$f(\boldsymbol{x})$	$\mathbb{R}^d \to \mathbb{R}$, latent mean function
$g(\boldsymbol{x})$	$\mathbb{R}^d \to \mathbb{R}$, latent variance function
$\mathcal{GP}$	GAUSSIAN process
$k_f(\boldsymbol{x}, \boldsymbol{x}')$	$\mathbb{R}^d \to \mathbb{R}$, covariance function of the GAUSSIAN process prior for the mean
$k_g(\boldsymbol{x}, \boldsymbol{x}')$	$\mathbb{R}^d \to \mathbb{R}$, covariance function of the GAUSSIAN process prior for the variance
$k_{\mathrm{s}}(x, x')$	$\mathbb{R} \to \mathbb{R}$, spatial component of covariance function k_f
$k_{\mathrm{t}}(x, x')$	$\mathbb{R} \to \mathbb{R}$, temporal component of covariance function k_f
$l_{\mathrm{t},i}$	lengthscale hyperparameter of one individual summative component of k_{t}
l_{fail}	expected distance of the search action to fail
l_{sp}	length of the search path
$L_{1:m}$	sequence of m link points
$\log$	natural logarithm (base e)
m_{grid}	number of person occurrence grid elements in horizontal direction
m_y	posterior predictive mean

M	map of the environment
$M_{1:n}$	sequence of n metric maps
n_{pl}	number of sampled places
n_{ver}	number of verification nodes in neighborhood for global loop search
n_{start}	number of successful loop closures before changing to local search
$\mathcal{N}$	normal distribution
$\mathbb{N}$	set of natural numbers
o_{grid}	number of person occurrence grid elements in vertical direction
p_{min}	probability threshold for point cloud classifier
$p_{\mathrm{s}}(t)$	$\mathbb{R} \to \mathbb{R}$, time-dependent probability of an action/condition to succeed
$p_{\mathrm{f}}(t)$	$\mathbb{R} \to \mathbb{R}$, time-dependent probability of an action/condition to fail
$p_{\mathrm{s,T}}(t)$	$\mathbb{R} \to \mathbb{R}$, time-dependent probability of a behavior tree to succeed
p'_{s}	desired confidence value for the time to succeed
$p_{\mathrm{f,T}}(t)$	$\mathbb{R} \to \mathbb{R}$, time-dependent probability of a behavior tree to fail
$\mathbb{P}_i$	point cloud at location $\boldsymbol{x}_i$
$\tilde{\mathbb{P}}_i$	filtered point cloud at location $\boldsymbol{x}_i$
r_{max}	maximum point range for feature computation
r_{ver}	radius of neighborhood for global loop search
r_{min}	minimal radius of the loop search sphere
r_{s}	spatial edge length of square-shaped grid cell
r_D	detection radius of a person detector
$\mathbb{R}$	set of real numbers
$\mathcal{S}_{\mathrm{A},i\to j}$	search action
$S_{i\to j}$	tuple describing a search path
$\mathbb{T}$	set containing all sampled time series
t_{max}	look-ahead time of a discrete-time MARKOV chain
$\bar{v}_{\mathrm{sp}}$	average velocity while searching
$\mathcal{W}_{\mathrm{A},i}$	wait action
y_i	(observed) people rate in a spatio-temporal grid cell
$\boldsymbol{d}_{\mathrm{ped},k}$	spatio-temporal point representing a detected pedestrian
$\boldsymbol{D}_{\mathrm{ped}}$	matrix of 2D people detections taken at different points in time
$\boldsymbol{f}$	distribution over latent mean functions
$\boldsymbol{g}$	distribution over latent variance functions
$\boldsymbol{K}_{\boldsymbol{ff}}$	covariance matrix of the GAUSSIAN process prior for the mean
$\boldsymbol{K}_{\boldsymbol{gg}}$	covariance matrix of the GAUSSIAN process prior for the variance
${}_{(j)}\boldsymbol{m}_i$	normally distributed pose of link point L_i on map M_j
${}_{(j)}\bar{\boldsymbol{m}}_i$	expected value corresponding to ${}_{(j)}\boldsymbol{m}_i$
$\boldsymbol{o}$	candidate periods for FOURIER transform
${}_{(i)}\boldsymbol{p}_j$	normally distributed origin of coordinate frame $(\mathrm{CF})_j$ relative to

	coordinate frame $(\mathrm{CF})_i$
${}_{(i)}\bar{\boldsymbol{p}}_j$	expected value corresponding to ${}_{(i)}\boldsymbol{p}_j$
$\boldsymbol{x}_i$	2D point on a person occurrence map
$\boldsymbol{Q}$	infinitesimal generator matrix of a discrete time MARKOV chain
$\boldsymbol{S}_{f,g}$	covariance matrices of the approximate posterior variational distributions (normal)
${}^{i}\boldsymbol{T}_j$	homogeneous transformation matrix expressing the configuration of coordinate frame $(\mathrm{CF})_j$ relative to coordinate frame $(\mathrm{CF})_i$
$\boldsymbol{u}_{1:K}$	sequence of K control inputs
$\boldsymbol{u}_f$	latent mean function values $f(\boldsymbol{Z})$ at inducing locations
$\boldsymbol{u}_g$	latent variance function values $g(\boldsymbol{Z})$ at inducing locations
$\boldsymbol{x}_{1:K}$	sequence of K locations
$\boldsymbol{X}$	training input dataset
$\boldsymbol{y}$	training observations
$\boldsymbol{z}_{1:K}$	sequence of K measurements
$\boldsymbol{z}_{ij}$	virtual measurement between node i and node j
$\boldsymbol{Z}$	location of inducing inputs

Greek Letters

α_{ind}	ratio the between number of inducing points and data points
α_{min}	minimum ratio between local and global nodes for loop search
β	scale factor for uncertainty-dependent loop search radius
γ_i	period hyperparameter of one individual summative component of k_{t}
Δ_i	observation duration of a spatio-temporal grid cell
$\zeta(x)$	$\mathbb{R} \to \mathbb{R}_+$, link function to guarantee positive values
λ	rate parameter of exponential distribution
λ_{max}	largest eigenvalue of a covariance matrix
Λ	number of sampled time series
μ	expected success rate of an action
ν	expected failure rate of an action
$\nu_f(\boldsymbol{x})$	$\mathbb{R}^d \to \mathbb{R}$, mean function of the GAUSSIAN process prior for the mean
$\nu_g(\boldsymbol{x})$	$\mathbb{R}^d \to \mathbb{R}$, mean function of the GAUSSIAN process prior for the variance
ρ	servicing ratio
σ_y^2	posterior predictive variance
$\sigma_{\mathrm{t},i}^2$	variance hyperparameter of one individual summative component of k_{t}
σ_{max}^2	maximum variance of all summative components of k_{t}
τ	temporal resolution of spatio-temporal grid cell
ψ	number of summative components of k_{t}
ψ_{max}	upper limit for ψ

$\boldsymbol{\mu}_{f,g}$	mean functions of the approximate posterior variational distributions (normal)
${}_{(j)}\boldsymbol{\Sigma}_i$	covariance matrix corresponding to ${}_{(j)}\boldsymbol{m}_i$
$\boldsymbol{\chi}_i$	feature descriptor of point cloud $\mathbb{P}_i$
$\boldsymbol{\chi}_i^{\mathrm{I}}$	feature descriptor containing real values
$\boldsymbol{\chi}_i^{\mathrm{II}}$	feature descriptor containing range histograms
${}_{(j)}^{p}\boldsymbol{\Sigma}_i$	covariance matrix corresponding to ${}_{(i)}\boldsymbol{p}_j$
$\boldsymbol{\Omega}_{ij}$	information matrix of virtual measurement $\boldsymbol{z}_{ij}$

Acronyms

BT	behavior tree
CoPA-Map	continuous pedestrian activity map
DTMC	discrete-time MARKOV chain
FOV	field of view
GP	GAUSSIAN process
GPR	GAUSSIAN process regression
LTA	long-term autonomy
MDP	MARKOV decision process
NUDFT	non-uniform discrete FOURIER transform
PSBT	person search behavior tree
RMSE	root mean square error
ROS	robot operating system
RTAB-Map	real-time appearance-based mapping
SBT	stochastic behavior tree
SLAM	simultaneous localization and mapping

1 Introduction

What once was an idea of science-fiction writers like Isaac Asimov in the mid-20th century is increasingly becoming a reality: robots that operate in everyday environments and perform work for humans. No longer bound to the factory floor, autonomous machines advance into areas shared with people, such as hotels, airports, shopping malls, and, not least, homes. After production activities, the focus of automated applications and intelligent value creation is shifting to *services*.

The International Standardisation Organisation defines the term *service robot* as a “robot in personal or professional use that performs useful tasks for humans or equipment" [Sec21]. Following this definition, a robot generally requires a “degree of autonomy", which is the “ability to perform intended tasks based on current state and sensing, without human intervention.” This human-centric definition contrasts with classical industrial robotics, which focuses on industrial automation applications in controlled environments. There is an increasing trend for robotics applications in the service sector, which is reflected in particular in the development of revenue as depicted in Figure 1.1.

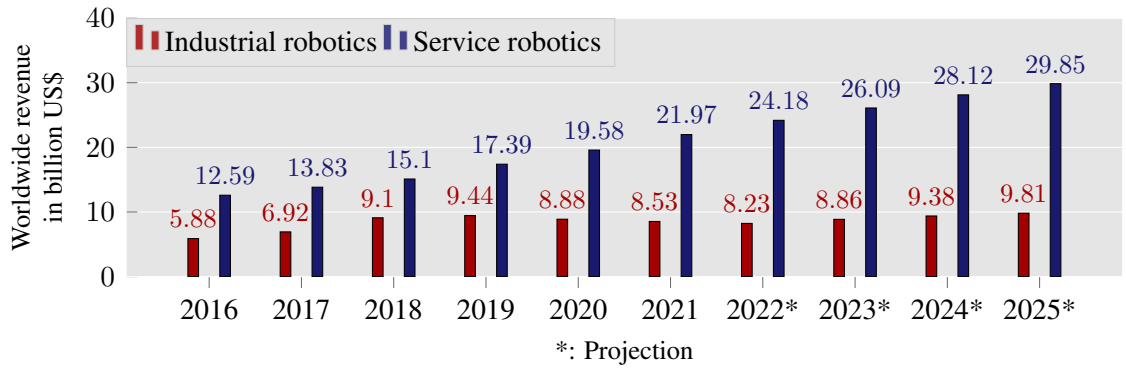


Figure 1.1: Development of worldwide revenue of service robotics compared to industrial robotics over time. [Sta22]

Progress in sensor technology, computing power, and methodologies for robust localization, mapping, and navigation enable deployments of autonomous service robots in a growing number of domains. One of the most visible application areas is the domestic domain, for example, in the form of cleaning robots or autonomous lawnmowers. In clinical environments, service robots can be used for different applications such as disinfection, surgery, logistics, monitoring, or rehabilitation [HKM+21]. In particular, logistics applications can also be found in other areas, such as for room service robots in hotels [MGS18] or autonomous food delivery [VCP21]. The common feature of the above topics is the need for autonomous task execution in human-populated environments.

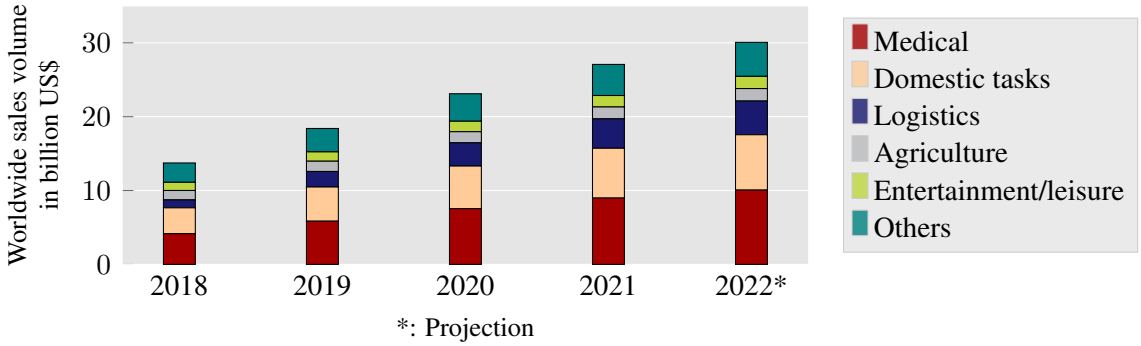


Figure 1.2: Service robotics sales volume by application area. [Sta21]

Applications in these environments represent a significant portion of service robotics, evident from the category-based sales volume in Figure 1.2.

For a lasting successful application in human-centered environments, service robots must demonstrate usefulness and user engagement in a long-term sense. This is a non-trivial problem, as dynamic environments, unpredictable situations, and complex human-robot interaction scenarios significantly complicate long-term deployments. Currently, commercially available robots are developed with a focus on specific applications assuming structured environmental conditions, limiting the robots' actual utility. Technical solutions that might work under laboratory conditions rarely prove robust and reliable in continuous use. The discontinuation of various (initially promising) commercial social service robots [Rob18; Sch19; Reu21][1], as well as retail robots [Nas20], and delivery robots [Con22] underscores this issue. Autonomously acting, flexibly deployable robots that provide real, lasting value are still rare in the real world. Therefore, achieving *long-term autonomy* remains an essential goal in industry and research to steadily expand service robot application areas.

1.1 Long-Term Autonomy in Robotics

Ensuring robot autonomy in real-world scenarios over long periods is a significant challenge. This makes long-term autonomy (LTA) an important research area for robots in various domains, including space, field, roads, or underwater. Opposed to robots in strictly controlled settings (e.g., industrial robots), the requirements for LTA robots are broader regarding their hardware and software, environment, and tasks. Requirements may also change over time, and the changes cannot always be characterized or detected in advance. Robots acting in human-populated environments might, for example, be subject to people moving in their field of view, parked cars changing their positions, or long-term seasonal changes and surface wear. [KHD+18]

[1] As emphasized by commentators, many companies fall into the trap of "selling intentions as results" and "other than a few robot vacuum companies (mainly iRobot), no company has developed a successful home robot." [Van19]

In particular, service robots face several difficulties concerning LTA, briefly summarized in the following. They perform tasks for or alongside humans in environments not explicitly customized for them. These environments are spatially and temporally *dynamic* as people and obstacles move or day-night changes occur. Dynamics lead to challenges in mapping and self-localization, as well as in task planning and control. Similar to dynamics, environments can be *heterogeneous*, i.e., of different types. Examples include areas consisting of narrow aisles and large open areas (e.g., in shopping malls or airports) or mixed indoor and outdoor environments (e.g., campus-like areas). Parts of the environment and its states may also not be fully known before deployment and could not be completely traversable due to impassable objects (e.g., newly occurring obstacles or doors that are opened or closed). Dealing with this *open world* requires reactive problem-solving and potential changes in task specifications [KHD+18]. Therefore, the robot's deliberation functions should allow it to plan tasks, act according to the plan, refine actions into closed-loop commands, and react to events [IG17]. Adapting to the environment's particulars and continuous learning is essential.

Several research initiatives have investigated LTA operations of service robots in offices [MMWG11; BV16; HBJ+17], stores [GBS+09], hotels [PMF+16], museums [BCF+98; DBH19] or care contexts [TVNB17]. The STRANDS [HBJ+17] and CoBot [BV16] projects focused on long-term mapping, navigation, and human-robot interaction (HRI) for LTA applications. Their robots reached multiple weeks of uninterrupted autonomy or cumulative traveled distances of more than 1,000 km, respectively. Other LTA projects focus on types of interaction, for example, by voice and facial recognition [Jon+18; WC18], specific environments (e.g., outdoors [CTW+08] or retirement homes [LMR+19]), or perception and social navigation [Tri+16]. Research is often based on commercially available robots, usually supplemented by additional hardware components. Many of those robots are designed for either exclusive indoor [BV16; HBJ+17; DBH19; PMF+16; MMWG11; WC18; LMR+19], or outdoor use [CTW+08].

Current Challenges

In addition to suitable hardware setups, appropriate mapping and localization methods represent a current research challenge for heterogeneous and dynamic environments. Few simultaneous localization and mapping (SLAM) approaches are suitable for long-term applications on real-world service robots while maintaining desirable properties such as bounded computational complexity, navigable maps, and multi-session operation.

Compared to autonomous mapping for self-localization and navigation, mapping other environmental dynamics has taken a low priority in the context of LTA. Although human movements and actions primarily cause the dynamics in the environment of service robots, long-term modeling of these effects has been largely neglected. However, predicting human activity in the form of movements and presence can improve navigation, task planning, and acceptance of service robots in general. In a long-term context, logistics robots might then plan routes along areas with fewer

people to increase efficiency and safety, or cleaning robots could actively avoid disturbing people. As also described in the survey [KHD+18], one major future challenge for robots lies in human-in-the-loop systems that “leverage human knowledge in unforeseen situations within long-term scenarios". The robot’s task execution could be modified based on additional information provided by humans when situations arise that were not anticipated at development time. When combining this principle with models of human activity, opportunities arise to increase long-term applications by anticipating human help in dynamic and heterogeneous environments.

Resultingly, this work aims to improve the LTA capabilities of mobile service robots by methods of environment and task modeling. Given the typical case of a service robot with multi-modal sensors, models are created that encode the heterogeneity and dynamics of the environment. Model knowledge of these dynamics is then used to adapt the robot’s task execution to mitigate failures actively, i.e., by improving the search for potential human helpers for problem-solving. Therefore, the overarching structure of this work can be seen as a classic sense-think-act loop, where the observation of dynamics results in an adjustment of robot behavior.

1.2 Research Questions and Contributions

In the following, the research questions of this thesis are formulated, and the main contributions are presented. The main goal is to introduce environment models suitable for long-term-autonomous applications in dynamic and heterogeneous environments and subsequent task synthesis. The requirement for a known robot pose in these environments leads to the first research question:

Research Question 1. *How can the robustness of current SLAM methods be improved in heterogeneous and dynamic environments?*

Methods are proposed to improve localization accuracy, robustness, and map generation for these dynamic and heterogeneous environments. The methods are based on combining different sensor modalities and exploiting environment-specific characteristics for algorithm parameterization. The aim is to extend widely used SLAM methods to increase robustness and other criteria concerning long-term autonomy. Given a sufficiently accurate self-localization and map, sensory detections of people made by the robot can then be used for environment modeling, which leads to the next research question:

Research Question 2. *How can human activity be spatio-temporally modeled based on long-term observations by mobile robots?*

A method for the spatio-temporal prediction of human presence given data collected by a mobile robot is introduced. Utilizing current non-parametric regression and variational inference approaches, the model can process multi-week data to create continuous prediction courses. By incorporating BAYESIAN stochastics, the model is capable of indicating predictive uncertainty.

From this, conclusions can be drawn where and when the model should be extended, which is particularly relevant for non-stationary robots. Finally, the model knowledge of human activity is used to adapt and optimize the task execution of robotic systems, which leads to the last research question:

Research Question 3. *How can models of human activity be integrated with autonomous executive systems and used to improve robots' long-term capabilities?*

Task execution systems for LTA robots are defined and specifically adapted based on environmental knowledge. A task control system is introduced that aligns with requirements resulting from LTA, such as modularity, reactiveness, and expandability. The system includes multi-stepped problem-solving procedures to actively involve human help. Model knowledge of human activity is incorporated by stochastic action primitives for efficient person search to maintain a readable and expandable structure. All introduced methods are evaluated on real-world datasets. Great parts of the evaluation are based on a social service robot system developed explicitly for LTA applications.

The main contributions of this work are outlined as follows, with corresponding publications:

- A novel service robot for LTA applications in heterogeneous (i.e., mixed indoor/outdoor) environments is introduced. Opposed to commonly utilized commercial robots, all software and hardware specifications are available as open source and published under a free license (chapter 2, [SWSS21]).
- A widely used SLAM method is extended to utilize multi-modal sensor data for mapping and localization in large-scale environments. Additionally, a map-management system is proposed in order to improve map quality and computational complexity in heterogeneous environments (chapter 3, [HSLS21; ESNO20]).
- Long-term prediction of human presence is realized by a novel regression model that actively compensates for effects that result from a moving robot (chapter 4, [SS22]).
- A control system for LTA tasks is presented that is partially synthesized from environment information directly at the task-description level (chapter 5, [SWSS21; SLPS21]).

1.3 Thesis Structure

This thesis is organized into one implementation-oriented chapter introducing the robot setup, three method-oriented main chapters, and two accompanying chapters giving an introduction and conclusion to the thesis. Figure 1.3 visualizes the structure and links between the different chapters.

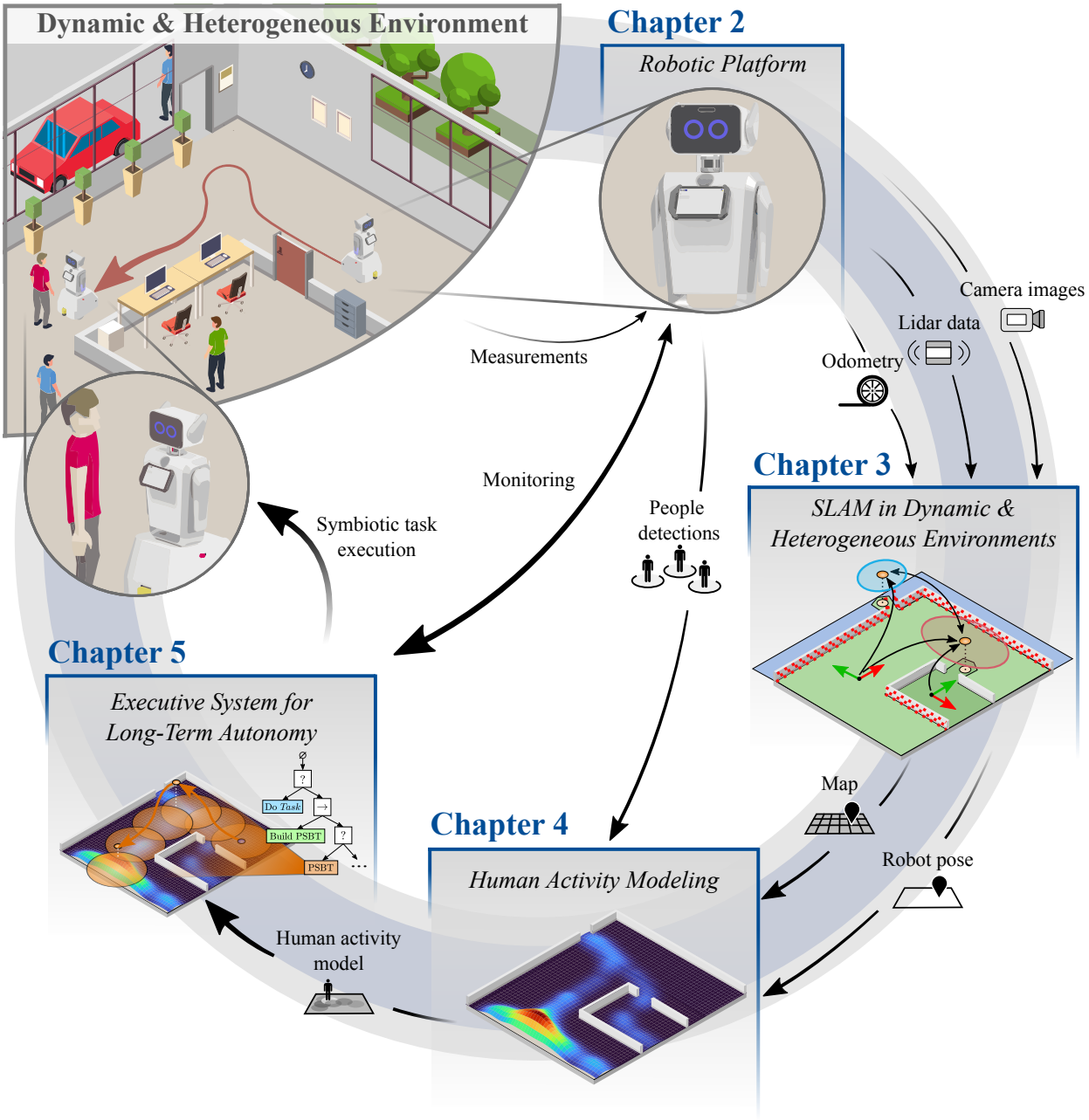


Figure 1.3: Graphical abstract showing the contents of the thesis.

Chapter 2 introduces the social service robot *Sobi*, developed for long-term-autonomous use in indoor and partially autonomous use in outdoor environments. An overview of Sobi's main hardware components, software modules, and communication structure is given (section 2.1). Since the robot serves as a test rig for the developed methods of this thesis, an evaluation of the field of view and accuracy of the robot's people detection modules is described (section 2.2).

Chapter 3 starts with an introduction to the general SLAM problem and related work in this field (section 3.1). A method is presented that incorporates laser-scan-based loop closures into graph-based SLAM approaches (section 3.2) to improve robustness and accuracy. The method

relates to the typical use case of robots equipped with multi-modal sensors. The idea of using this multi-modal information selectively is then extended in the form of a map-management system that uses environment-specific parameterizations for automated mapping (section 3.3). Both methods are evaluated using real-world datasets of public indoor, outdoor, and combined environments.

Chapter 4 introduces continuous pedestrian activity map (CoPA-Map), a method for long-term modeling of spatio-temporal human activity. After an overview of related work on spatial and temporal modeling in section 4.1, the method is described in section 4.2. The problem is addressed via variational GAUSSIAN process regression (GPR), where a significant part is the definition of prior information, particularly via frequency-based data preprocessing. In section 4.3 CoPA-Map is evaluated on real-world datasets. In this regard, an emphasis is placed on application-oriented evaluation to determine the model's usefulness for path planning.

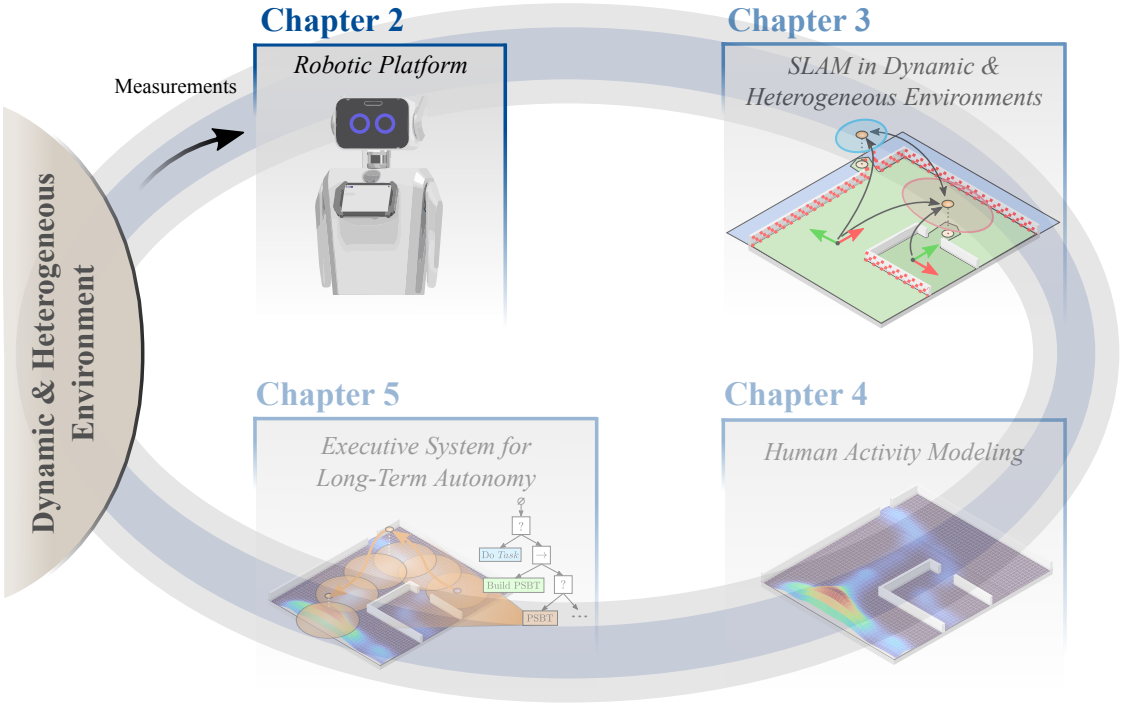
Chapter 5 presents an executive system for task control and monitoring of long-term-autonomous service robots. Starting with section 5.1 on related work, motivation is given to the utilized method *behavior trees*, which are further introduced in section 5.2. Subsequently, the execution system is described and evaluated for general use in LTA applications in section 5.3. In section 5.4, it is then extended to utilize environment knowledge given by a human activity model to realize an efficient search for potential helpers in case of problems.

Finally, in chapter 6, the thesis is summarized and discussed critically. Limitations of the proposed methods are described, and an outlook on possible future work is given.

Programming code for all methods presented in this thesis is available on the author's GitHub page[2] and as part of the open-source project *Sobi*[3].

[2] `https://github.com/MarvinStuede`
[3] `https://marvinstuede.github.io/Sobi`

2 Mobile Service Robot for Long-Term Autonomy



This chapter introduces a novel robot for autonomous and semi-autonomous service tasks. The presented robot *Sobi* is designed for direct user interaction as part of info-terminal services and guiding applications. It is aimed at long-term-autonomous use in heterogeneous open spaces, multiple days and weeks indoors, and frequent partially supervised deployments outdoors. Contrary to other LTA robots, not only the complete robot's software is provided as open-source, but the system is also available as an open-source hardware project, including 3D models and design data, mechanical and electrical drawings and parts lists[1].

The robot's design and components are presented in section 2.1. After introducing the application requirements, the consequently selected hardware components are described. Subsequently, the software structure and its most important modules are presented, with the main focus being the person detection method. This method forms the basis of models presented in later chapters. Consequently, a targeted evaluation of its detection area and accuracy is carried out in section 2.2.

The contents of this chapter were partially published at the peer-reviewed "European Conference on Mobile Robots" (ECMR) 2021 [SWSS21]. Several components of the robot's guiding interface are

[1] `https://marvinstuede.github.io/Sobi`

described in work published at the peer-reviewed “International Conference on Control, Automation and Systems" (ICCAS) 2019 [SWTO19].

2.1 Robot Design and Components

As a social robot for info-terminal and guiding services, Sobi must answer voice- and touch-based requests, e.g., directions, room plans, canteen menus, or small talk. It is supposed to operate in various buildings and the outdoor area of the Faculty of Mechanical Engineering campus of Leibniz University Hannover. This scenario requires a fully autonomous operation inside the robot’s home building and a partially autonomous operation in other areas. Thus, the application area represents a heterogeneous and dynamic environment. The following application demands and component design are based on these conditions.

2.1.1 Application Demands and Design

Heterogeneous environments pose specific challenges due to their spatial structure and inhibited dynamics. These challenges must be addressed by the robot’s design, selection of components, and software structure.

The following requirements on *mechanical and electrical design* are integrated:

- For mixed indoor and outdoor use cases, the robot should have essential splash water protection to withstand spills or brief drizzles.
- The exterior design should not only protect the robot’s components but also look approachable and appealing due to its use case as a social robot.
- The kinematic and mechanic wheel structure should allow for applications on typical indoor and outdoor surfaces (e.g., cobblestones, asphalt, concrete).
- To enable continuous long-term task execution and data collection, the battery should allow for operation for multiple hours.

Based on these goals, the shape and color concept of the robot was developed in close cooperation with the Hanover University of Applied Science and Arts.

Further requirements are implemented regarding the *sensoric setup*:

- Navigation in narrow spaces and long periods of dead reckoning (e.g., hallways) requires good odometry.
- The sensors should allow for robust long-term mapping and localization under environmental variations such as light changes, varying surface materials (e.g., glass and concrete), and wide or narrow spaces.
- Due to frequent contact with humans, reliable person detection and tracking in a large area around the robot should be possible.

Specific requirements for human-robot interaction that influenced the robot's design are not part of this thesis but are described in [SWTO19; SWSS21]. As a result of all mentioned requirements, Sobi is designed for mixed indoor and outdoor usage. This distinguishes the robot from state-of-the-art systems, which are primarily designed for either exclusive indoor [BV16; HBJ+17; DBH19; PMF+16; MMWG11; WC18; LMR+19] or outdoor use [CTW+08]. Their used sensors and actuators would make them unsuitable for the respective other area.

2.1.2 Hardware Components

The selection of hardware components is based on the application requirements presented. The robot's exterior design aims at a futuristic appearance with simple geometric bodies to have an inviting effect on users, avoid water build-up, and allow for easy sealing of transition points. Flat elements are laser cut from aluminum and ABS plastic. The outer covers of the robot, except for the base cover, are laser sintered and coated to repel water. All components mentioned and the robot's exterior design are shown in Fig. 2.1.

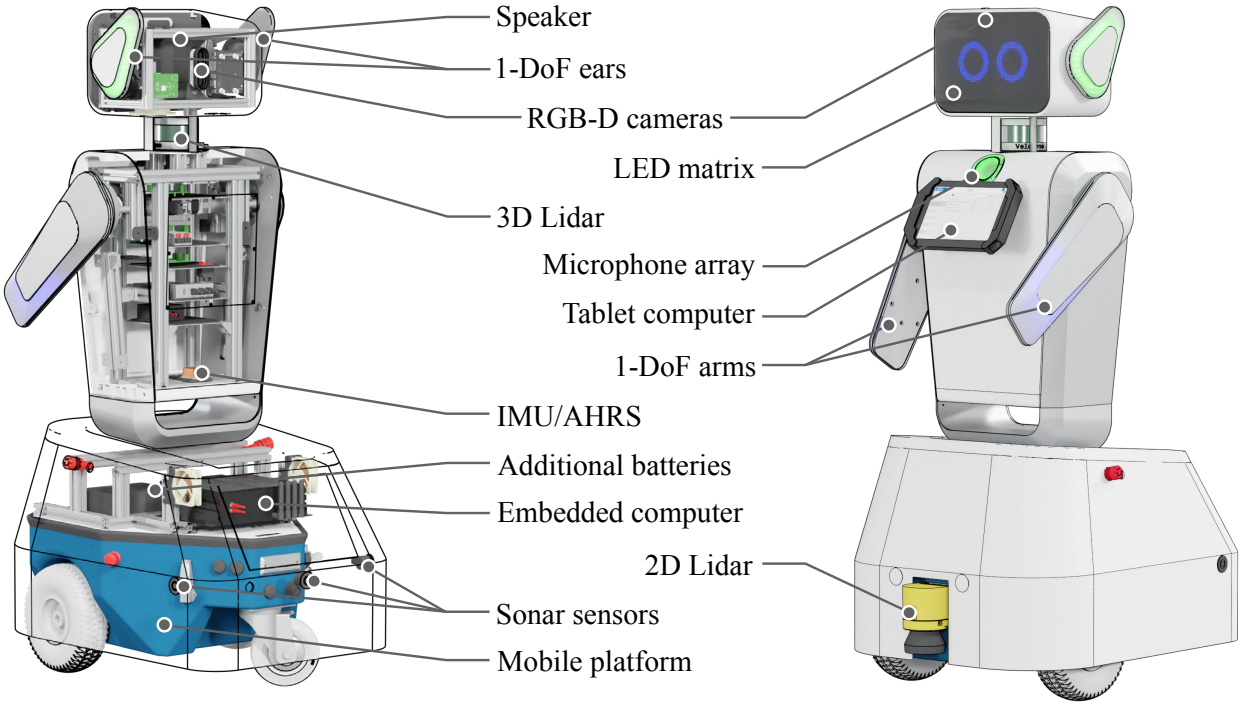


Figure 2.1: Design of the robot *Sobi* with a description of its components (left: rear, right: front view)

The **wheeled platform** (MP-500, Neobotix GmbH) has a differential drive with one caster wheel as the kinematic structure. Due to the relatively large front wheel diameter of 260 mm, the robot can move on uneven surfaces such as paving stones. Additional components of the platform are an onboard computer and a **2D Light detection and ranging (Lidar)** sensor, which can trigger low-level emergency stops for collision avoidance. It includes two 12 V absorbent-glass-mat (AGM)

batteries in series with a total capacity of 50 Ah, retrospectively extended by two smaller batteries for a total capacity of 75 Ah. This capacity gives a minimum time of four to five hours until a recharge is required for heavy usage of the complete robotic system and over twelve hours of standby time.

As a **main computer** (Vecow EVS-1010, Intel i7-7700T, 16 GB RAM, GeForce GTX1050 GPU) serves an embedded system with wide-range voltage input, operated with battery voltage. It has two built-in Wi-Fi modules to permanently connect to the internet and provide a Wi-Fi hotspot for external access in field operation.

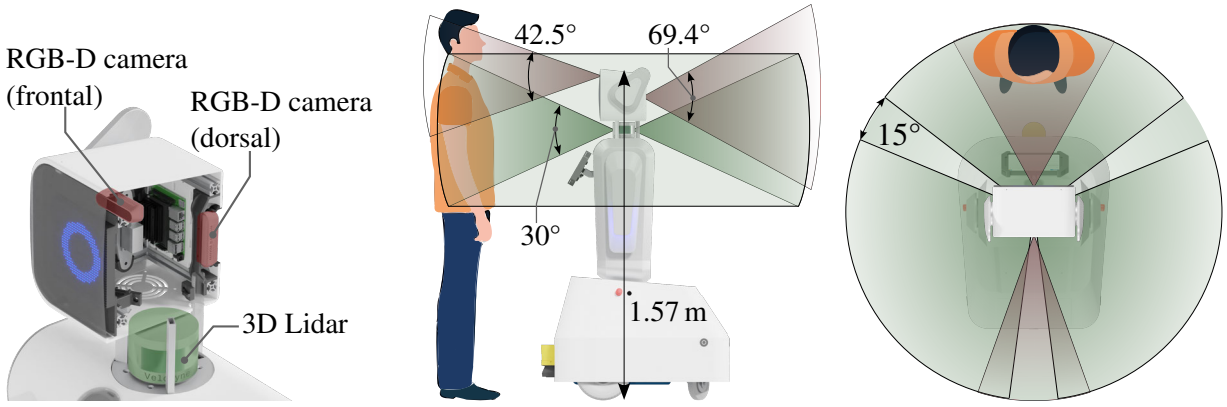
The robot includes the following sensors for localization, navigation, and interaction:

- An **attitude heading reference system (AHRS)** (MTi-30, Xsens), which includes an inertial measurement unit (IMU) with built-in KALMAN filtering for pose estimation. The sensor is mainly used to improve odometry calculation.
- **Sonar Sensors** (Parkpilot URF7, Bosch, 3×) for collision avoidance
- **Microphone array** (ReSpeaker v2.0, SEEED) for speech recognition
- **3D Lidar** (Puck, Velodyne) for localization, people perception, and collision avoidance
- Two **red green blue depth (RGB-D) cameras** (RealSense D435, Intel), which are mounted facing frontally and dorsally and used for localization and people perception. The cameras are connected to an **Nvidia Jetson Nano** single board computer inside the robot's head, which provides the synchronized image data via network and thus reduces the load on the main computer.

The only actuators beside the platform are brushless DC motors for arm movement and servo motors for the ears. Like most state-of-the-art social service robots [HBJ+17; BV16; PMF+16; Tri+16; LMR+19], Sobi lacks physical manipulators, which poses a disadvantage regarding long-term autonomy. Methods to overcome this limitation make up a significant part of a later chapter of this thesis (cf. section 5.4).

Long-term applications place variable demands on a robotic system, such as varying lighting conditions or environments of different sizes and shapes. Therefore, a 3D Lidar sensor and two RGB-D cameras are used as perceptual hardware to compensate for respective disadvantages. The arrangement of the sensors covers a large field of view (FOV), as shown in Figure 2.2.

One camera covers the dorsal area of the robot since three support rods for mounting the head structure restrict the FOV of the Lidar sensor in this area. However, due to beam divergence and multiple reflections, this occlusion is only effective for the close range of the robot. As the coverage area of the frontal camera overlaps with that of the Lidar sensor, redundant information can be taken into account for person detection or self-localization.



(a) Perceptive sensors (b) Sensor FOV side view (c) Sensor FOV top view

Figure 2.2: Arrangement of Lidar and camera sensors (a) covering the area around the robot (b). The Lidar sensor is obstructed by three support rods, resulting in a partial occlusion shown in (c).

Hardware Communication Structure

To ensure a modular functional structure, all four computers in Sobi run Linux (Ubuntu) and the robot operating system (ROS) as a framework for communication and control. Sensors and controllers are mainly connected via universal serial bus (USB) and Ethernet. System clock times of all participants of the ROS network are synchronized via the network time protocol (NTP). Since all applications with hard real-time requirements (i.e., motor controllers and sensor interfaces) use separate control hardware, this approach is sufficient to ensure measurement data synchronization. Once the ROS network is active, defined sensor and processing data is continuously stored on an external server running a MongoDB instance. An overview of the sensors, processing, and output hardware and their essential communication is shown in Figure 2.3.

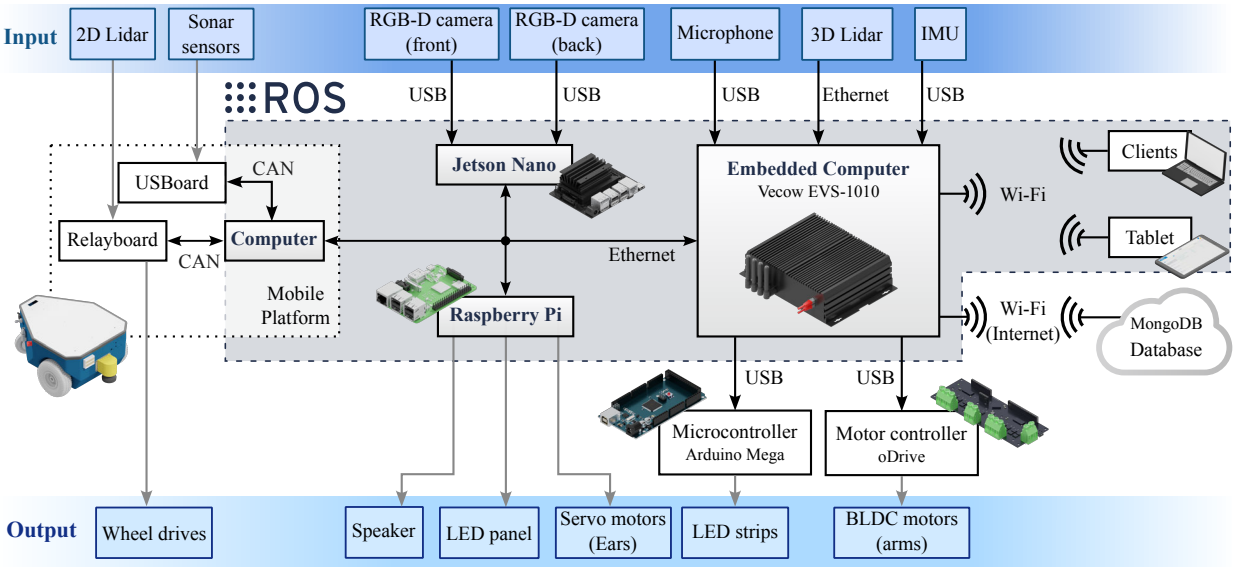


Figure 2.3: Connections and communication between **computers** and hardware components.

2.1.3 Software Architecture

Since the robot mainly relies on ROS for process communication, a modular software structure is used. The various modules are arranged hierarchically and are based on self-developed solutions and freely available ROS packages, partially adapted for the specific robot setup. Figure 2.4 shows the layered architecture.

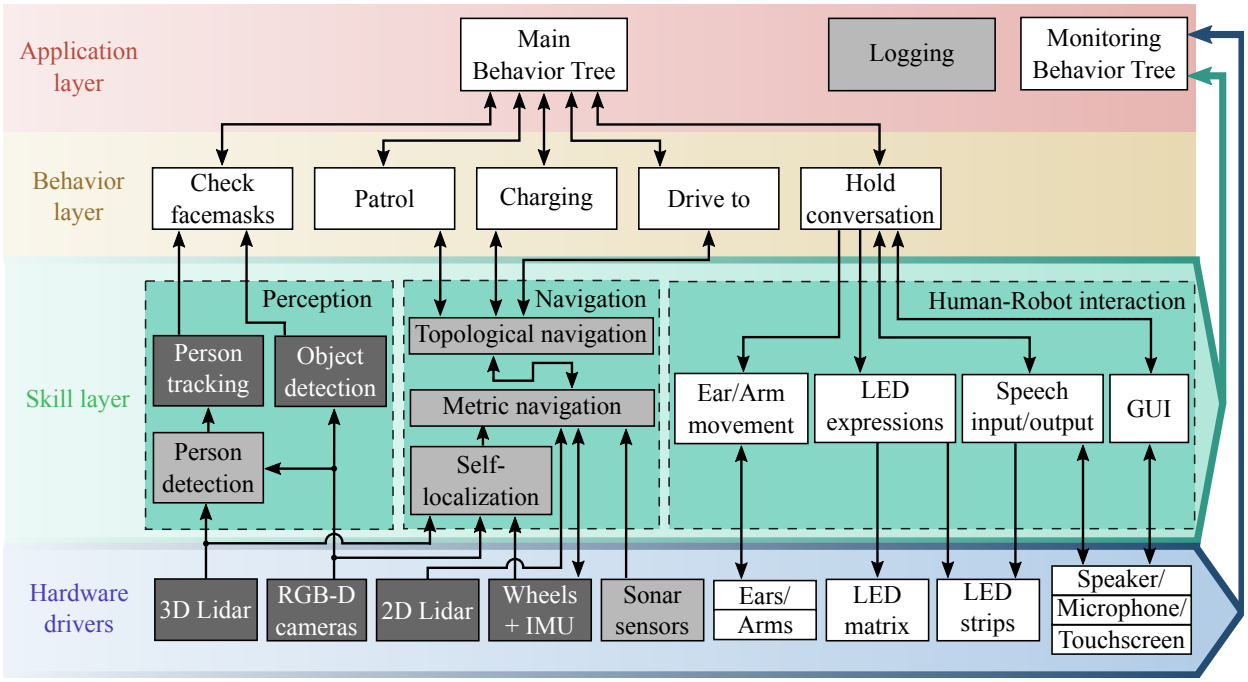


Figure 2.4: The layered software structure of the robot. White boxes indicate self-developed programs; dark grey boxes indicate third-party ROS programs, and light gray boxes indicate modified or extended third-party ROS programs. ([SWSS21] ©2021 IEEE)

Components in the *hardware driver* layer provide raw sensor data or receive low-level controls. This data is processed or sent by specific programs (*skills*), realizing the robot's basic functionality, such as self-localization, person detection, or speech output. Different skills are combined into *behaviors* that accomplish goal-directed tasks. The autonomy and tangible utility of the service robot is generated via *applications*, which consist of patrolling and information scenarios. The central control unit in the application layer is a behavior tree (BT), partly composed of sub-trees of the behavior layer. Since the robot must act purely reactively on immediate requests, modeling via BTs is well-suited, and using a task scheduler is unnecessary. A concise introduction on BTs is given in section 5.2.

People and Object Perception

In service robotics, detecting humans is particularly important for domestic and public environments and an essential aspect of safe and valuable robot behaviors. Therefore, Sobi includes a people

detection and tracking pipeline, allowing for the perception of humans in an area surrounding the robot. People detection is conducted multi-modally by using both RGB-D cameras in conjunction with the 3D Lidar sensor. Detection in the RGB image has the advantage that it functions robustly even with partial occlusion. A disadvantage is the dependence on ambient light and the limited range of active stereo-based depth detection technology [HSKV19].

In contrast, a Lidar sensor provides long-range and wide-angle measurements, which are usually very accurate and not affected by lighting conditions [YDB17]. Combining the different sensor modalities enables robust tracking in a large field of view. The people detection and tracking pipeline consists of image-based and range-based detectors whose outputs are fused and then combined with a motion model to obtain persistent people tracks throughout multiple consecutive measurements. A schematic representation of the pipeline is shown in Figure 2.5.

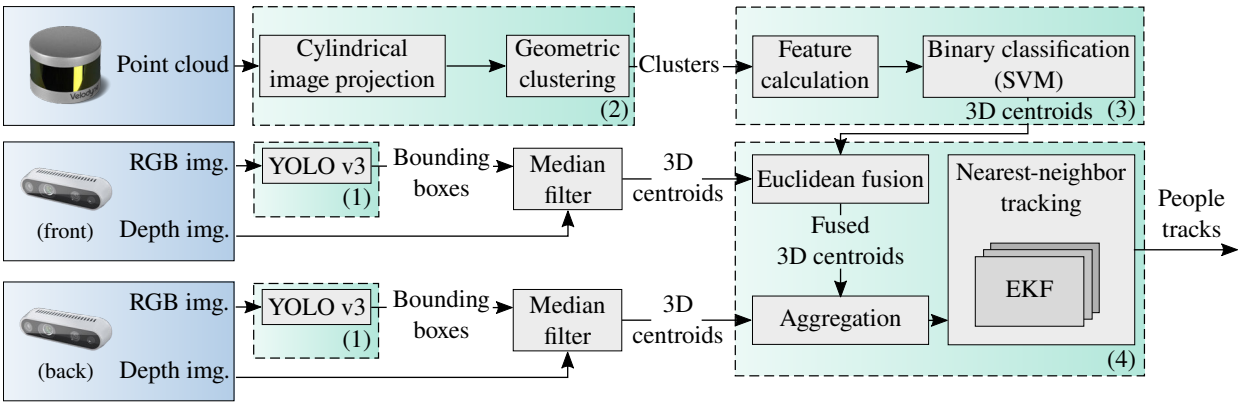


Figure 2.5: Person detection and tracking pipeline.

Detection in the RGB image is performed with the YOLOv3 system (1), which applies a single neural network to the whole input image and is orders of magnitudes faster than other state-of-the-art algorithms. This enables object detection in real-time while providing sufficient accuracy for the given application [RF18]. The authors provide network weights trained on the large-scale object recognition dataset COCO, which is commonly used and contains 80 object classes, including a class for *person* [LMB+14]. A median filter obtains the centroid of the detected object in three-dimensional space within the shrunk bounding box on the depth data of the respective camera.

People detection based on the Lidar data is a two-step process. The first step is clustering the depth data, and the subsequent second step determines which clusters correspond to people. For clustering (2), the approach presented in [BS16] is used, providing reliable segmentation and having small computational demands, which is achieved by converting the point cloud to a cylindrical range image. To determine which clusters correspond to people, an offline-trained support vector machine (SVM) performs binary feature-based classification (3). In total, five features are used, a subset of the features described in [NMH10] and [KMW+11]. The specific combination of these features was proposed in [YDB17], aiming at decreasing the computational load while maintaining

an almost similar classification performance. The features, e.g., describe the number of points in the cluster, distance to the sensor, or the moment of the cluster's inertia tensor.

Both the RGB and Lidar-based detectors operate in real-time with the frequency of incoming measurements (30 Hz and 7 Hz, respectively). Enabling the robot to re-identify the same person over multiple consecutive detections requires fusing the detections with a motion model. Fusion has the additional advantage that tracks can also be maintained if a temporary occlusion of the person occurs. The SPENCER tracking framework [LBLA16] is utilized for this task, which can carry out the consecutive steps of fusion, aggregation, track estimation, and association (4). Fusion is performed for the sensors with overlapping detection areas (frontal camera and 3D Lidar sensor, cf. Figure 2.2) by taking the weighted average pairwise between the detections with minimum EUCLIDEAN distance. The field of view of the dorsal camera and the 3D Lidar sensor are not overlapping since the rear support rod carrying the robot's head covers the scanner's field of view. Therefore, the corresponding measurements of the rear camera are aggregated with (i.e., appended to) the fusioned data of the frontal and lateral areas. The motion of the target is then predicted by maintaining an extendend KALMAN filter (EKF) for each person, where the associated model is a constant velocity model with additive process noise [LGA15]. New incoming data is associated with existing tracks by a global nearest-neighbor association based on the MAHALANOBIS distance. The outputs of the complete pipeline are tracked people measurements, each consisting of a 2D point with timestamp $\boldsymbol{d}_{\mathrm{ped},k} = (x_{1,k}, x_{2,k}, t_k)^{\mathrm{T}}$, an ID $i_{\mathrm{ped},k} \in \mathbb{N}$, and current velocity $\boldsymbol{v}_{\mathrm{ped},k} \in \mathbb{R}^2$.

Localization and Navigation

Determining the mobile robot's pose in the environment is a fundamental but challenging problem, especially if the environment changes dynamically. Since Sobi is used in heterogeneous settings, both inside and outside buildings, a suitable SLAM solution is required. In order to utilize the multi-modal sensory system consisting of RGB-D and Lidar sensors, the SLAM framework real-time appearance-based mapping (RTAB-Map) [LM19] is used (introduced in detail in section 3.1.2). Methods to improve mapping and localization specifically in heterogeneous and dynamic environments for long-term autonomy are presented in this thesis's upcoming chapter 3.

The input of the SLAM front-end is wheel odometry, fused with inertial measurements via an EKF. This approach enables locally robust odometry estimation, which is particularly beneficial when moving in confined areas and feature-poor environments. Besides the estimated robot pose, another output of the SLAM system is a globally referenced metric 2D occupancy grid map created from the internal 3D representation via projection onto the ground plane. Navigation on this map, respectively local and global path planning, is then implemented with standard planners of the ROS framework (i.e., dynamic window approach [FBT97] and A* graph search [HNR68]). Above the metric level, a topological map is created to account for the semantic structure of the environment. Nodes represent relevant locations, such as specific rooms or facilities, or waypoints between which the robot can navigate directly or move using problem-specific planners. Whereas

regular movement between nodes is executed with metric navigation, specific nodes require custom approaching behaviors, such as the charging station. Docking the charging station is based on a triangular landmark that the 2D laser scanner can detect. A Dubins path consisting of circular and linear segments is planned and subsequently followed with a pure pursuit controller [Cou92]. An advantage of this approach is low computational complexity and robust results, even when deviations in the initial starting position occur [Har20].

Human-Robot Interaction

Sobi's main task is to provide environment-specific information and guiding applications. The human-robot interface consists of speech processing and touch-based operation with a graphical user interface (GUI) to provide intuitive accessibility (see Figure 2.6). The system's current status is additionally indicated by the robot's LED lights and ear movement (see Figure 2.7). Speech processing is realized by a combination of Google's Speech-to-Text and Text-to-Speech services and the natural language processing pipeline Dialogflow [SWTO19]. All information can be accessed via speech and touch interfaces, which include the following functions: queries and display of a canteen menu, public transport timetables, staff offices, and room locations, as well as options for small talk.

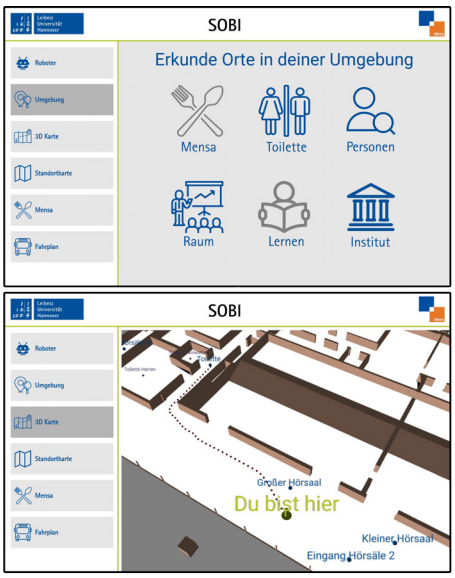
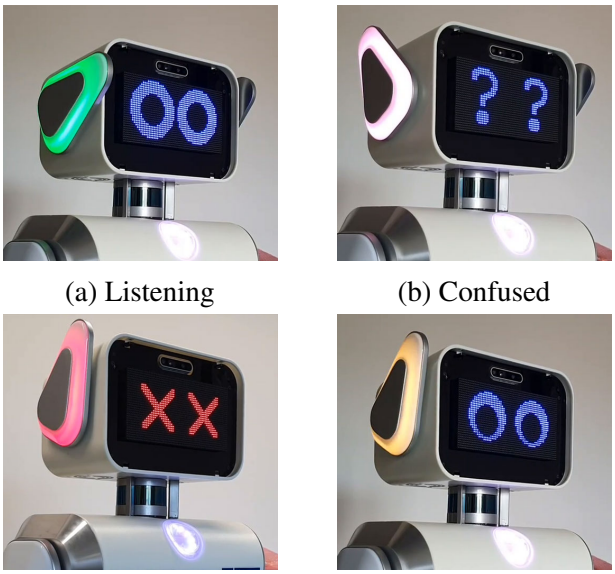


Figure 2.6: GUI of the tablet. ([SWSS21] ©2021 IEEE)



(a) Listening (b) Confused

(c) Error (d) Processing

Figure 2.7: LEDs and ears indicate the system status. [Wes20]

2.2 Results

This section contains a validation of the robot's perceptive components required for people-centered experiments of the following chapters. The focus is on the evaluation of person detection and tracking, as later chapters contain results on mapping and localization (cf. chapter 3) as well as task control and long-term autonomy (cf. chapter 5).

The 3D Lidar sensor, RGB-D cameras, and IMU are calibrated extrinsically [RNS+16] and are referenced to the mobile platform based on CAD data. The training dataset for the Lidar-based binary classifier consists of 2,000 clusters per class (*human* or *non-human*), resulting in a 1 to 1 ratio of positive and negative samples. Data collection is carried out in various areas inside and outside of buildings, in particular in crowded areas (for example, a canteen) [Pet19]. Training is then conducted via 10-fold cross validation, resulting in 98.5 % accuracy.

Due to the different viewing areas of the sensors and partial occlusion, a spatially-dependent people tracking quality can be assumed. An accuracy and precision evaluation is performed to verify in which area the pipeline can reliably track people. Measures commonly used for this purpose are the CLEAR-MOT metrics [BS08], which aim at evaluating multi-object tracking performance and define an aggregate error measure called multi-object tracking accuracy (MOTA) as

$$\mathrm{MOTA} = 1 - \frac{\sum_t \left(\mathrm{M}_t + \mathrm{FP}_t + \mathrm{MME}_t\right)}{\sum_t \mathrm{GT}_t}, \tag{2.1}$$

where M_t, FP_t and MME_t are the number of misses, false positives, and mismatches, respectively, for time t. The optimal MOTA value is 1.0, and the score can also reach negative values if the sum of errors is larger than the number of ground truth objects GT. The ground truth is obtained by an optical localization system consisting of six cameras and infrared light sources (Oqus 4, Qualisys GmbH). For pose acquisition, reflective markers are placed in unique arrangements on the head of one to two people and centrally above the robotic system. Due to the limited field of view of the external cameras, a maximum distance of 4.5 m between marker arrangements can be used in the present measurement setup. A total of 16 measurement runs are performed. As part of these runs, a person moves towards the robot, moves laterally to the robot, follows a defined path, and performs random movements with a second person [Ben21]. Figure 2.8 and Table 2.1 show the results of the minimum achieved MOTA metric, which is given for the different fields of view of the respective sensors. As a measure of precision, the root mean square error (RMSE) is also given. The MOTA values are similar to the values reported in the literature [LGA15; LBLA16]. With regard to the areas marked in green in Figure 2.8, it can be concluded that person tracking is possible according to the state-of-the-art. This enables robust tracking in an approximately circular area with a radius of ca. 4.4 m around the robot.

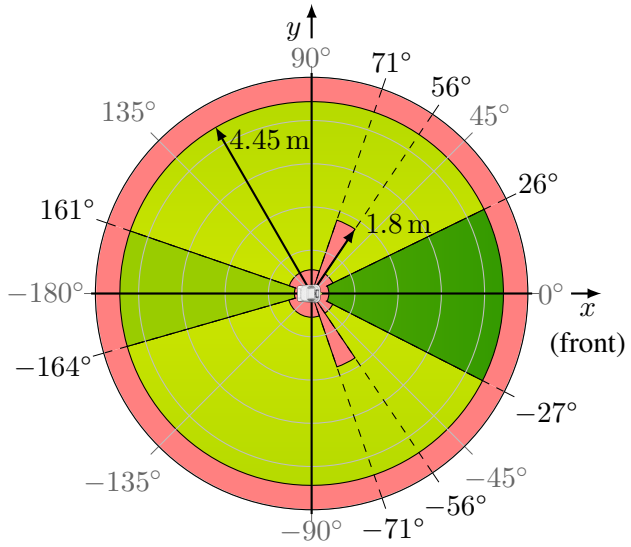


Figure 2.8: Spatially dependent error of the people tracking pipeline (top-down view).

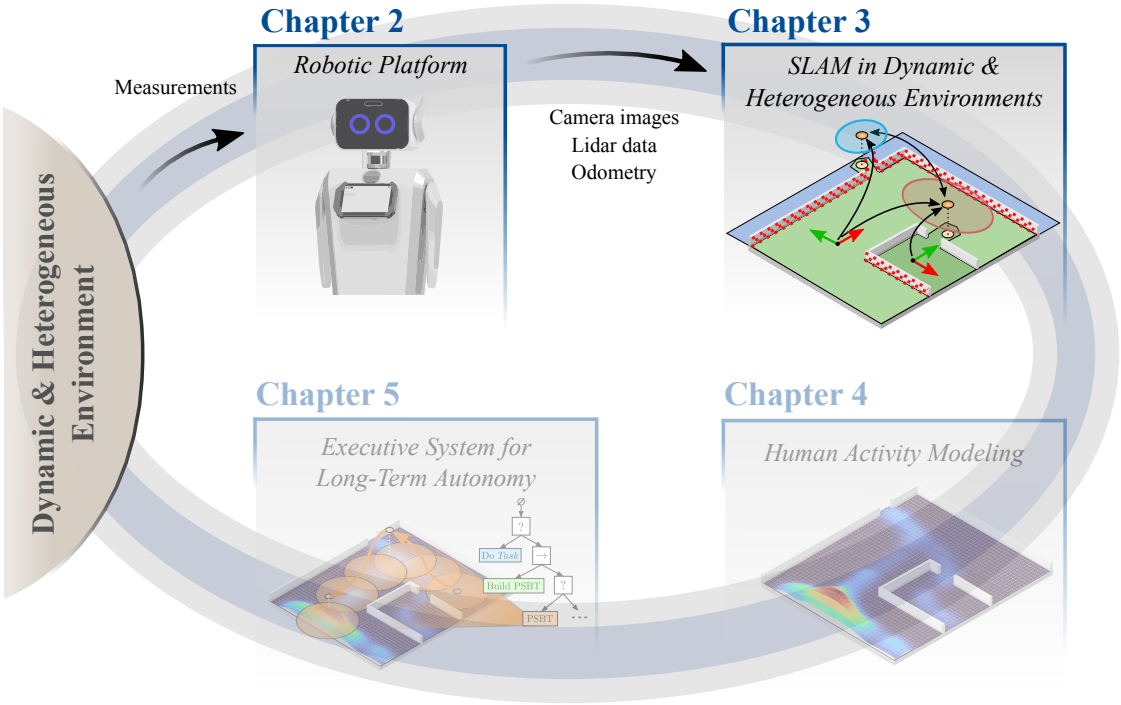
Area	MOTA (min.)	FP%	M%	RMSE in m
	89.7 %	2.1 %	3.3 %	0.2
	84.3 %	4.6 %	6.6 %	0.25
	70.1 %	17.2 %	8.6 %	0.21
	No detections			

Table 2.1: Accuracy and precision metrics of the different segments.

2.3 Conclusion

This chapter introduces Sobi, a social service robot developed for information and guiding applications. Besides a friendly-looking design for human-robot interaction [SWSS21], the robot features hardware components that are intended to enable long-term use in heterogeneous environments. This sets the robot apart from comparable state-of-the-art systems for long-term autonomy, which are mainly designed for indoor use only. A modular and layered software structure utilizing the ROS framework enables the standardized integration of the methods presented in the further course of this work. The robot features multi-modal perceptual sensing, which allows sensor data to be used with redundancy. This characteristic is particularly utilized for the presented person detection and tracking pipeline. Its qualitative evaluation shows that a robust detection of persons near the robot is possible.

3 Mapping and Localization in Dynamic and Heterogeneous Environments



One key challenge for developing autonomous robots is solving the SLAM problem, which considers the creation of a map of the environment and localization therein. Since this knowledge is an essential requirement for autonomous navigation, SLAM became a significant research area in the robotics community in the last two decades [SLT16]. Whereas for static environments, the SLAM problem can be viewed as solved [CCC+16], long-term applications in dynamic environments are still an open problem. Especially for permanent application in service robots, the SLAM system should fulfill additional requirements besides creating a map and enabling localization. These requirements include execution under limited computational resources, expansion of existing maps, applicability to different sensor modalities, and use of the map for robust navigation. However, existing approaches often focus on only one mentioned aspect and usage in non-heterogeneous environments.

Therefore, this chapter introduces methods to improve mapping and localization in dynamic and heterogeneous environments. Section 3.1 first presents an overview of the SLAM problem and state of the art for solving it using cameras, Lidar sensors, and in long-term applications. Then,

in section 3.2, a method is presented that improves the long-term robustness of graph-based SLAM by using multiple sensor modalities (cameras and Lidar sensors) in parallel. Finally, a map-management system is introduced in section 3.3 to implement mapping for heterogeneous environments (e.g., a mixture of indoor and outdoor areas) by dynamically selecting the appropriate SLAM parameter configuration for each environment.

Section 3.2 of this chapter was in great part published at the peer-reviewed “IEEE/ASME International Conference on Advanced Intelligent Mechatronics" (AIM) 2021 [HSLS21]. The content beginning with section 3.3 was published at the peer-reviewed “IEEE International Conference on Robotics and Automation" (ICRA) 2020 [ESNO20].

3.1 Foundations and Related Work

In the following section 3.1.1, a general introduction to the SLAM problem is given, and the basic solution strategies are presented. Subsequently, current state-of-the-art approaches are described in section 3.1, particularly for long-term-autonomous use in dynamic environments and the service robot context. Finally, a summary and critical discussion are given in section 3.1.3.

3.1.1 The SLAM problem

The objective of solving the SLAM problem is to build a representation (map) of the environment while at the same time estimating the robot's state (i.e., pose) with respect to this map. Due to noisy sensor measurements, the problem is usually stated in a probabilistic fashion.

The robot is assumed to move along a trajectory described by a sequence of locations $\boldsymbol{x}_{1:K} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_K\}$. Odometry characterizes the relative motion between two consecutive locations with control inputs $\boldsymbol{u}_{1:K} = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_K\}$, which might be derived, e.g., from the wheel encoders. Perception of the environment is given via measurements $\boldsymbol{z}_{1:K} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_K\}$. Solving the *full* SLAM problem then consists of estimating the posterior probability

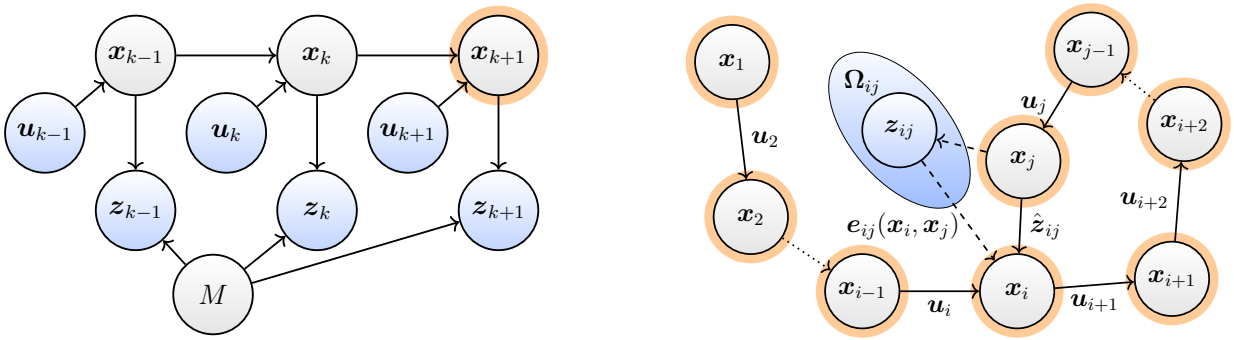
$$p(\boldsymbol{x}_{1:K}, M | \boldsymbol{z}_{1:K}, \boldsymbol{u}_{1:K}) \tag{3.1}$$

for all unknown locations $\boldsymbol{x}_{1:K}$ and environmental map M based on the available sensor data. The representation of M can, for example, be based on spatial landmarks or visual features and depends on the sensor setup of the robot, the environment itself, and the specific SLAM algorithm.

As an estimation of the full trajectory $\boldsymbol{x}_{1:K}$ can be difficult to handle in terms of computational complexity, the *online* SLAM problem aims at estimating only the current state $\boldsymbol{x}_k$ and map M based on the posterior probability

$$p(\boldsymbol{x}_k, M | \boldsymbol{z}_k, \boldsymbol{u}_k), \tag{3.2}$$

which allows for solving the problem incrementally. Therefore, estimation techniques of the online SLAM problem are generally *filter-based*. One other main category form *graph-based* approaches, which usually address the full SLAM problem, albeit incremental techniques exist as well. Both solution approaches are briefly presented in the following sections. Graphical representations of the online and the full problem are shown in Figure 3.1.



(a) The *online* SLAM problem as a BAYESIAN network graph.

(b) Pose-graph representation of a SLAM process, stated as a *full* problem.

Figure 3.1: Graphical depiction of SLAM problem formulations.

Filter-Based SLAM

The underlying principle of filter-based approaches is rooted in the BAYES theorem and consists of a two-step process with successive iterations. In the first step, a prediction of the robot's state and the map is made based on an evolution model and control input $\boldsymbol{u}_k$. The second step uses the current sensor measurement $\boldsymbol{z}_k$ for comparison against the map, correcting the potential error of the previously predicted state.

A major branch of filter-based techniques is EKF-SLAM [CSS87; SSC90], which optimally linearizes the usually non-linear motion models around the actual value of the state vector. Despite the decisive advantage of utilizing non-linear models, EKF approaches are unable to support large-scale and long-term SLAM due to the adversely scaling of their update time as map sizes grow [SLT16; BAYG17].

The other main category of filtering algorithms is based on particle filters, which sample the state with a set of particles based on a probability density. The most widely used algorithm in this category is FastSLAM [MTKW02], which maintains individual data association hypotheses for landmarks and trajectories per particle. Particle-based approaches do not require GAUSSIAN pose distribution and can solve both online and full SLAM problems. Due to its efficient implementation and extensions to occupancy grid maps [GSB07], it is the de facto standard SLAM approach for Lidar-based robots using the ROS. However, its estimation accuracy may suffer from particle depletion and growing complexity for increasing dimensions [APSL08].

Graph-Based SLAM

Graph-based methods view the SLAM problem as a graphical representation, where robot poses and landmark locations are considered nodes in a graph. Consecutive locations $\boldsymbol{x}_{k-1}, \boldsymbol{x}_k$ are linked by an edge resulting from an odometry input $\boldsymbol{u}_k$. Collecting sensor data creates *virtual* measurements (links) $\boldsymbol{z}_{ij}$ [GKSB10], which connect node i with node j by a transformation that makes their respective observations overlap maximally. Identifying these problem constraints based on control inputs and sensor data and using the latter for correspondence detection is referred to as the SLAM *front-end*.

From the configuration of the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ an expected measurement $\hat{\boldsymbol{z}}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be created, which then forms the basis for an error function

$$\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{z}_{ij} - \hat{\boldsymbol{z}}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\pi}\left({}^{i}_{(\boldsymbol{z})}\boldsymbol{T}_j^{-1}\,{}^{i}_{(\boldsymbol{x})}\boldsymbol{T}_j\right). \tag{3.3}$$

The rightmost term expresses the error through homogeneous transformations ${}^{i}_{(\boldsymbol{z})}\boldsymbol{T}_j$ and ${}^{i}_{(\boldsymbol{x})}\boldsymbol{T}_j$ resulting from the observation and odometry, respectively. The mapping $\boldsymbol{\pi} : \mathbb{SE}(3) \to \mathbb{R}^6$ creates a perturbation, and solving the SLAM problem can then be regarded as finding the minimum of a cost function

$$F(\boldsymbol{x}_{1:K}) = \sum_{ij} \boldsymbol{e}_{ij}^{\mathrm{T}} \boldsymbol{\Omega}_{ij} \boldsymbol{e}_{ij}, \tag{3.4}$$

where $\boldsymbol{\Omega}_{ij}$ is the information matrix of the respective virtual measurement. The optimal trajectory $\boldsymbol{x}^*_{1:K}$ can be obtained by optimization:

$$\boldsymbol{x}^*_{1:K} = \underset{\boldsymbol{x}_{1:K}}{\arg\min}\, F(\boldsymbol{x}_{1:K}). \tag{3.5}$$

Refining the robot's trajectory and the map given the constraints defines the second subsystem of graph-based approaches (*back-end*). Due to the sparsity of the graph, eq. 3.5 results in a sparse system of equations which can be solved by methods such as GAUSS-NEWTON, LEVENBERG-MARQUARDT or gradient-descent [GKSB10; SLT16]. As the error function $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ directly depends on measurements, significant constraints for the optimization result when the robot reenters an already known area. The SLAM front-end then seeks matches of the current measurement data with past measurements, a process called *loop closing*. If a match is found between the new node and an existing node in the graph, a constraint is added, labeled with the relative transformation that best overlaps the measured data of the two nodes (cf. Figure 3.1.b).

Compared to filter-based approaches, graph-based SLAM has the advantage of processing larger mapping areas due to its sparse formulation. It can constantly linearize the error function, providing equal or better accuracy and consistency than filters [APSL08; SMD12; BAYG17]. A drawback of graph-based methods is the computational cost of graph optimization, especially for large graphs with many constraints. Therefore, different approaches were presented to perform the optimization

incrementally [OLT07; KJR+12] or to merge [JKFL13] or discard [LM11; CE13] nodes to decrease the graph's size. Their accuracy and flexibility make graph-based methods the primary choice for applications in long-term scenarios or large-scale areas [BAYG17].

3.1.2 SLAM: State of the Art

Due to the high relevance of the SLAM problem in recent years, numerous approaches have been presented that utilize filter-based and graph-based formulations as their underlying principle. The topic has been fundamentally discussed in tutorials [DB06; BD06], as well as several reviews in terms of mapping [Thr+02], autonomous driving [BAYG17], visual methods [FRR15], or in general [HZL19]. A significant driving factor for the type of SLAM approaches is the available underlying sensor hardware, making most methods Lidar or image data-based. For both types, one current challenge is mapping and localization in non-static environments. Therefore, besides presenting methods for Lidar-based and visual SLAM, this section introduces current strategies to enable SLAM in long-term applications and dynamic environments. Special attention is given to the RTAB-Map method [LM19], which allows robust usage in the context of long-term autonomy and for service robot systems.

Lidar-based SLAM

Lidar determines ranges by emitting a laser and measuring the time it takes the reflected light to return to the receiver. Lidar sensors can be divided into 2D Lidar and 3D Lidar, defined by the presence and resolution of vertically oriented Lidar beams. One key advantage of Lidar is that it is not affected by light variations and has a large angle of view and range [CTL+15]. The technological threshold for deployment is comparably high due to cost and complexity. However, reliable and adaptable applications can increasingly be achieved with miniaturization and the development of solid-state Lidar systems [KZI+21].

The most popular Lidar-based *2D SLAM* approach is GMapping [GSB07], which is based on RAO-BLACKWELLIZED particle filtering and extends FastSLAM [MTKW02] with grid maps. Examples for graph-based approaches are KartoSLAM [KGK+10] and LagoSLAM [CACB12]. By adopting sub-mapping and loop-closure capabilities, Google Cartographer [HKRA16] achieves product-grade performance as a SLAM solution for 2D or 3D. *3D SLAM* approaches have the advantage that one more data dimension is available, and six degree of freedom (DoF) poses can be estimated. A popular real-time method for 3D Lidar SLAM is LOAM [ZS14], which handles the problems of odometry estimation and mapping separately.

An essential aspect of Lidar-based SLAM is the detection of loops using Lidar data, which has been implemented through various approaches. Some methods transform the scan into an image based on the range [SRGB11] or additional intensity data of points [MFB11]. Other approaches reduce the dimension of the point cloud by creating regional point descriptors [BZ10], continuous

functions using the normal distribution transform (NDT) [MANL09], or range and height data of points [RMS15]. By azimuthal and radial binning of the Lidar data, an image-like descriptor called scan context [KK18b] can be created, which is also useable for rough localization [KPK19]. In recent years (deep) learning-based approaches gained popularity to increase robustness and accuracy, e.g., PointNetVLAD [UL18] for end-to-end descriptor computation, SegMap [DCD+18] for segment extraction or L3-Net for localization [LZW+19].

Visual SLAM

Due to the increasing computational power of central processing units (CPUs) and graphical processing units (GPUs) as well as cheap camera sensors, the past decade has seen the rapid development of visual SLAM. Compared to Lidar, visual systems can be more lightweight and cheaper, enabling the applicability even in smartphones [MMT15; LGQ+18; KPR+15].

Several approaches exist that use a monocular camera without depth measurements. Methods use sparse features such as oriented FAST and rotated BRIEF (ORB) for ORB-SLAM [MMT15], semi dense representations such as LSD-SLAM [ESC15] or SVO [FZG+16], or dense representations of the full image, such as MLM SLAM [GOLR16] or REMODE [PFS14]. However, one problem of monocular approaches is scale ambiguity and drift [FRR15], preventing the scale of the environment from being estimated and limiting the use of the resulting map for robot navigation.

This problem can be mitigated by combining the camera with inertial measurements or using stereo or RGB-D cameras. Popular visual-inertial graph-based SLAM approaches are VINS-Mono [LGQ+18] and maplab [SDF+18], allowing for localization on large-scale environments. ORB-SLAM2 [MT17] and S-PTAM [PFC+17] are state-of-the-art feature-based methods that can be used with a stereo camera. Learning-based approaches also play an increasing role for visual SLAM systems, for example, for depth estimation of monocular cameras (D3VO) [YSWC20] combining SLAM with semantic mapping (Semanticfusion) [MHDL17] or object detection (Detect-SLAM) [ZWZ+18]. As most visual-based approaches suffer from obstructions or feature-poor environments, some approaches address this problem by using multiple cameras (MCPTAM) [HTS15] or different odometry sources, such as RGBDSLAMv2 [EHS+14].

SLAM in Long-Term Applications

Deploying robots in real-world environments for extended time places high demands on their SLAM approaches. Despite extensive progress in solving the SLAM problem, most approaches assume a static world and do not consider long-term updates of the map or robustness regarding environmental variations. Mapping and localization in changing environments is a current research problem, and approaches to solving it are briefly presented in the following.

Various approaches are based on finding a *robust map representation* invariant to changes over time. Examples include finding robust features [VL10; MUN15], using short-term and long-term

databases [DCD11; LM18] or using sequences instead of single images [MW12; NBS18]. Feature correspondences can be scored by semantic information [TSH+18] or chosen to be robust to varying lighting conditions [RMI13; KCAK17], which is particularly important for visual-based approaches. Lidar-based methods suffer from scene variations due to obstructions or shape changes over long periods (e.g., trees across seasons). To counteract this, point clouds can be represented by error distributions [WN17] or robust scan descriptors [KPK19].

A parallel strategy to using a robust map is to *maintain multiple map representations* and select the most relevant model at the appropriate time and location. Dynamic maps can then handle changes by using multiple timescales [BD05] or clustering of maps created at different times [SB05], or with varying views [KB09]. Similar observations at the same spatial location may be summarized and chosen by matching the data to the current sensory input for localization [CN13]. Particularly for robot applications in large-scale environments [BFG08] or over several floors [SME+12], segmenting the environment into multiple maps is appropriate. Sub-mapping SLAM approaches create local metrical maps of the environment, which are connected by a topological graph for efficient map organization [BNLT04], loop-closure detection [ENT05], or navigation [KMM11]. Spatial segmentation specifically requires detection of environmental transitions [BJL+16], e.g., through object detection [EKJ07], spectral clustering [BKR07], or VORONOI graphs [WSB08]. One common property of the approaches above for sub-mapping [BNLT04; ENT05; KMM11; SME+12] and most SLAM approaches in general [CCC+16], is the need for manual, non-adaptive, parametrization. For example, parameters for feature matching or outlier rejection are kept constant for the complete map.

Contrary to learning the persistent elements of a scene, other approaches aim at *modeling the environmental dynamics*. Specific indicators of the map, such as the occupancy of cells in an occupancy map, might then distinguish between static and dynamic regions [TMB13] or represent periodic processes [KFC+14]. Other approaches explicitly separate static and dynamic regions [HDFF16]. A summary of state-of-the-art approaches in this field is included in section 4.1 since this thesis's next chapter 4 puts a particular emphasis on modeling environment dynamics.

RTAB-Map: Real-Time Appearance-Based Mapping

As previously described, most SLAM approaches are Lidar-based [HZL19] or visual-based [FRR15], and only a subset of methods aim at long-term use, ongoing map extension, or robust localization. One graph-based method that flexibly combines different sensor modalities and focuses on the long-term mapping of large environments is real-time appearance-based mapping (RTAB-Map) [LM19].

As one drawback of graph-based approaches is increasing computational demands for larger graph sizes (cf. section 3.1.1), RTAB-Map implements loop-closure detection [LM13] with a memory-management approach [LM11] as its core. Loop-closure detection is based on bag-of-words representations of visual features [LM13]. When a new node is created, features [BETV08;

Low04; CLSF10; RRKB11] are extracted from the RGB image and added to an incremental visual vocabulary. A discrete BAYESIAN filter keeps track of loop-closure hypotheses, adding links with associated transformations to the graph if required. If Lidar data is available, the transformation is further refined via the iterative closest point algorithm (ICP) [BM92] for each loop closure.

RTAB-Map's memory-management limits the graph size so that the processing time of loop-closure detection and graph optimization stays within real-time constraints. The nodes in the memory are assigned into a working memory (WM) and a long-term memory (LTM), where only the nodes in the WM are considered for loop closure. When RTAB-Map's update time exceeds a given threshold, nodes from the WM are transferred to the LTM based on heuristically defined weights. An overview of the method is shown in Figure 3.2.

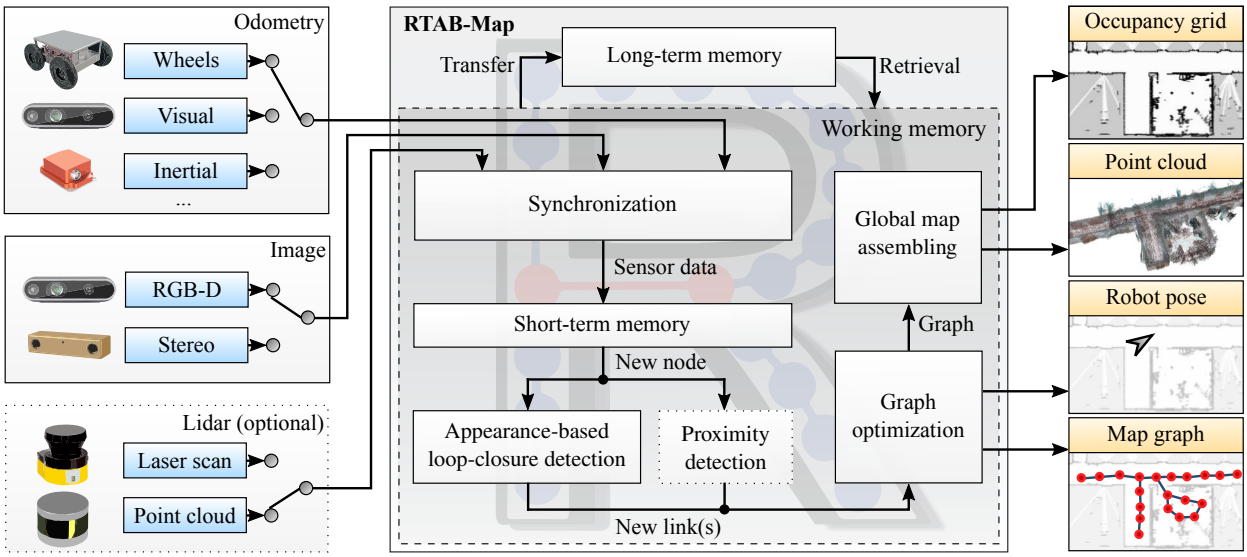


Figure 3.2: Overview of RTAB-Map's main modules, input and output data.

Since appearance-based loop-closure detection is limited by the viewing angle and direction of the utilized camera, RTAB-Map was extended with a *proximity detection* module to use Lidar data to correct odometry drift [LM18]. It was mainly developed for situations where no appearance-based loop closures may be detected, e.g., when mapping a long hall in two directions with a limited field of view.

Besides the already mentioned advantages regarding RTAB-Map's flexibility, it can be considered the most suitable approach for long-term service robot applications due to the following aspects [LM19]:

- **Different modalities**: RTAB-Map supports both visual and Lidar data for SLAM or as odometry input. The latter may additionally consist of wheel-based or inertial data. This flexibility allows for many sensor configurations and easy integration into different systems.
- **Online processing**: As RTAB-Map's computing time is bounded, it can efficiently be used with limited CPU time. Boundedness is especially important in a robotic system where several other processing modules must be executed.

- **Multisession mapping**: The approach allows relocalizing on an existing map and adding new nodes to the graph using image data. In the long-term context, the map can thus be frequently extended on large scales, and new (visual) features can be added.
- **Occupancy map generation**: Most navigation approaches are based on occupancy grid maps. As opposed to many other current SLAM algorithms, RTAB-Map can directly generate such maps, allowing for seamless integration of navigation algorithms.
- **Accuracy**: Compared to other state-of-the-art visual or Lidar-based SLAM approaches, RTAB-Map delivers comparable accuracy while generally providing a larger flexibility.

3.1.3 Conclusion

In summary, it can be stated that Lidar- and visual-based SLAM approaches achieve promising results, especially when static environments are assumed. Long-term applications in dynamic environments are enabled by modeling environmental dynamics or maintaining one or multiple robust maps. However, most approaches focus on single sensor modalities, such as exclusively using Lidar or camera data, each with their respective disadvantages. Furthermore, while basic SLAM functionality is demonstrated for many approaches, it is not always possible to use the resulting map for navigation, which prevents its application in the context of service robotics.

A widely used method for real robot applications is RTAB-Map, which is useable with a wide range of sensors in long-term and large-scale settings while maintaining a bounded computation time. However, as it mainly relies on visual loop closures, the approach has drawbacks that can limit its applicability to dynamic environments. The *marginal use of Lidar data* prevents global loop-closure detection with heavy offsets, e.g., due to varying orientations or for relocalizations. The *constant loop search radius* amplifies this, which does not capture increasing pose uncertainties in long-term applications or for large loops. Applying RTAB-Map to dynamic environments may benefit by extending the framework with Lidar-based loop-closure detection in an adaptive search radius, which is addressed in the next section 3.2.

Another open problem in current SLAM approaches is their reliance on manual parametrizations, whereby the parameters are kept constant for the complete map. Most approaches are *fine-tuned to a specific environment*, which prevents a successful deployment in other areas (e.g., transitions between indoor and outdoor or small-scale and large-scale areas). Automatically adapting the specific SLAM method or configuration based on the type of environment and creating an efficient map structure could be beneficial for (large-scale) heterogeneous environments. However, this is still an open research problem that is addressed in section 3.3 of this chapter.

3.2 Lidar-Based Loop-Closure Detection

The long-term autonomy of robots in dynamic environments has two main requirements: The computational complexity of graph optimization (back-end) must be limited, and the process must be robust against environmental changes. As stated in the previous section, most SLAM approaches focus on single sensor modalities, and only very few use a combination, e.g., of both Lidar and camera data. However, multiple modalities can be especially useful in dynamic and heterogeneous environments. Appearance-based loop inferences, for example, are well suited for indoor areas with distinctive features such as patterns on walls. On the other hand, loop detections with Lidar data are not affected by light variations and are better at representing large-scale building structures. Therefore, this section introduces an extension of the widely used graph-based RTAB-Map method to improve long-term robustness by more extensive use of 3D Lidar data. The proposed method consists of a multi-step process, where node pairs for possible loop closures are searched for in a variable radius based on odometry data, scan descriptors, and binary classification (cf. section 3.2.1). If a loop is detected, the respective point clouds are registered, and the graph is optimized with the relative transformation (cf. section 3.2.2). A schematic overview of the extension is shown in Figure 3.3.

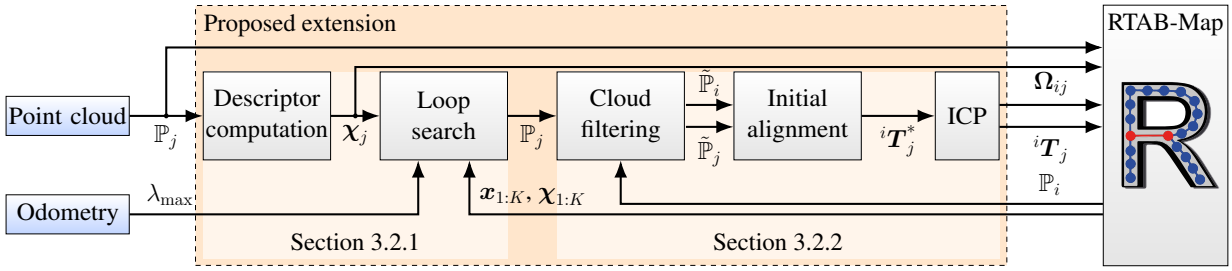


Figure 3.3: Schematic overview of the proposed Lidar-based extension for loop-closure detection.

3.2.1 Loop Classification

Consider a robot that is equipped with a 3D Lidar sensor and moves through the environment, creating nodes in the SLAM graph at locations $\boldsymbol{x}_{1:K}$ based on its odometry inputs $\boldsymbol{u}_{1:K}$ (cf. section 3.1.1). Each node i can then be assigned a point cloud

$$\mathbb{P}_i = \left\{ \boldsymbol{p}_i^k \right\}_{k=1}^{N}, \quad \boldsymbol{p}_i^k \in \mathbb{R}^3, \tag{3.6}$$

containing N points representing the environment. To determine whether the robot revisits a known location, the most recent point cloud $\mathbb{P}_j$, $j > K$, is compared with point clouds $\mathbb{P}_{1:K}$ in the graph. Detecting a loop can then be viewed as a binary classification problem, where two point clouds either represent the same environment or not. Due to the high data amount of point clouds (e.g.,

$N \approx 45{,}000$ with the robot presented in chapter 2), feature-based approaches are suitable for this purpose. As introduced in [GS10], each point cloud $\mathbb{P}_i$ is described by global features

$$\boldsymbol{\chi}_i = \left(\boldsymbol{\chi}_i^{\mathrm{I}}, \boldsymbol{\chi}_i^{\mathrm{II}}\right)^{\mathrm{T}}, \quad \boldsymbol{\chi}_i \in \mathbb{R}^{n_\chi}, \quad n_\chi \ll N, \tag{3.7}$$

where $\boldsymbol{\chi}_i^{\mathrm{I}}$ and $\boldsymbol{\chi}_i^{\mathrm{II}}$ are two different types of features. The first type $\boldsymbol{\chi}_i^{\mathrm{I}}$ maps the point cloud to a real number, typically geometric properties such as the point cloud's average range, volume, or centroid. For type $\boldsymbol{\chi}_i^{\mathrm{I}}$, 32 features are computed. The second type of features $\boldsymbol{\chi}_i^{\mathrm{II}}$ are range histograms with nine varying bin sizes $b_{1:9}$. Using these two types of features is generally faster than, e.g., computing the normal distribution transform (NDT) [MANL09] and more descriptive than scan context [KK18b]. Starting at the sensor's origin, its detection area is divided into annular regions, each with ring size b_j to create the range histograms (see Figure 3.4). The EUCLIDEAN distance r_k to the sensor's origin is calculated for each point $\boldsymbol{p}^k$. If the range r_k is above a maximum defined range $r_{\max}$, the point is translated towards the origin so that $r_k = r_{\max}$ applies. With the proposed parametrization from [GS10], the 41 used features result in a length of the feature vector $n_\chi = 843 \ll N$, which is significantly lower than the dimension of the point cloud. Since the descriptor is computed with basic mathematical operations, it is determined for each new point cloud and assigned to each node in the graph. Additionally, the descriptor is rotationally invariant, reducing the perspective-dependent data-association problem.

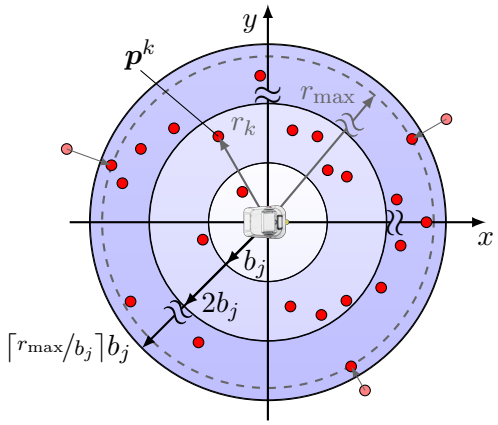


Figure 3.4: Two-dimensional depiction of annular regions with bin size b_j. Counting the points $\boldsymbol{p}^k$ in the different areas results in the respective range histograms.

Two difference metrics are used to compare a descriptor $\boldsymbol{\chi}_j$ from a new point cloud with a descriptor $\boldsymbol{\chi}_i$ from the graph. For scalar features $\left\{\boldsymbol{\chi}_i^{\mathrm{I}}, \boldsymbol{\chi}_j^{\mathrm{I}}\right\}$, the element-wise absolute value of the feature vector difference is computed. PEARSON'S correlation coefficient between the respective range histograms $\left\{\boldsymbol{\chi}_i^{\mathrm{II}}, \boldsymbol{\chi}_j^{\mathrm{II}}\right\}$ is used for the second feature type. Comparing similar feature vectors results in a vector with 32 entries close to 0 and the last nine entries close to 1. This vector then forms the input of the classification problem. As the classification is considered binary, the output is defined as ${}^i y_{\mathrm{c}} = \{0, 1\}$ for negative and positive loop pairs.

Due to its performance and robustness against overfitting, [GCRN09] proposes to use an AdaBoost classifier. AdaBoost [FS97] learns in an iterative procedure, forming a strong classifier from a combination of T weak classifiers (decision stumps). In each learning round, the data is reweighted based on the current prediction error, with incorrect classifications having higher prioritization. With the choice of the specific value of T significantly influencing the accuracy, Granström et al. [GS10] evaluate different combinations and recommend $T = 50$ training rounds for the used descriptor in indoor and outdoor applications. To receive a binary output from the classifier, the output probability ${}^{i}p_{\mathrm{c}}$ is thresholded with a parameter $p_{\min}$, so only pairs with ${}^{i}p_{\mathrm{c}} > p_{\min}$ are treated as positive loop pairs.

Defining the Search Space

Although processing descriptors via the AdaBoost classifier is not computationally costly, comparing a new descriptor from each iteration with all existing ones from the graph is unnecessarily expensive. This applies especially to long-term operations, which can lead to graphs with thousands of nodes. A standard method in graph-based SLAM is to include heuristics, which only choose a subset of all available nodes for loop-closure detection. Similar to RTAB-Map's proximity detection (cf. section 3.1.2), the current descriptor is only compared with other nodes within a certain radius r around the estimated pose (see Figure 3.5.a).

However, searching in a constant radius is disadvantageous since the estimated pose might be highly inaccurate in long phases without any loop closures. Therefore, a variable search space depending on the pose accuracy is used. The search radius

$$r(\lambda_{\max}) = r_{\min} + \beta g_{\max}(\lambda_{\max}) \quad \text{with} \quad g_{\max}(\lambda_{\max}) = 2\sqrt{5.991\lambda_{\max}} \tag{3.8}$$

consists of two parts. The constant radius $r_{\min}$ defines the minimal size of the search circle, which applies if the estimated pose can be assumed to be without errors. This is the case at the start of the process and directly after a loop closure occurs. Between these events, the position of the created nodes is subject to erroneous odometry estimates. Common error models for odometry estimation, especially KALMAN-filter-based sensor fusion approaches, model the resulting odometry estimate as normally distributed. Therefore, the second part of the search radius depends on the length $g_{\max}$ of the longest major axis of the 95 % confidence ellipse, resulting from the largest eigenvalue $\lambda_{\max}$ of the odometry estimate's covariance matrix [Hoo84]. The scale factor β determines the overall influence of the second part and can be chosen heuristically, e.g., based on the scale of the environment. Due to error propagation, the covariance matrix values are monotonically increasing and are frequently reset when a loop closure occurs.

Searching loops locally is only possible when a new graph is created, at least one link has been found, or an initial guess exists. For localization or extension of an existing map in a multi-session operation (wake-up robot problem), a global search must be performed to connect the local (new)

map with the existing map. This is detected by a ratio $\alpha = \frac{n_{\text{local}}}{n_{\text{WM}}}$, where n_{local} is the number of nodes in the local map and n_{WM} the number of all nodes in working memory (WM) (see Figure 3.5.b). A low ratio α indicates that the relative positions of many nodes in WM are unknown, requiring searching within the entire WM for loop pairs until $\alpha \geq \alpha_{\text{min}}$, with threshold α_{min}. The fast detector enables processing hundreds of nodes in the WM. However, since adding a wrong loop pair is fatal for the map's integrity, for the first n_{start} detected loops, there must also exist a pre-defined minimum number n_{ver} of detected pairs with nodes in the immediate neighborhood with radius r_{ver} around the target node for verification.

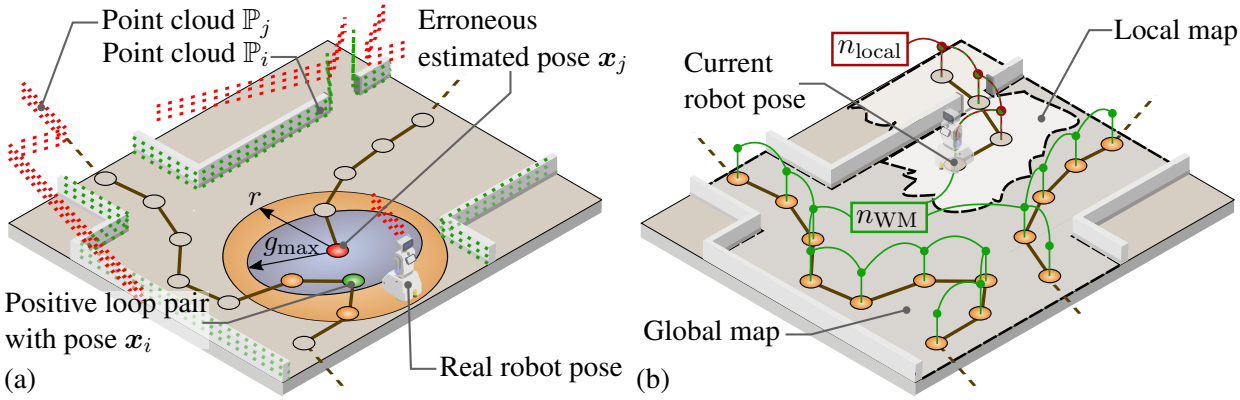


Figure 3.5: Searching for loop pairs in the local (a) and global (b) case. The filled dots are nodes that are considered for loop pairing.

3.2.2 Point-Cloud Registration

In the case of positive loop detection, the point clouds are registered, i.e., their respective homogeneous transformation ${}^{i}\boldsymbol{T}_{j}$ must be found. The transformation is computed by a two-step registration, with an initial feature-based global alignment followed by a fine (local) registration based on the ICP algorithm. The registration process is based on [HIT+15] and is illustrated in Figure 3.6.a.

Registering raw point clouds is computationally expensive, and certain regions are uninformative for correspondence detection. Examples are points on the ground or regions resulting from erroneous laser beam multi-reflections on glass fronts. To encounter these effects, the following filters are executed consecutively:

1. **Voxel grid** filter to consolidate points into box centroids,
2. **Height** filter to remove points below a specific height,
3. **Intensity** filter to remove points with low intensity, e.g., caused by multi-reflections,
4. **Range** filter to remove points far away from the sensor origin,
5. **Random downsampling** to randomly remove points down to a maximum number of points.

Next, features based on fast point feature histograms (FPFH) [RBB09] are computed. These are robust multidimensional features describing a local spatial geometry around a point.

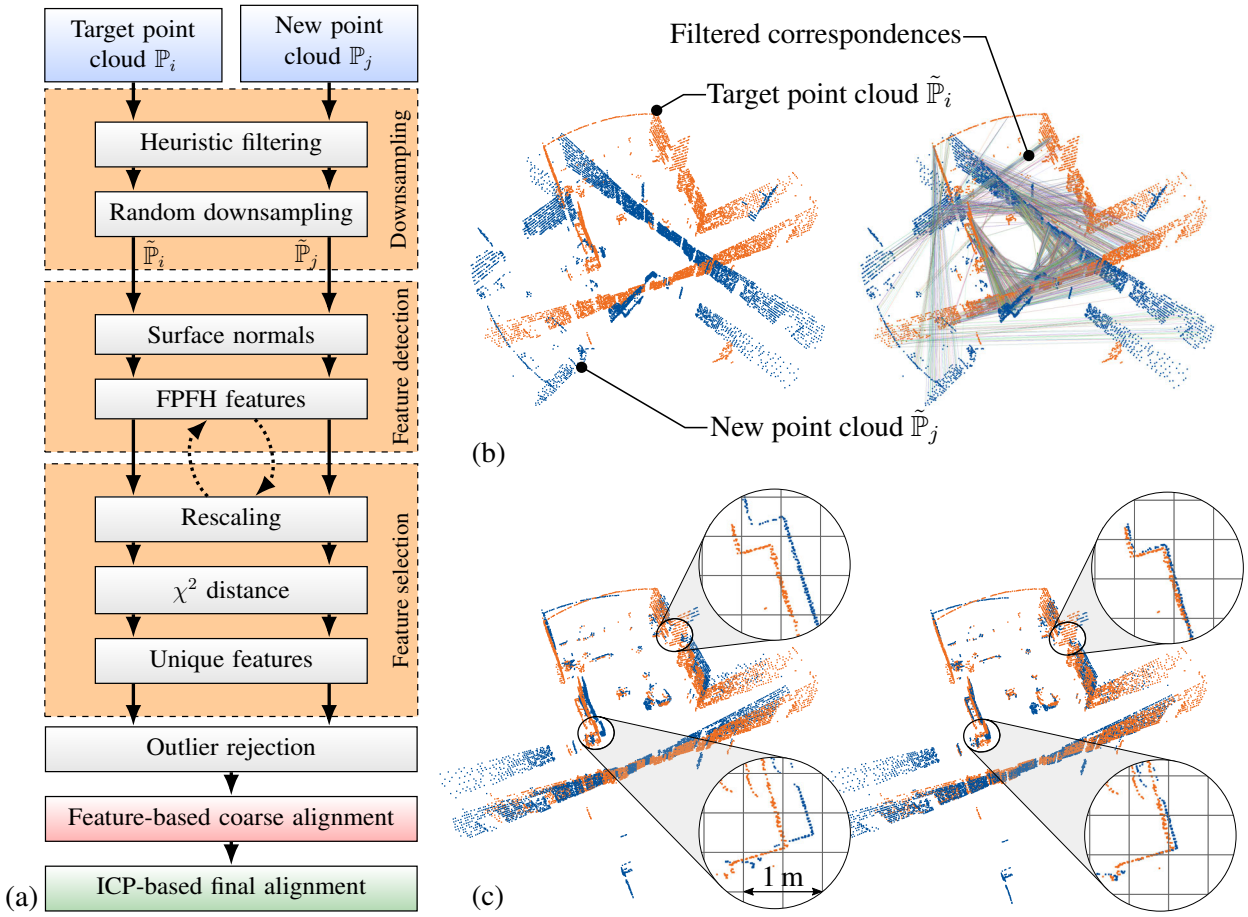


Figure 3.6: Sequence of the registration (a). Registration result: shown are the relative positions with filtered FPFH correspondences of downsampled point clouds in (b), and a detailed view of coarse (left) and final (right) alignment in (c).

For robust correspondence estimation, features are computed for different support sizes (scales), and only those are kept which prove to be persistent over multiple scales based on the χ^2-distance metric between feature vectors. Correspondences are then determined by nearest-neighbor search in the feature space. As the point clouds overlap only partially, false correspondences are rejected via random sample consensus (RANSAC) [FB81]. By reiteratively computing transformations, only the correspondences leading to small distances between source and target points are accepted. The coarse alignment is computed using singular-value decomposition based on the filtered correspondences (see Figure 3.6.b). Coarse alignment provides an initial guess for the ICP-based refinement to finally receive the transformation ${}^{i}\boldsymbol{T}_{j}$. Since graph optimization requires an uncertainty estimate, described by information matrix $\boldsymbol{\Omega}_{ij} \in \mathbb{R}^{6\times6}$, a nearest-neighbor correspondence estimation is carried out between the aligned clouds. The information matrix is then defined as $\boldsymbol{\Omega}_{ij} = {}^{1}/_{\hat{\sigma}^2}\boldsymbol{I}$, where the variance $\hat{\sigma}^2$ is estimated based on the median absolute deviation of correspondence distances.

3.2.3 Results

The evaluation of the proposed RTAB-Map extension is divided into three parts: At first, the detector is trained and tested on data from unseen environments. Secondly, multi-session experiments are performed under challenging conditions to investigate the robustness to environmental changes. The first two parts use the robot and sensor setup described in section 2.1.2. Lastly, the general applicability of the proposed approach is demonstrated with the widely used KITTI dataset [GLU12].

Detector Performance

Descriptors, calculated based on indoor and outdoor areas of a university campus, serve as training data for the detector. Since there is no ground-truth position, the estimated poses from RTAB-Map are used to generate 1,248 nodes with a total path length of 697 m. The data acquisition is performed while keeping viewing areas and lighting conditions as constant as possible to obtain a robust localization with the sole use of RTAB-Map. The mapping process for ground-truth creation is conducted offline in order to avoid reduction of accuracy due to computing time restrictions. A loop pair is treated as positive when the EUCLIDEAN distance between the two nodes is less than 3 m, and all other cases are treated as negative pairs. Due to the proportionally much higher number of negative pairs, the data set is re-balanced by random subset selection so that there are equal numbers of positive and negative pairs. As the specific random subset significantly influences the classifiers's performance, 50 AdaBoost classifiers are trained on different subsets, each via 10-fold cross validation. For comparison, common criteria consisting of the true positive rate (TPR) and false positive rate (FPR) are used:

$$\mathrm{TPR} = \frac{\text{\# Positive data pairs classified as positive}}{\text{\# Positive data pairs}},$$

$$\mathrm{FPR} = \frac{\text{\# Negative data pairs classified as positive}}{\text{\# Negative data pairs}}.$$

Since adding incorrect loop closures is significantly more fatal than adding fewer correct loop closures, $\mathrm{FPR} < 1\,\%$ is set as the target, and the best classifiers with respect to the TPR is chosen by incrementally increasing threshold parameter $p_{\min}$. This target value for the FPR is suitable because the proposed extension and RTAB-Map still verify every possible loop pair before adding it to the graph. The best of the 50 detectors is tested on data from unseen indoor and outdoor environments to make a specified statement about the choice of $p_{\min}$ in addition to cross validation. The loop in the indoor hall is illustrated in Figure 3.7.a together with receiver operating characteristic (ROC) curves for the indoor and outdoor test datasets in Figure 3.7.b.

The classification matrix visualizes the test results for the indoor data (see Figure 3.7.d) and distance matrix (see Figure 3.7.e). The latter represents the ground truth since all pairs with a distance less

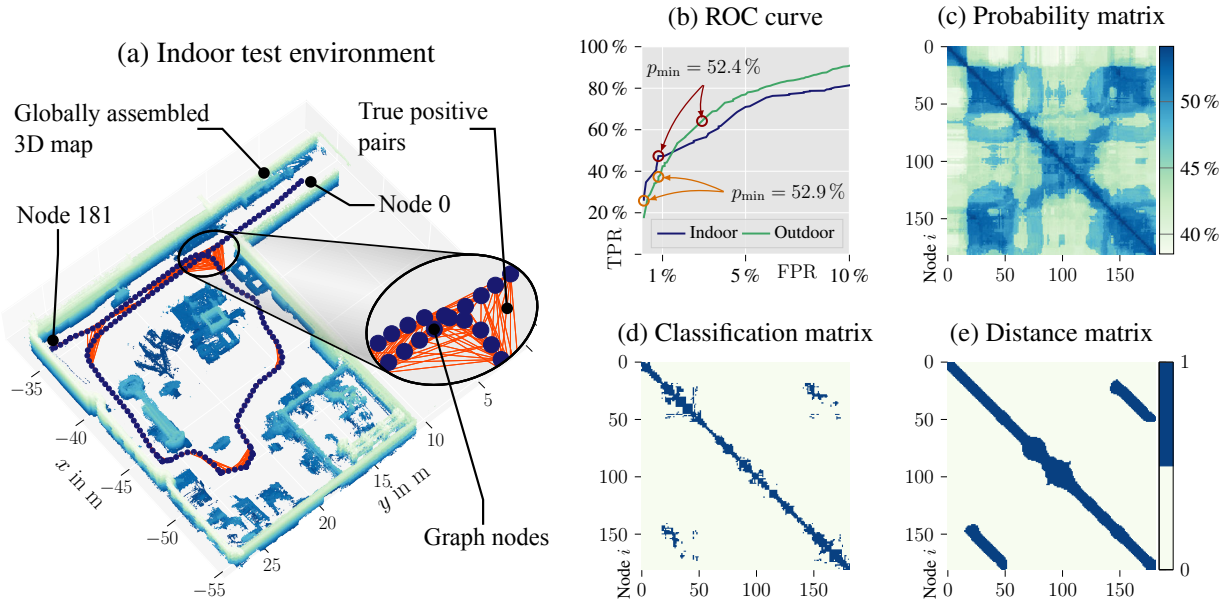


Figure 3.7: Indoor test environment including nodes and true-positive detections in (a). ROC shows the performance of the best detector for different thresholds $p_{\min}$ for both datasets. The two given values of $p_{\min}$ indicate the performance at the desired boundary $\mathrm{FPR} < 1\,\%$. Thresholding the probability matrix (c) with $p_{\min} = 52.4\,\%$ results in the classification matrix (d) that can be compared with the distance matrix (e) resulting from the ground truth. The matrices (c, d, e) refer to the indoor dataset.

than 3 m are treated as a loop. These matrices are roughly similar, and all detected loops are located in areas where the pairs are not far from each other. Generally, the classifier leads to better results for the indoor case ($\mathrm{TPR} = 47.3\,\%$, $\mathrm{FPR} = 0.8\,\%$, $p_{\min} = 52.4\,\%$) than for the outdoor case ($\mathrm{TPR} = 37.3\,\%$, $\mathrm{FPR} = 0.8\,\%$. $p_{\min} = 52.9\,\%$). This is mainly due to the challenging outdoor environment used for testing, which consists of repetitive structures (mainly architectural pillars, Figure 3.8 from the following section shows a similar environment). Holding the $\mathrm{FPR} < 1\,\%$ boundary then requires choosing the value of $p_{\min}$ more strictly.

Relocalization

This evaluation compares the complete detection and registration pipeline with RTAB-Map's default operation. The experiments involve mapping three environments (two indoor hallways and one outdoor campus) and a subsequent map extension with data from another day at another time of day (wake-up robot problem). Relocalization is performed on different paths, each with a two-minute movement duration within each environment. Between the initial mapping and relocalization is a timespan of eight weeks. The most significant changes between the time points are in the different lighting conditions and the appearance/disappearance of various objects. The specific parameters of the experiment are given in Appendix A, with the most crucial difference being the different loop detection threshold $p_{\min}$ for indoor and outdoor environments. Figures 3.8 and 3.9 show the

outdoor and indoor environments as 3D maps and the paths for mapping and relocalization with the name labels indicating the starting positions. Relocalization is successful if at least one loop closure is found and a correct reference to the original map is calculated. Table 3.1 shows the result of the experiment, with the proposed extension denoted as RTAB-Map+LL (LL=Lidar loop).

Method	Campus (outdoor)	Offices (indoor)	Entry hall (indoor)
RTAB-Map	0 / 6	3 / 5	0 / 3
RTAB-Map+LL	**5** / 6	**4** / 5	**3** / 3

Table 3.1: Results of the relocalization experiment (# successful / # total)

RTAB-Map's original configuration struggles to detect correspondences, mainly influenced by changes in sunlight and artificial light, people in the pictures, and different fields of view. Additionally, observing the three-dimensional surrounding structure via RTAB-Map+LL leads to significantly better results, especially in environments with strongly varying sunlight.

A major challenge, both for visual and Lidar-based approaches, are feature-poor environments (e.g., long hallways) or strongly repetitive structures which result in two unsuccessful segments of RTAB-Map+LL.

KITTI dataset

A final evaluation is done with the widely used KITTI [GLU12] odometry benchmark to demonstrate the general applicability of the proposed extension. The dataset sequences containing loops are mapped via RTAB-Map+LL using the RTAB-Map parameters for KITTI [LM19] with the odometry being calculated based on the front stereo camera. Loops are detected using the same trained detector with indoor and outdoor data from the previous section. The method's parameters are adjusted for the car and road traffic and are given in appendix A. The comparison of RTAB-Map without and with the LL extension is based on the absolute trajectory error

$$\mathrm{ATE} = \left(\frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{e}_{\mathrm{t},i} \|^2 \right)^{\frac{1}{2}}, \tag{3.9}$$

where $\boldsymbol{e}_{\mathrm{t},i}$ describes the translational offset between the i-th estimation and ground truth with the same timestamp and a total number of n positions defining the trajectory. The results in Table 3.2 show a significant enhancement, especially for sequences with street sections that are traversed multiple times in different directions, such as sequence 08 (see Figure 3.10). To summarize, RTAB-Map generally benefits from the additional loop closures detected based on Lidar data.

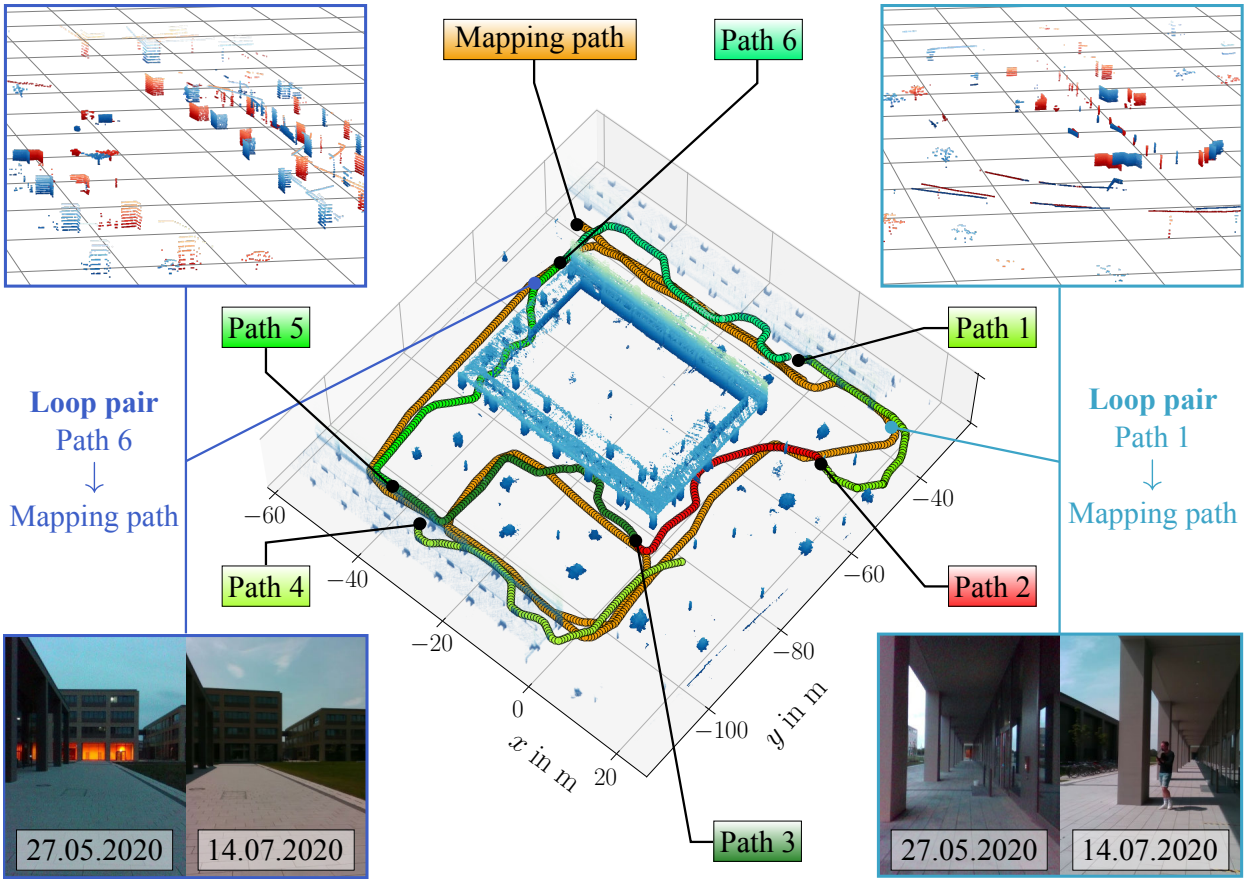


Figure 3.8: Outdoor campus environment for the relocalization experiment. On greenish (reddish) illustrated paths relocalization with the proposed method was successful (unsuccessful). For two exemplary loops, the camera images and detected corresponding point clouds (pre-registration) are given.

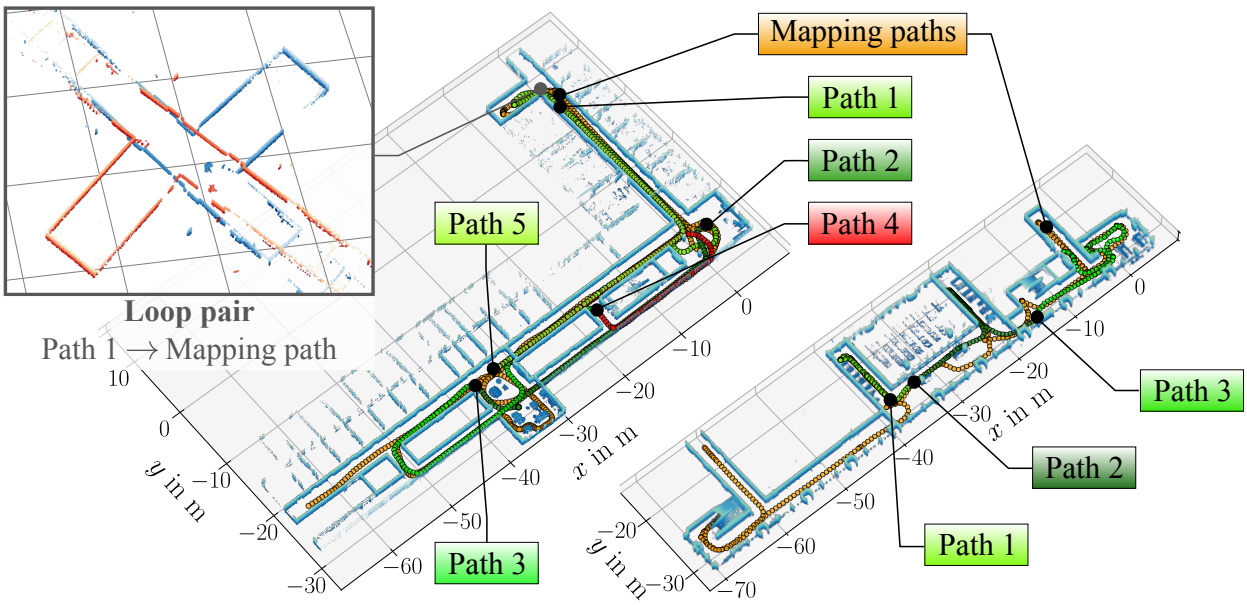


Figure 3.9: Indoor environments office (left) and entry hall (right) for relocalization experiment. The same notation as for Figure 3.8 applies.

Sequence	00	02	05	06	07	08	09
w/o LL, ATE	1.01	4.4	**0.51**	0.74	0.48	3.85	2.77
with LL, ATE	**0.93**	**4.19**	0.63	**0.72**	**0.47**	**2.9**	**2.6**
# visual loops	156	51	88	54	18	0	3
# LiDAR loops	126	5	72	38	18	30	1

Table 3.2: Results for the KITTI odometry dataset. ATE given in m.

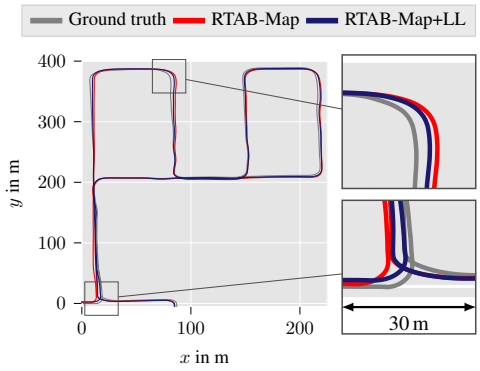


Figure 3.10: SLAM paths and ground truth for sequence 08.

3.3 Modular Multi-Environment Mapping

Most SLAM algorithms generate maps based on specific sensor data, such as point clouds or images, and parameter configurations specified by human experts in advance. As already indicated in the results from the previous section 3.2.3, the configuration might only be suitable for a specific type of environment. Once the environment changes while mapping (for example, an indoor/outdoor change), the SLAM algorithm might produce suboptimal results. Since service robots often operate in heterogeneous environments, such as airports or hospitals with small and large rooms, creating a single (large) map with one specific parameter and sensor configuration is disadvantageous concerning SLAM accuracy, scalability, and map quality.

Therefore, this section introduces a mapping approach, which detects transitions in the environment to create individual, environment-specific maps to automatically choose the appropriate sensor and SLAM configuration. It is not a SLAM algorithm but a map-management system, which organizes maps created by arbitrary SLAM methods in a topological-metric form. This arrangement has the advantage that there is still a global map, e.g., for navigation. Small specific maps are used in the field to better represent the environment and use fewer computing resources. The graph structure is presented in the next section 3.3.1, followed by the criteria to trigger new maps and choose environment-specific map configurations in section 3.3.2. Individual maps are linked to create a globally consistent structure in section 3.3.3, and the method is finally evaluated in section 3.3.4.

3.3.1 Graph Structure

The main idea of the map-management system is to create a higher-level graph structure that spatially links different maps. Since the basic layout of the operational environments of service robots can be considered structured and separated into rooms, doorways are chosen as transition points between the different maps. The graph structure is described as a hybrid map $H = \langle M_{1:n}, L_{1:m}, E_{1:p} \rangle$, where $M_{1:n} = \{M_1, ..., M_n\}$ is a set of metric maps, $L_{1:m} = \{L_1, ..., L_m\}$ is a

set of nodes and $E_{1:p} = \{E_1, ..., E_p\} \subseteq L_{1:m} \times L_{1:m}$ is a set of edges. Each map in the set $M_{1:n}$ represents a single room, corridor, or outdoor environment separated by doors. The graph's nodes $L_{1:m}$ are viewed as link points connecting adjacent maps in an overlapping area of both maps and are placed directly in front of or behind doorways. Each link point's pose is fully described by a tuple $\langle {}_{(j)}\bar{\boldsymbol{m}}_i, {}_{(k)}\bar{\boldsymbol{m}}_i \rangle$ where ${}_{(j)}\bar{\boldsymbol{m}}_i, {}_{(k)}\bar{\boldsymbol{m}}_i \in \mathbb{R}^3$ represent 2D-poses in overlapping areas of maps M_j and M_k (with $j \neq k$) so that ${}_{(j)}\bar{\boldsymbol{m}}_i, {}_{(k)}\bar{\boldsymbol{m}}_i \in M_j \cap M_k$. The poses ${}_{(j)}\bar{\boldsymbol{m}}_i = {}_{(j)}\left(x_i, y_i, \theta_i\right)^{\mathrm{T}}$ and ${}_{(k)}\bar{\boldsymbol{m}}_i = {}_{(k)}\left(x_i, y_i, \theta_i\right)^{\mathrm{T}}$ are defined relative to their respective map frames $(\mathrm{CF})_j$ and $(\mathrm{CF})_k$ and are equal relative to the global coordinate frame $(\mathrm{CF})_0$. The graph's edges $E_{1:p}$ result by fully connecting all link points that share the same map. Each edge E_i is assigned a weight based on the EUCLIDEAN distance between the corresponding nodes, which enables the use of graph-search algorithms for path planning and global tracking between different maps. Figure 3.11 shows an exemplary arrangement of metric maps and their connection via link points. At every doorway is an overlap between two adjacent maps and a connecting link point (see Figure 3.11.a), consisting of the tuple $\langle {}_{(j)}\bar{\boldsymbol{m}}_i, {}_{(k)}\bar{\boldsymbol{m}}_i \rangle$ (see Figure 3.11.b).

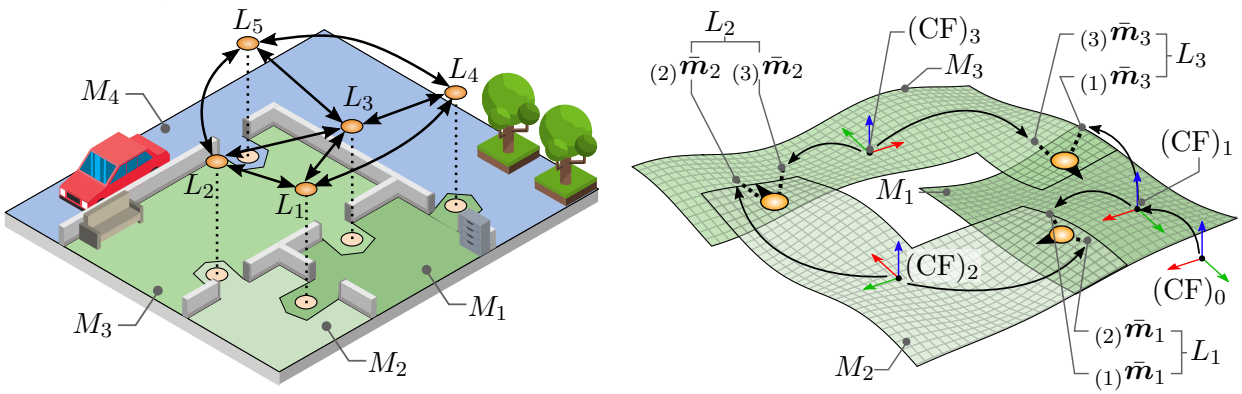


Figure 3.11: Topological map structure of areas with a different type (e.g., indoor, greenish and outdoor, blueish) in (a). Coordinate frames of the first three corresponding metric maps and link points are shown in (b).

3.3.2 Environment Transitions and Configuration Changes

An indicator signaling a doorway's passing is required to place link points in overlapping areas of adjacent maps directly in front of or behind doors. Regarding superordinate map management, this can be an arbitrary detector, which, e.g., processes Lidar or visual data. Therefore, approaches based on the peak signal-to-noise ratio, structural similarity [WBSS04], or image classifiers based on indoor/outdoor datasets [ZLK+18] are conceivable. A simple but effective approach for doorway detection with 360° 2D laser-scan data is presented in the following.

Three criteria to detect a doorway are considered where at least two must be met: (C1) front door post detection, (C2) back door post detection, and (C3) significant change in wall distances.

Door post detection utilizes the depth variation at the posts compared to the surrounding wall. The transition is determined via the spatial derivative of the laser-scan ranges, resulting in two characteristic range steps when the robot is in front of the door. The same applies after the doorway has been traversed for the area behind the robot (C2). Doors that are close to orthogonal walls may result in small derivatives of laser-scan data. To counteract this, the third criterion enhances the robustness by average-low-pass filtering the laser-scan data and transforming it into the map frame $(\mathrm{CF})_i$. By placing a bounding box $\mathcal{B}$ around the filtered data, the rough shape of the room is available and invariant to the robot's orientation. Criterion (C3) is fulfilled if any of the bounding boxes' minimum and maximum derivative values in ${}_{(i)}x$- or ${}_{(i)}y$-direction change over a specific time. Figure 3.12 illustrates the three criteria based on two robot positions in front of and behind and door.

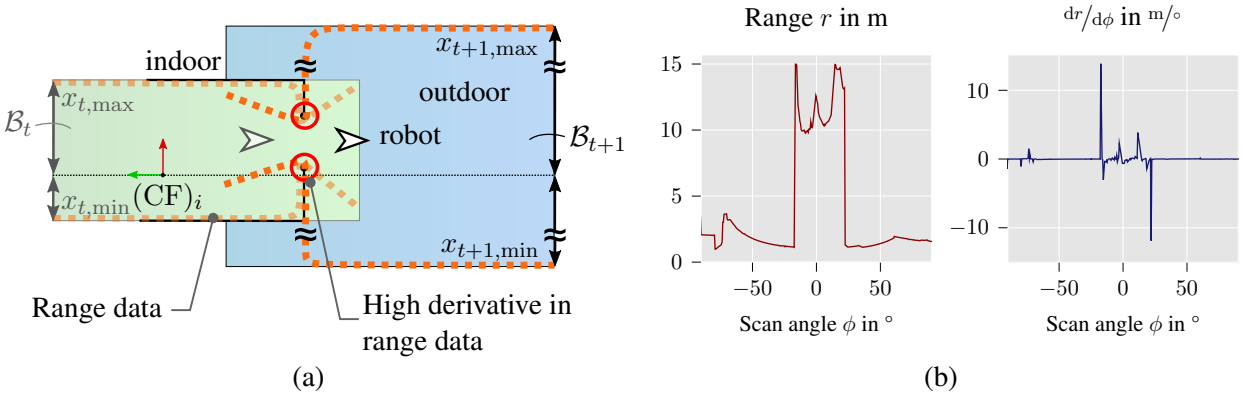
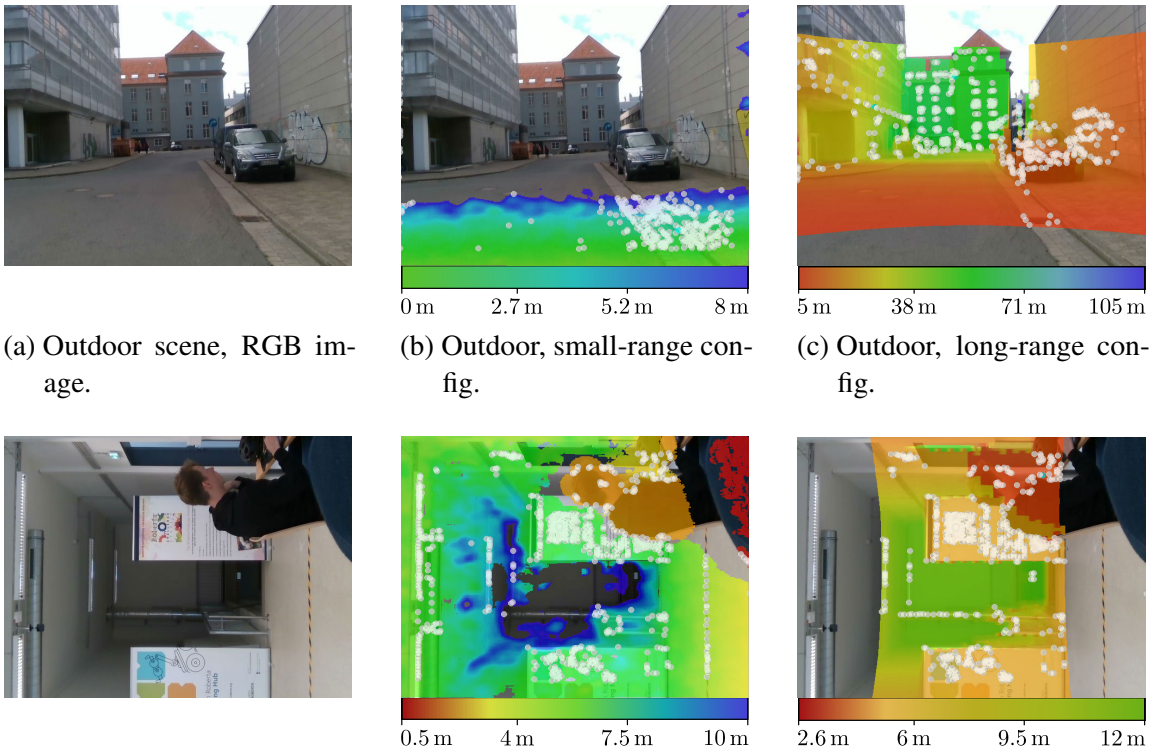


Figure 3.12: Criteria to detect door passing and environment changes: In (a) high derivatives of range data in front of and behind the robot are shown (criteria (C1) and (C2)). The respective bounding boxes indicate abrupt changes in the scale of the environment (criterion (C3)). (b) shows an exemplary range measurement in front of a door and its derivative with clearly visible peaks.

Changing SLAM Configurations

After door passing and before triggering a new mapping instance, the SLAM parameter configuration for the new map must be determined. The main difference between the configurations in the proposed method is based on the dimensions of the respective environment, i.e., if the environment is small- or large-scale. This is motivated by the fact that in small-area environments, data distribution near the robot is denser, which means that other feature types could be suitable. Furthermore, it allows using completely different sensors, e.g., camera-based RGB-D data in small indoor areas and 3D Lidar data with a higher range in large outdoor areas. Since the distinguishing feature is range-based, the previously introduced bounding boxes $\mathcal{B}$ around filtered range data are again used to determine which SLAM configuration to select. If the width and height of the bounding box are over a specified threshold after door passing, the configuration for large rooms and high distances applies. The threshold can be empirically determined based on specific sensor parameters, such as the accuracy or maximum range of an RGB-D camera.

Besides changing common SLAM parameters, such as feature type, maximum ranges, or confidence thresholds, the specific data source could also be changed. When a 3D Lidar is available in conjunction with an RGB-D camera (cf. section 2.1.2), it might be suitable to replace the camera's depth data with the Lidar data as shown in Figure 3.13. The Lidar points are projected into the camera sensor frame, and continuous depth information is generated by interpolation, resulting in a smaller resolution. However, features can be extracted from distances up to 100 m (see Figure 3.13.c) opposed to less than 10 m based on infrared-stereo data (see Figure 3.13.b), which also suffers from sunlight [HSKV19]. In small-range (i.e., indoor) areas, the smaller field of view resulting from the projection might be disadvantageous, and the RGB-D data could be used (compare Figure 3.13.e and Figure 3.13.f).



(a) Outdoor scene, RGB image.
(b) Outdoor, small-range config.
(c) Outdoor, long-range config.
(d) Indoor scene, RGB image.
(e) Indoor, small-range config.
(f) Indoor, long-range config.

Figure 3.13: RGB images overlaid with depth images and registered scale-invariant feature transform (SIFT) features (white circles) created for two different scenes and configurations. The long-range configuration uses the Lidar data registered with the RGB image, whereas the small-range configuration uses the stereo-depth image from the RGB-D camera. Images (d)–(f) were captured with a vertically oriented camera.

3.3.3 Global Graph Creation

When new link points are created, due to odometry drift it is not guaranteed that the currently estimated robot pose corresponds to its actual pose. This error can accumulate when multiple maps

are created, resulting in a graph not correctly representing the environment. Therefore, tracking the robot pose globally and establishing a metric relation between the different maps is necessary. The poses of the link points relative to the adjacent map origins are saved in the graph nodes.

Let L_i be the link point which connects map M_i and map M_{i+1}, and ${}^{i}\boldsymbol{T}_{L_i} \in \mathbb{SE}(2)$ is a homogeneous transformation matrix describing the pose of L_i in coordinate frame $(\mathrm{CF})_i$ and ${}^{i+1}\boldsymbol{T}_{L_i}$ equally describing the link point's pose in $(\mathrm{CF})_{i+1}$. The pose describing the translation and rotation between both origins follows as

$$ {}_{(i)}\bar{\boldsymbol{p}}_{i+1} = \boldsymbol{\pi}\left({}^{i}\boldsymbol{T}_{i+1}\right) = \boldsymbol{\pi}\left({}^{i}\boldsymbol{T}_{L_i}\left({}^{i+1}\boldsymbol{T}_{L_i}\right)^{-1}\right), \quad {}_{(i)}\bar{\boldsymbol{p}}_{i+1} \in \mathbb{R}^3, \tag{3.10} $$

where $\boldsymbol{\pi} : \mathbb{SE}(2) \to \mathbb{R}^3$ is a function that maps the homogeneous transformation to a 2D pose, which is then regarded as the expected value ${}_{(i)}\bar{\boldsymbol{p}}_{i+1}$ of a normal distribution with covariance matrix ${}^{p}_{(i)}\boldsymbol{\Sigma}_{i+1}$. Let the uncertain pose of a link point ${}_{(i)}\boldsymbol{m}_i \sim \mathcal{N}({}_{(i)}\bar{\boldsymbol{m}}_i, {}_{(i)}\boldsymbol{\Sigma}_i)$ also be a normally distributed random variable with expected value ${}_{(i)}\bar{\boldsymbol{m}}_i$ and covariance matrix ${}_{(i)}\boldsymbol{\Sigma}_i$. Considering ${}_{(i)}\boldsymbol{p}_{i+1}$ and ${}_{(i)}\boldsymbol{m}_i$ as random variables enables to incorporate uncertainty resulting from erroneous odometry estimates. Based on common error models, the odometry is estimated as normally distributed, allowing to directly obtain monotonically increasing covariance matrices ${}^{p}_{(i)}\boldsymbol{\Sigma}_{i+1}$ and ${}_{(i)}\boldsymbol{\Sigma}_i$, respectively. To define uncertainty propagation, the pose ${}_{(i)}\boldsymbol{m}_{i+1}$ of link point L_{i+1} in coordinate frame $(\mathrm{CF})_i$ is defined as

$$ {}_{(i)}\boldsymbol{m}_{i+1} = {}_{(i)}\boldsymbol{p}_{i+1} \oplus {}_{(i+1)}\boldsymbol{m}_{i+1}, \tag{3.11} $$

where the composition operator $\oplus$ concatenates the transformation of the second pose ${}_{(i+1)}\boldsymbol{m}_{i+1}$ to the reference system *already transformed* by the first pose ${}_{(i)}\boldsymbol{p}_{i+1}$, as shown in Figure 3.14.

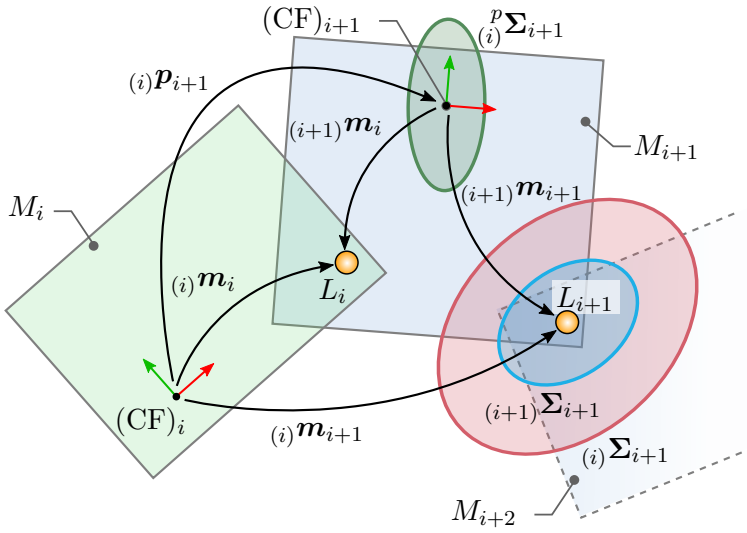


Figure 3.14: Composition of link point poses and uncertainties over multiple maps.

The uncertainty propagation is then obtained by a first-order TAYLOR expansion [SC86] of this equation, thus,

$$ {}_{(i)}\boldsymbol{m}_{i+1} \approx {}_{(i)}\bar{\boldsymbol{m}}_{i+1} + \boldsymbol{J}_1 \left({}_{(i)}\boldsymbol{p}_{i+1} - {}_{(i)}\bar{\boldsymbol{p}}_{i+1} \right) + \boldsymbol{J}_2 \left({}_{(i+1)}\boldsymbol{m}_{i+1} - {}_{(i+1)}\bar{\boldsymbol{m}}_{i+1} \right), \tag{3.12} $$

$$ \text{with} \quad \boldsymbol{J}_1 = \frac{\partial({}_{(i)}\boldsymbol{p}_{i+1} \oplus {}_{(i+1)}\boldsymbol{m}_{i+1})}{\partial {}_{(i)}\boldsymbol{p}_{i+1}} \quad \text{and} \quad \boldsymbol{J}_2 = \frac{\partial({}_{(i)}\boldsymbol{p}_{i+1} \oplus {}_{(i+1)}\boldsymbol{m}_{i+1})}{\partial {}_{(i+1)}\boldsymbol{m}_{i+1}}. $$

Assuming uncorrelated noise sequences, which is given when the uncertainty is reset for each new map, the estimated covariance matrix of ${}_{(i)}\boldsymbol{m}_{i+1}$ is computed from eq. 3.12 as

$$ {}_{(i)}\boldsymbol{\Sigma}_{i+1} \approx \boldsymbol{J}_1 \, {}^{p}_{(i)}\boldsymbol{\Sigma}_{i+1} \boldsymbol{J}_1^{\mathrm{T}} + \boldsymbol{J}_{2\,(i+1)}\boldsymbol{\Sigma}_{i+1} \boldsymbol{J}_2^{\mathrm{T}}. \tag{3.13} $$

The transform between the robot pose ${}_{(k)}\boldsymbol{p}_{\mathrm{r}}$ on map M_k to the world origin $(\mathrm{CF})_0$, which is usually the origin of map M_0, can then be determined by

$$ {}_{(0)}\boldsymbol{p}_{\mathrm{r}} = {}_{(0)}\boldsymbol{p}_k \oplus {}_{(k)}\boldsymbol{p}_{\mathrm{r}} = {}_{(0)}\boldsymbol{p}_{\boldsymbol{v}[1]} \oplus {}_{(\boldsymbol{v}[1])}\boldsymbol{p}_{\boldsymbol{v}[2]} \oplus \ldots \oplus {}_{(\boldsymbol{v}[l-1])}\boldsymbol{p}_k \oplus {}_{(k)}\boldsymbol{p}_{\mathrm{r}}, \tag{3.14} $$

where $\boldsymbol{v} \in \mathbb{N}^l$ is a vector of length l which contains the map-indices of the shortest map path to $(\mathrm{CF})_k$ and the equation is evaluated from left to right. The path is computed by Dijkstra's algorithm [Dij59] based on the graph structure introduced in section 3.3.1.

Loop Search and Graph Optimization

As explained in previous sections, loop closures must be made in graph-based mapping approaches to prevent the estimation error from growing. The same applies to the presented approach for map management, as this allows for correcting the relative poses of the maps to each other. During the mapping process, all locations for potential link points with another map are marked by placing the robot in front of opened doors to detect the door posts. Later on, this will allow for topological loop closures. Environmental data, such as point clouds or camera images, are then saved as part of a *link point candidate* LC_i. By saving this data for every new map and link point created, existing link points and candidates can be checked for loop closures. Loop-closure detection and registration can then be conducted, e.g., vision-based [GO15; LM13] or Lidar-based as presented in section 3.2. The search space is adjusted based on the longest major axis of the confidence ellipse of ${}_{(0)}\boldsymbol{\Sigma}_i$, as in eq. 3.8. Similar to the approach described in section 3.2.1, this decreases the number of possible loop pairs to be checked. As illustrated in Figure 3.15, only a subset of link points and link point candidates will be checked for loop closure.

After successful loop detection, the graph is optimized by creating a cost function in the form of eq. 3.4. The error function is created based on the relative map poses ${}_{(i)}\boldsymbol{p}_{i+1}$, and the information matrices follow from inverting the covariance matrices ${}_{(0)}\boldsymbol{\Sigma}_i$. Graph optimization is implemented in C++ with the g2o framework [KGS+11].

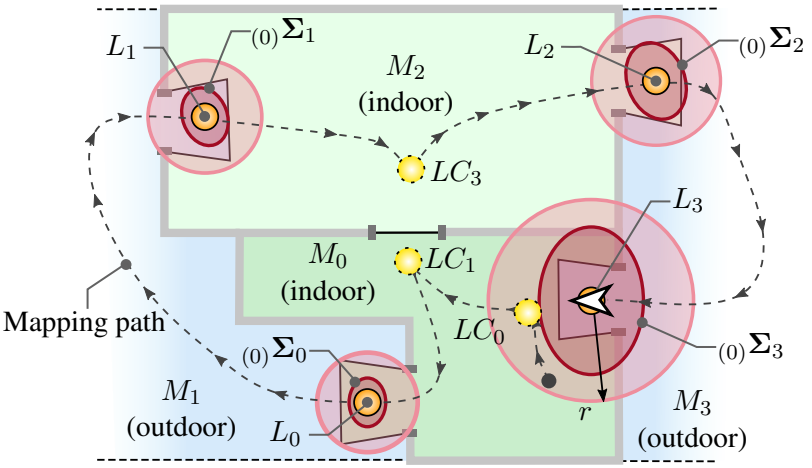


Figure 3.15: Global tracking of the robot's position along a mapping path to find eligible loop closures. The search radius r becomes larger when uncertainty increases. In this example, only the link point candidate LC_0 is checked for loop closure.

3.3.4 Results

The map-management system is evaluated on two real-world datasets consisting of small and large-scale indoor and outdoor environments with a total area of several hundred square meters. The data is collected with the mobile robot described in section 2.1.2, with the primary sensor modalities being the front and rear-facing RGB-D cameras, 3D Lidar, and EKF-fused inertial and wheel odometry. Again RTAB-Map is used as the basic SLAM method, which is especially suitable for this purpose due to its memory-management [LM11] and multi-session [LM18] capability and its applicability to multiple sensor modalities. However, like most SLAM methods, RTAB-Map needs to be configured differently for each type of environment. The most important settings and differences between the configuration for small and large-scale areas are shown in Table 3.3.

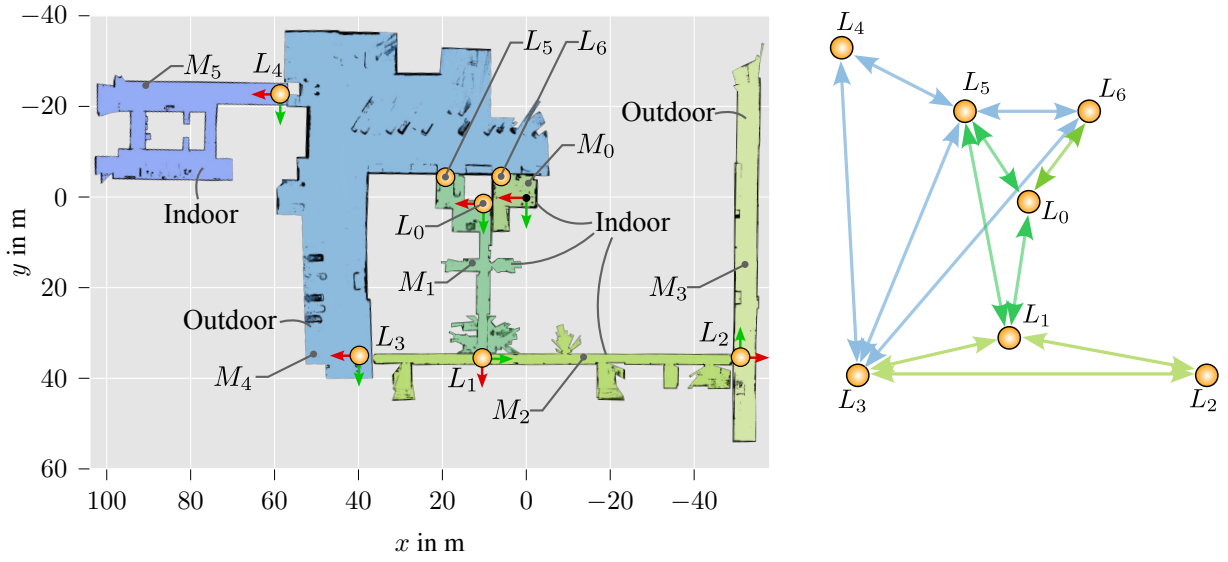
Parameter/Method	Small-range	Long-range
Depth source	RealSense D435	Velodyne VLP-16
Vis/FeatureType	GFTT/BRIEF	SIFT
Grid/RangeMax	10 m	30 m
Grid/MaxObstacleHeight	1.7 m	3.5 m
Mem/STMSize	30	60
Reg/Force3DoF	true	true
Grid/FromDepth	false	false
Rtabmap/TimeThr	750 ms	750 ms

Table 3.3: Different RTAB-Map configurations for small and large-scale areas used in the experimental evaluation

Both configurations are set to 2D-SLAM (Reg/Force3DoF) and real-time execution (Rtabmap/TimeThr). The long-range configuration is set to detect objects at longer distances (Grid/RangeMax) and height (Grid/MaxObstacleHeight) which is not necessary indoors and saves memory. The main difference between both configurations is the source of depth data, whose characteristics are according to the explanations in section 3.3.2. The decision about which configuration to use is solely based on the bounding box criterion (C3) introduced in section 3.3.2 since this indicates the change in area of consecutive environments. In both evaluations, the ground floor of a university building is mapped, with the first focusing on graph creation and the second on a comparison to the single-map approach with RTAB-Map.

Modular Mapping of a University Campus

The first evaluation run consists of indoor areas of two buildings and outdoor areas in between, mapped in a continuous path of 1,410 m length. A total of six maps are created with the proposed map-management approach, as shown in Figure 3.16.



(a) Metric occupancy maps, link points, and coordinate frames. Greenish maps are created based on the small-range configuration, blueish with the long-range configuration.

(b) Topological graph structure created based on the map structure in (a). Edge colors correlate with maps.

Figure 3.16: Mapped heterogeneous university campus. Link points are marked with orange circles.

All doors and environment changes are automatically detected, with each sub-map's appropriate SLAM configuration selected. Each sub-map is illustrated in color based on the respective mapping configuration. Note that the configuration setting does not distinguish between indoor and outdoor. Although M_3 is an outdoor map and M_5 is an indoor map, the small-range configuration is used in the first case and the long-range configuration in the second case. Due to the spatial dimensions of

the respective environments, this is the appropriate choice for these cases and error-free maps can be created. This demonstration shows the mapping result of a large area with indoor and outdoor maps of different shapes and appearances. In addition, maps can be easily added, modified, or deleted later. The topological structure (see Figure 3.16.b) also allows for efficient navigation over multiple maps.

Comparison to Single-Map Approach

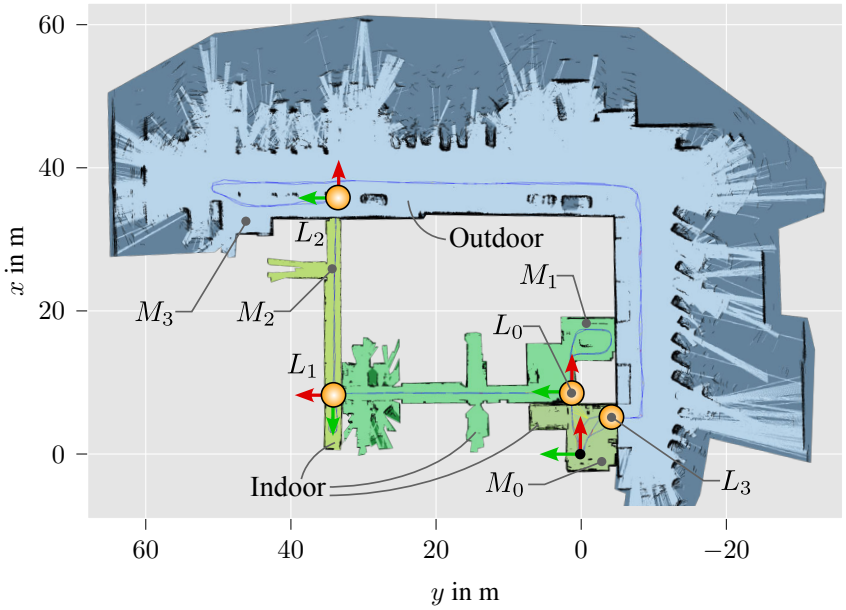
In the following, the proposed multi-map approach is compared to the single-map approach using RTAB-Map. For this, a large loop is traveled twice, resulting in a total path length of 506 m to detect loop closures, as shown in Figure 3.17. A total of three different mapping approaches are then conducted on the same input data: the proposed multi-mapping (Figure 3.17.a), a single map based on the small-range configuration (Figure 3.17.b), and a single map based on the long-range configuration (Figure 3.17.c).

Although all depicted approaches result in a consistent map, there are some advantages in separating the map. While the indoor area in Figure 3.17.b has about the same map quality compared to Figure 3.17.a, the outdoor area is covered by a smaller map with less range. Additionally, a mapping error occurs due to utilizing RGB-D data with only a few meters of reliable range measurements, resulting in a skewed graph at the bottom of the map. However, if the indoor areas are mapped with the long-range configuration, this leads to the errors shown in Figure 3.17.c. In this case, parts of the ceiling and door frames are detected as obstacles, leading to an erroneous occupancy map projection. All these map errors can be solved by using an appropriate configuration for the environment, as presented in Figure 3.17.a.

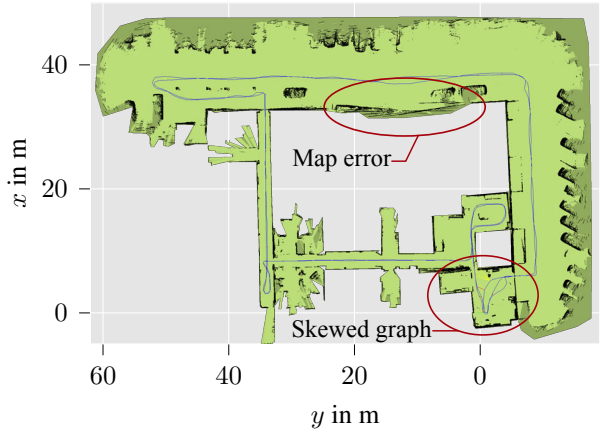
3.4 Conclusion

In this chapter, methods for improving mapping and localization in dynamic and heterogeneous environments are presented and evaluated. The popular RTAB-Map framework is extended with a trained classifier and registration procedure to enable Lidar-based loop-closing in addition to existing image-based methods. The method is validated on indoor and outdoor datasets from typical service robot deployment environments and the freely available KITTI benchmark. The results demonstrate an improvement in accuracy and robustness concerning environmental changes.

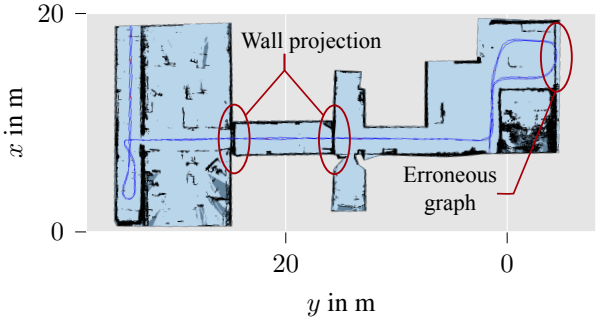
In addition, an approach for mapping large-scale environments of varying types with different SLAM configurations is presented. The appropriate SLAM method or configuration can be chosen by automatically analyzing environmental transitions and triggering new map creation. A metric-topological representation allows the linking of small specific maps instead of using computationally intensive large maps. Experiments show superior results compared to single map solutions and significantly enhanced map quality.



(a) Map-managing approach.



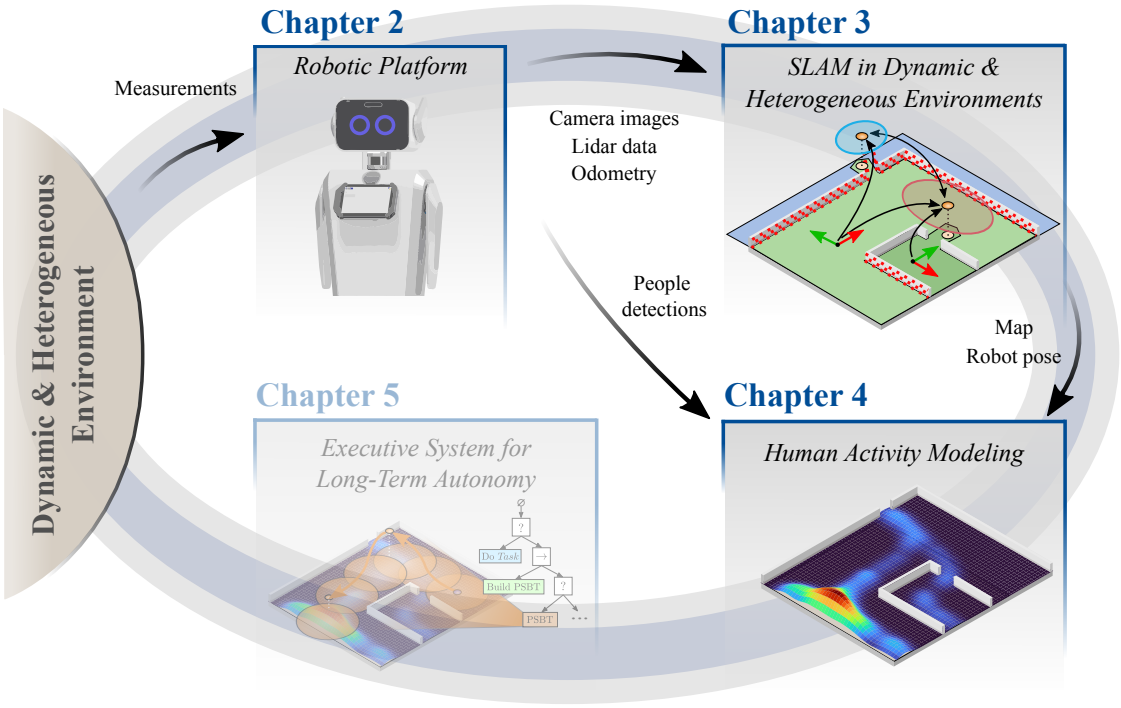
(b) Single map, small-range configuration.



(c) Single map, long-range configuration (cropped).

Figure 3.17: Maps created with large loops inside and outside of a building

4 Long-Term Human-Activity Modeling



Environment models are a crucial requirement for the (long-term) autonomy of mobile robots, as already shown in the last chapter. The ability for self-localization is a fundamental necessity. It creates the basis for further applications like navigation and task planning. Explicitly considering environmental information can help to improve localization – especially if the environments are heterogeneous. However, environmental changes and dynamics have an impact not only on the ability to localize but also on the applications based thereupon. Due to the direct relation of robotic service tasks to humans, human behavior represents an extensive influencing factor in these dynamics. In order to be socially accepted and not be perceived as disturbing, robots should adapt to this behavior and intelligently decide where they move and how to schedule and perform their tasks.

Accurate models of human activity, i.e., spatio-temporal occurrence, and movements of pedestrians can help robots to achieve this purpose, for example, by improving navigation [OSAR11], task planning [VYE+20], or human-centered task execution [RV12]. Human activity is subject to routines, and it is evident that, for example, the number of people in shopping malls or restaurants is different in the morning than at noon. As a result, it is crucial to consider temporal and spatial variability in these models for LTA applications.

This chapter presents a novel method for the long-term modeling of human activity based on an existing (pre-captured) map of the environment. The focus is on the use in the field of mobile robotics, which is characterized by varying dwell times of the robot at different locations. These variations lead to an inhomogeneous and sparse distribution of the measurement data, which is explicitly considered in the method.

After an overview of related work in section 4.1, section 4.2 introduces the method continuous pedestrian activity map (CoPA-Map) as a GAUSSIAN process (GP)-based regression model. Section 4.3 compares experimental results of CoPA-Map to several state-of-the-art baselines on two publicly available datasets.

The work in this chapter was in great part published at the peer-reviewed “IEEE/RSJ International Conference on Intelligent Robots and Systems" (IROS) 2022 [SS22].

4.1 Related Work

Historical knowledge of human behavior can be included in a map in different levels of detail and abstraction. The underlying approaches generally consider spatial or temporal variations or a combination of both. Purely spatial models without time dependency are presented in section 4.1.1. Approaches introduced in section 4.1.2 include a temporal component but focus on short-term predictions in the range of seconds. Extending this notion, section 4.1.3 presents models for long-term spatio-temporal human-activity prediction. An overall conclusion is given in section 4.1.4.

4.1.1 Spatial Modeling

There are two options to spatially encode the local variability of human behavior: discrete and continuous. Uniformly discretizing the environment is a straightforward way that results in a grid, where each cell is considered separately. Each cell may then represent an independent Poisson process [LDA11] updated by a Bayesian rule and can also include the directions, speed, and acceleration of pedestrians [NN19]. In [SR18], the authors introduce a directional grid map that probabilistically models long-term human motion through angular directions. Angular representations incorporating motion speed and partial observability are presented in [KMS+17]. These representations result in a vector field, which can be efficiently combined with a motion planner to achieve natural trajectories [PKM+17]. Other discretization techniques divide the map as a graph, for example, to predict corridor traversability [RRH+20], or consider separate semantic locations such as specific stores [KMKI20].

As argued by [OR12], *continuous* spatial maps have better properties than discrete models for robot navigation or sparse data distribution, as information from neighboring regions is not neglected. They apply spatially continuous GP models for occupancy mapping of static objects based on

range data. These non-parametric methods are kernel-based, can distinguish well between empty and occupied space, and can also capture nonlinear or obstructed patterns. These principles are subsequently used for spatially continuous navigational maps [OSAR11], trajectory prediction for flow field creation [TL08; ESR09; KLE11], or planning of socially acceptable motion [CH10]. The main drawback of GP-based methods is the adverse complexity scaling regarding the number of data points. Therefore, later works [RO16; GVD18] develop techniques to counteract this by kernel approximations or incremental updating.

4.1.2 Temporal Short-Term Modeling

The approaches above do not consider historical knowledge and do not model temporal changes in the environment. Incorporating short-term temporal changes of the map by observations can again be realized discretely by employing dynamic models separately for each cell. The paths of pedestrians or cyclists can then be predicted by an input-output MARKOV model [WAJF14], or individual MARKOV chains [SAL12].

Many methods exist for short-term trajectory prediction of pedestrians due to the strong relevance in autonomous driving or navigation planning of mobile robots in general [RPH+20]. These approaches are usually learning-based for complex environments with different participants instead of employing specific motion dynamics models. In this field, long short-term memory (LSTM) networks have become a widely popular modeling approach. By combining LSTM with contextual information from multiple weeks of data, the authors of [SYM+18] realize 3-DoF pedestrian pose trajectory predictions. In [SZDZ17] LSTM are combined with deep GPs to predict trajectories in crowded scenarios based on surveillance imagery. Instead of individual trajectories, the crowd density may be forecasted, as shown in [MYNU20] for prediction horizons of multiple seconds. A combination of an intention-aware filtering process with LSTM is presented in [HHS+20] to allow for robust predictions even under abnormal intention-changing scenarios.

4.1.3 Spatio-Temporal Long-Term Modeling

Most works in trajectory prediction focus on an immediate temporal horizon of a few seconds. However, modeling the environment changes over long periods can be beneficial for global navigation [VBR+22] and task planning purposes [HHK17]. Studies in the literature show that the use of LSTMs for long-term predictions is limited [VMS+19; Mel21]. LSTM-based predictions of occurrence counts or rates converge quickly to the mean, requiring specific methods for long-term modeling.

Predicting long-term environmental dynamics requires analyzing time series gathered over long periods. These may then be analyzed by seasonal windows, as in [VKS08] or encoded in seasonal patterns to predict the time of occurrence of human activities via VON-MISES distributions [CMHC17]. The time series of long-term temporal patterns usually combine trend, seasonal and

cyclic effects. Due to their predominant influence, approaches primarily focus on periodic changes, which can be modeled kernel-based [TR18] or with spectral analysis, e.g., by the frequency map enhancement (FreMEn) method [KFSD17]. FreMEn is a method for non-uniform frequency transforms with an application to mobile robotics and was initially developed to model the evolution of binary states over time, such as the occupancy of grid cells. Extensions have been made to model human activity quantitatively using spatially discrete POISSON processes for intensities [JWHK16] or predominant directions of human flow [MCD19]. Like purely spatial discrete models, restrictions apply since these methods only consider temporal variations [VKS08; CMHC17; TR18] or neglect interdependencies of separate spatial regions [KFSD17; JWHK16; MCD19].

To overcome this, continuous spatio-temporal models can represent correlations between different regions and sparse data inputs. For instance, based on a limited number of temporal periods, the authors of [ZM15] predict the demand for ambulances by a GAUSSIAN mixture model (GMM). GPs have been applied to the spread of disease prediction [SOR16], employing variational inference and periodicities obtained by general knowledge. An estimation of periodic parameters in a GP framework is realized in [JH18], also by a variational approach. The method is evaluated to forecast the time and locations of taxi pickups over multiple hours. In the field of mobile robotics, particularly human-activity recognition, the authors of [VYDK19] propose a spatio-temporal continuous model extending the aforementioned FreMEn approach. The model is based on a projection of data points to a circular space with subsequent clustering by GMMs and was later extended to incorporate human flow [VMS+19]. However, since clustering is performed directly on the data points (people detections), it is prone to erroneous predictions when the robotic system moves through the environment and collects varying amounts of data at different locations.

4.1.4 Conclusion

Like occupancy mapping of static obstacles, many approaches for human-activity modeling spatially discretize the environment. The resulting individual regions may then encode counts or rates of people or movement direction and velocities. A significant drawback of discrete modeling is that information from neighboring regions is neglected. However, this information may be beneficial in case of occlusions and sparse data distribution resulting from the robot's limited field of view. Continuous models can better represent these properties.

Models that encode temporal variability mainly focus on short-term predictions of human trajectories. LSTM-based approaches make up the majority but are inferior for long-term predictions. Long-term temporal effects, such as periodicities in human behavior, can be captured by frequency-based analyses to predict multiple hours and days in advance. In robotics, long-term spatio-temporal models primarily utilize discretizations. Only one method [VMS+19] makes spatio-temporally continuous predictions but focuses on stationary data collection.

To summarize, the state of research does not provide models that can predict human activity based on data collected by *mobile* robots in the *long term* in a *spatio-temporal continuous* manner. As

shown in related research areas, GPR can be used with sparse data distributions and account for a priori known information. The application of variational approaches allows the use in large-scale environments, which has been little investigated for human-activity models in the robotics domain.

4.2 Long-Term Spatio-Temporal Human-Activity Modeling

Following the research gap in terms of continuous models, this section presents a novel method for the long-term prediction of person occurrence as a spatio-temporally continuous map. The method, named continuous pedestrian activity map (CoPA-Map), aims at capturing the presence of humans in a robot's operational area. This data is then used to anticipate human presence in the future as a function of location and time (see Figure 4.1). Its main idea is to model human activity as a spatially and temporally varying rate function based on measurements collected by the mobile robot.

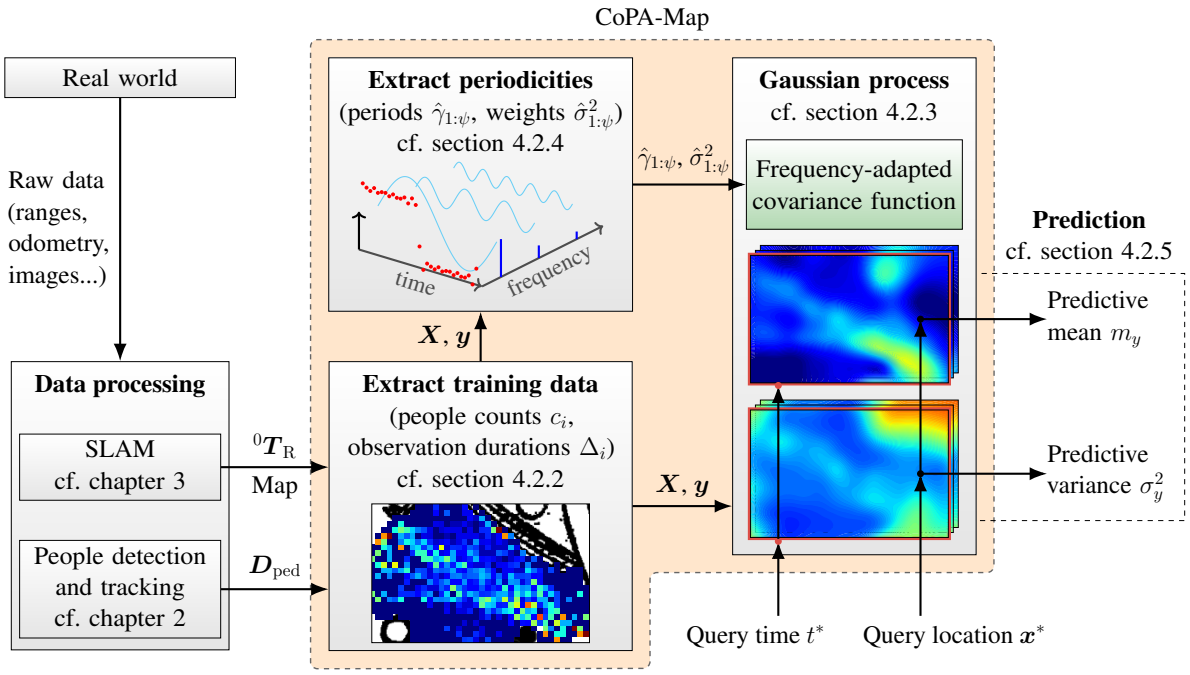


Figure 4.1: Schematic overview of the proposed model CoPA-Map.

The SLAM system provides a map of the environment and the robot's pose, given as a transformation ${}^0\boldsymbol{T}_\mathrm{R}$. This allows to transform the detected pedestrian data into an inertial (map) coordinate frame. The transformed data is converted by spatial and temporal binning to an input and output dataset. The search of the rate function is then formulated as a regression problem, which is solved in a GP framework. This allows continuous modeling with sparse data and provides an uncertainty estimate for each prediction that indicates areas of the input space that require further exploration or data collection. CoPA-Map exploits the idea that human activity is periodic in time. The number of people at different locations is subject to regularity, for example, based on the time of day, working

hours, or store opening hours. This characteristic can be assumed to influence the general time course of human activity much more than trends and cyclic patterns. Therefore, these are neglected, and the time-dependent person rate is specified to be subject to a number of ψ periodicities as prior information. A multidimensional covariance function is defined for the GP prior and initialized with parameters based on frequency analysis. Starting with a brief introduction to GAUSSIAN process regression in the following section, the modules of CoPA-Map are introduced as illustrated in Figure 4.1.

4.2.1 Gaussian Process Regression

For a dataset of n training inputs $\boldsymbol{X} = \left\{\boldsymbol{x}_i \in \mathbb{R}^d\right\}_{i=1}^n$ and observations $\boldsymbol{y} = \{y_i \in \mathbb{R}\}_{i=1}^n$ GAUSSIAN process regression (GPR) aims at inferring a latent function $f : \mathbb{R}^d \to \mathbb{R}$ via a noisy observation model

$$y_i = f(\boldsymbol{x}_i) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{4.1}$$

The GP is defined as a distribution over functions $\boldsymbol{f} = f(\boldsymbol{x}) \sim \mathcal{GP}(\nu_f(\boldsymbol{x}), k_f(\boldsymbol{x}, \boldsymbol{x}'))$ with mean function $\nu_f(\boldsymbol{x})$ and covariance function $k_f(\boldsymbol{x}, \boldsymbol{x}')$. Given a GAUSSIAN likelihood $p(\boldsymbol{y}|\boldsymbol{f})$, the posterior $p(\boldsymbol{f}|\boldsymbol{y})$ is inferred to obtain the predictive distribution $p(f_*|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_i^*) = \mathcal{N}\left(f_*|m(\boldsymbol{x}_i^*), \sigma^2(\boldsymbol{x}_i^*)\right)$ at a test point $\boldsymbol{x}_i^*$ with mean and variance respectively expressed as

$$m\left(\boldsymbol{x}_i^*\right) = k_f(\boldsymbol{x}_i^*, \boldsymbol{X}) \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{4.2}$$

$$\sigma^2\left(\boldsymbol{x}_i^*\right) = k_f(\boldsymbol{x}_i^*, \boldsymbol{x}_i^*) - k_f(\boldsymbol{x}_i^*, \boldsymbol{X}) \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right)^{-1} k_f(\boldsymbol{X}, \boldsymbol{x}_i^*), \tag{4.3}$$

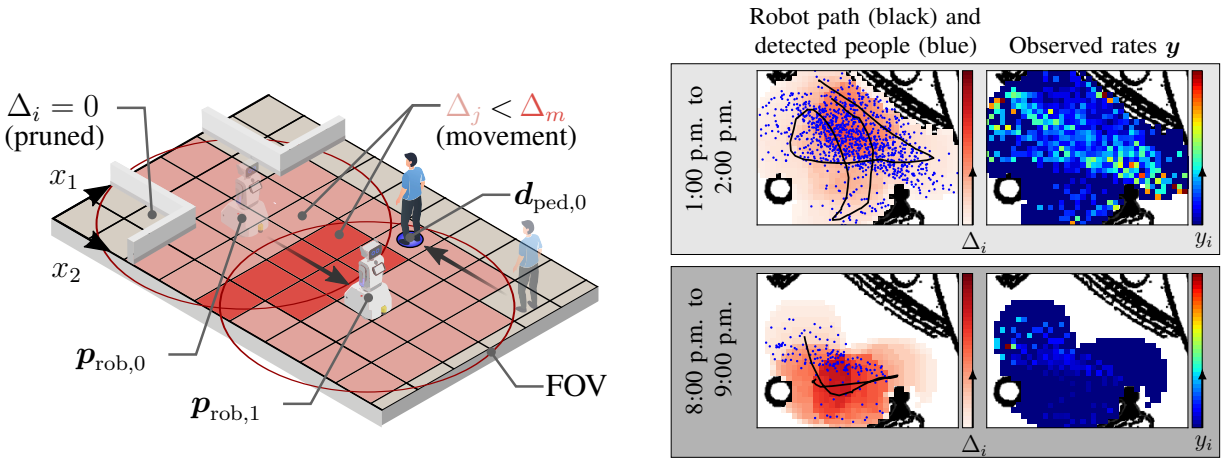
where $\boldsymbol{K}_{\boldsymbol{ff}} = k_f(\boldsymbol{X}, \boldsymbol{X})$. Optimizing the regression model then aims at finding hyperparameters inside the covariance matrix $\boldsymbol{K}_{\boldsymbol{ff}}$ by maximizing the marginal likelihood as

$$\log p(\boldsymbol{y}) = -\frac{n}{2} \log 2\pi - \frac{1}{2} \log \left|\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right| - \frac{1}{2} \boldsymbol{y}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}. \tag{4.4}$$

The required inversion of the $n \times n$ matrix $\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}$ gives rise to the most prominent weakness of standard GPs, which is the cubic complexity in the number of training inputs $\mathcal{O}(n^3)$ for the corresponding CHOLESKY decomposition. This adverse scaling limits their usability, especially for robotics applications and large datasets. A common approach to overcome this problem is to sparsely approximate the kernel matrix $\boldsymbol{K}_{\boldsymbol{ff}}$ using the NYSTRÖM low-rank representation $\boldsymbol{K}_{\boldsymbol{ff}} \approx \boldsymbol{K}_{\boldsymbol{fu}_f} \boldsymbol{K}_{\boldsymbol{u}_f \boldsymbol{u}_f}^{-1} \boldsymbol{K}_{\boldsymbol{fu}_f}^{\mathrm{T}}$. For this, a number of $m \ll n$ inducing points (or pseudo-inputs) must be chosen at locations $\boldsymbol{Z} = \left\{\boldsymbol{z}_i \in \mathbb{R}^d\right\}_{i=1}^m$ to represent the training data, decreasing the computational cost to $\mathcal{O}(m^2 n)$ [WS00]. The corresponding function values are denoted as $\boldsymbol{u}_f = f\left(\boldsymbol{Z}\right)$. As the quality of the approximation largely depends on the number and location of inducing inputs, it is suitable to treat the inducing points as hyperparameters and optimize their locations $\boldsymbol{Z}$ with respect to the marginal likelihood of eq. 4.4.

4.2.2 Extraction of Training Data

The model's input is formed based on a mobile robot with a sensor for people detection (e.g., as presented in section 2.1.3). A detected pedestrian is represented as a spatio-temporal point ${}_{(0)}\boldsymbol{d}_{\text{ped},k} = (x_{1,k}, x_{2,k}, t_k)^{\text{T}}$ in world coordinates (coordinate frame $(\text{CF})_0$) corresponding to a measurement taken at time t_k. For the transformation ${}^0\boldsymbol{T}_{\text{R}}$ to $(\text{CF})_0$, the robot is assumed to act in an environment with a known map and to be localized within this environment (cf. chapter 3). The goal then is to model human activity as an intensity function of space and time based on the data $\boldsymbol{D}_{\text{ped}} = (\boldsymbol{d}_{\text{ped},0}, \boldsymbol{d}_{\text{ped},1}, \ldots)$. For this, people are counted within a spatio-temporal domain $\mathbb{S}_i \subset \mathbb{R}^3$ so that the people count is defined as $c_i = |\{\boldsymbol{D}_{\text{ped}} \cap \mathbb{S}_i\}|$. By partitioning the environment into an evenly spaced grid of n cells, each domain $\mathbb{S}_i$ is created as a cell with a square spatial shape with edge length r_{s} and temporal resolution τ. Since the robot moves through the environment, each cell is visible for a different duration. This duration is calculated from the FOV of the sensor, which is approximated by a geometrical shape. For example, the projected 2D detection area of a 3D-Lidar-based detector can be described by a circle. This circle is pruned at known obstacles in the environment map based on a ray casting model, as shown in Figure 4.2.a.



(a) Robot movements lead to varying cell observation durations Δ_i. Each detected pedestrian is included in the respective cell as an increment.

(b) Exemplary input data of the model at midday and evening with real-world data collected in a shopping center from [BKIM13].

Figure 4.2: Illustration of the spatio-temporal model input and output data generation.

A people count $c_i \geq 0$ and observation duration $0 < \Delta_i \leq \tau$ is then assigned to each visible cell. Consequently, the robot's deployments over time generate the set of input data

$$\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{n} = \{(x_{1,i}, x_{2,i}, t_i)\}_{i=1}^{n}\,, \tag{4.5}$$

which are the center points of the spatio-temporal cells $\mathbb{S}_1 ... \mathbb{S}_n$. The corresponding targets are the observed rates

$$\boldsymbol{y} = \{y_i\}_{i=1}^{n} = \left\{ \frac{c_i}{\Delta_i} \right\}_{i=1}^{n} \tag{4.6}$$

of people in each cell.

Using relative rates instead of absolute counts is based on the following idea: Since a target value y_i can both be large due to a large c_i or a small Δ_i, it varies more smoothly at edges between areas with shorter and longer observation periods Δ_i. As people continuously move through the environment, areas with consistent values y_i indicate homogeneous activity, which merits greater weighting when optimizing the model. However, irregular spatial patterns of the values in $\boldsymbol{y}$ indicate either short observation durations or irregular occurrences of people, which in contrast, should be captured by a more prominent input noise in the likelihood function to give less weight to these areas during the model optimization. An exemplary distribution of input data resulting from real-world data is shown in Figure 4.2.b.

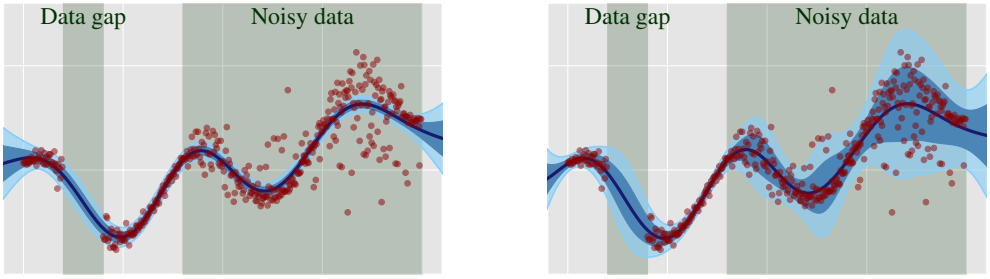
4.2.3 Gaussian Process Model

As indicated in section 4.2.1, GPR infers a latent function between input and output data. After introducing the required likelihood function in the following section, the covariance function of the GP prior to modeling the interrelationships between data points is defined. Finally, the posterior distribution is obtained by an approximation technique.

Likelihood Function

As part of the GPR, a likelihood function must be chosen to fit a model that represents the distribution of observations $\boldsymbol{y}$ best. In other words, the likelihood function represents the probability of observing the data points under a given hypothesis (e.g., that the data is normally distributed with parameters following from the GP). Count data, such as person occurrences, may be viewed as events from an inhomogeneous Poisson process [LDA11; JWHK16; JH18]. However, this requires strong assumptions on the independence of events (e.g., people cannot arrive in groups), considers discrete data instead of a continuous rate y_i, and the variance of the POISSON distribution is directly coupled to its rate parameter. In contrast, here the rate $y_i \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma_i^2)$ is considered to be normally distributed with input-dependent noise σ_i^2 to realize the aforementioned spatio-temporal dependency of input weighting. This noise parameter can be defined independently from the latent mean function $f(\boldsymbol{x}_i)$ and leads to a heteroscedastic GP model. Heteroscedastic GPs have the advantage that data gaps and inhomogeneous data distributions have stronger influence on the predictive variance, exemplified in Figure 4.3.

To learn the noise parameter from data, it is defined as $\sigma_i^2 = \zeta\left(g(\boldsymbol{x}_i)\right)$, where $\boldsymbol{g} = g(\boldsymbol{x}) \sim \mathcal{GP}(\nu_g(\boldsymbol{x}), k_g(\boldsymbol{x}, \boldsymbol{x}'))$ is a second latent function that is also modeled by a GP. The function



(a) GAUSSIAN likelihood (b) Heteroscedastic GAUSSIAN likelihood

Figure 4.3: Exemplary 1D dataset with gaps and noise, fitted with a GP model with different likelihoods.

$\zeta(x) : \mathbb{R} \to \mathbb{R}_+$ is a link function to guarantee positive values for the noise parameter, defined as the softplus function $\zeta(x) = \log(\exp(x) + 1)$ in the present case. Resultingly, with the common presumption of a zero-mean GP model, the full model is defined as

$$y_i \sim \mathcal{N}(f(\boldsymbol{x}_i), \zeta(g(\boldsymbol{x}_i))), \quad f(\boldsymbol{x}) \sim \mathcal{GP}(0, k_f(\boldsymbol{x}, \boldsymbol{x}')), \quad g(\boldsymbol{x}) \sim \mathcal{GP}(0, k_g(\boldsymbol{x}, \boldsymbol{x}')). \tag{4.7}$$

Although the latent function $f(\boldsymbol{x}_i)$ may result in negative values, the rescaled predictive output of a tuned model contained very few zero-crossings on all tested datasets, making it sufficient to set negative values of the model output to zero for predictions.

Covariance Functions

To fully describe the GP model, the covariance functions k_f and k_g of both latent models must be defined. A covariance function (or *kernel*) allows encoding of prior beliefs about the latent function of interest and can be viewed as a measure of how *similar* two functions are. Different suitable covariance functions can also be connected as compositions. [Duv14]

To represent interdependencies of human activity through $f(\boldsymbol{x})$, each data point is separated into its spatial components $x_{\mathrm{s}1}, x_{\mathrm{s}2} \in \mathbb{R}$ and temporal component $x_{\mathrm{t}} \in \mathbb{R}$ and the following covariance function is defined:

$$k_f(x_{\mathrm{s}1}, x_{\mathrm{s}2}, x_{\mathrm{t}}, x'_{\mathrm{s}1}, x'_{\mathrm{s}2}, x'_{\mathrm{t}}) = k_{\mathrm{s}}(x_{\mathrm{s}1}, x'_{\mathrm{s}1})\, k_{\mathrm{s}}(x_{\mathrm{s}2}, x'_{\mathrm{s}2})\, k_{\mathrm{t}}(x_{\mathrm{t}}, x'_{\mathrm{t}}). \tag{4.8}$$

This multidimensional product kernel connects a spatial covariance function k_{s} with a temporal covariance function k_{t}, resulting in a prior over functions varying across all three dimensions. Combining separate kernels that are defined on different inputs of the covariance functions is a common way to ensure that the resulting kernel will have high value only if all base kernels have high value (AND-operation). Additionally, if all of these base kernels are stationary (i.e., only depending on the absolute difference $|x - x'|$ instead if the inputs themselves) the resulting kernel

is again stationary, which eases the optimization process. As the spatial kernel, the Matérn-$5/2$ covariance function

$$k_{\mathrm{s}}(x, x') = \sigma_{\mathrm{s}}^2 \left(1 + \frac{\sqrt{5}|x - x'|}{l_{\mathrm{s}}} + \frac{5|x - x'|^2}{3l_{\mathrm{s}}^2}\right) \exp\left(-\frac{\sqrt{5}|x - x'|}{l_{\mathrm{s}}}\right) \tag{4.9}$$

is chosen, where the length scale l_{s} and variance σ_{s}^2 are hyperparameters. This type of covariance function is a common choice to model structural correlations, as it provides a good balance between smoothness and capturing sudden changes [KK13].

The temporal periodicity of $f(\boldsymbol{x})$, which commonly represents human-activity patterns, is realized by a periodic kernel [Mac+98], that is defined as a sum of trigonometric functions

$$k_{\mathrm{t}}(x, x') = \sum_{i=0}^{\psi} \sigma_{\mathrm{t},i}^2 \exp\left(-\frac{1}{2}\frac{\sin^2\left(\gamma_i^{-1}|x - x'|\right)}{l_{\mathrm{t},i}^2}\right) \tag{4.10}$$

where the variances $\sigma_{\mathrm{t},i}^2$, periods γ_i and length scales $l_{\mathrm{t},i}$ are hyperparameters and ψ is the total number of periodic components. The variances $\sigma_{\mathrm{t},i}^2$ determine the overall influence of the specific component and $l_{\mathrm{t},i}$ controls the smoothness. As illustrated in Figure 4.4, the multiplication of the spatial (here one-dimensional) and the temporal component results in a smooth and periodic prior. Combining the kernels in this way allows for automatic relevance determination (ARD) during optimization because the length scale (relevance) of the spatial and temporal dimensions are estimated separately.

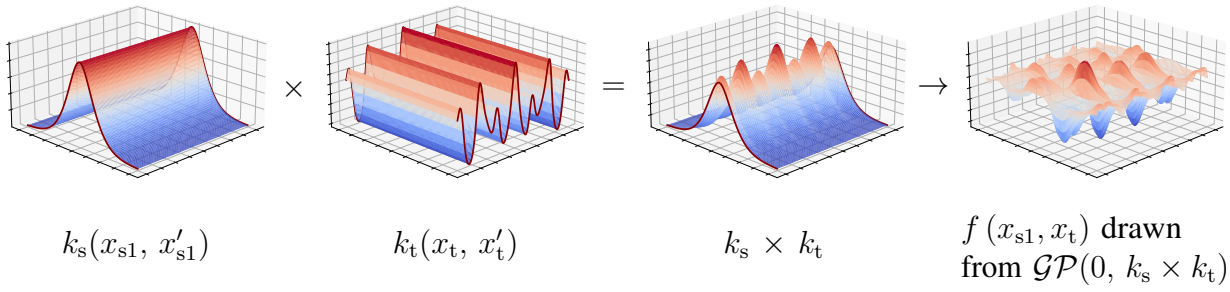


Figure 4.4: A product of one-dimensional spatial and temporal kernels gives rise to a spatio-temporal prior over functions.

Regarding human activity, the kernel k_f thus represents two important properties:

1.) Spatial continuity, i.e., if people are seen at a specific location, it is more likely to see people at very close locations. Since humans continuously move through space, this property is desirable to model.

2.) Temporal periodicity, i.e., when people are seen repeatedly at a specific location (e.g., every morning at an entrance), people are expected to be there in the future at that time of day.

Opposed to the problem-specific definition of k_f, the kernel k_g corresponding to the latent function $g(\boldsymbol{x})$ is realized by a radial-basis function (RBF) kernel

$$k_g\left(x, x'\right) = \sigma_g^2 \exp\left(-\frac{\left(x - x'\right)^2}{2l_g^2}\right) \tag{4.11}$$

to let the predictive variances of different areas align for larger prediction horizons.

Posterior approximation

In a model with multiple latent functions, the marginal likelihood $p(\boldsymbol{y})$ is not analytically tractable, which prevents the application of eq. 4.4 and requires posterior approximations. Instead of calculating the intractable posterior $p(\boldsymbol{f}, \boldsymbol{g}|\boldsymbol{y})$, it can be lower bounded with variational distributions $q(\boldsymbol{f})$ and $q(\boldsymbol{g})$ of known form (e.g., GAUSSIAN), a technique called *variational inference*. Compared to other techniques such as MARKOV Chain Monte Carlo (MCMC) sampling, variational inference has significant advantages in computational complexity while maintaining desirable approximate results [Tit09; LOSC20]. Its main principle is the estimation of the parameters of $q(\boldsymbol{f})$ and $q(\boldsymbol{g})$ by minimizing their distance to the true posterior distribution $p(\boldsymbol{f}, \boldsymbol{g}|\boldsymbol{y})$ measured by the Kullback-Leibler-divergence $\mathrm{KL}\left(q(\boldsymbol{f})q(\boldsymbol{g}) \parallel p(\boldsymbol{f}, \boldsymbol{g}|\boldsymbol{y})\right)$. Assuming that the latent functions $\boldsymbol{f}$ and $\boldsymbol{g}$ are a priori independent for each data point, Saul et al. [SHVL16] derive the variational lower bound

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{n} \int q\left(\boldsymbol{f}_i\right) q\left(\boldsymbol{g}_i\right) \log p\left(\boldsymbol{y}_i \mid \boldsymbol{f}_i, \boldsymbol{g}_i\right) \mathrm{d}\boldsymbol{f}_i \, \mathrm{d}\boldsymbol{g}_i \\ &- \mathrm{KL}\left(q\left(\boldsymbol{u}_f\right) \parallel p\left(\boldsymbol{u}_f\right)\right) - \mathrm{KL}\left(q\left(\boldsymbol{u}_g\right) \parallel p\left(\boldsymbol{u}_g\right)\right).\end{aligned} \tag{4.12}$$

This bound leverages sparse approximations (cf. section 4.2.1) to calculate sparse approximate posteriors as normal distributions $q(\boldsymbol{u}_f) = \mathcal{N}\left(\boldsymbol{u}_f|\boldsymbol{\mu}_f, \boldsymbol{S}_f\right)$ and $q(\boldsymbol{u}_g) = \mathcal{N}\left(\boldsymbol{u}_g|\boldsymbol{\mu}_g, \boldsymbol{S}_g\right)$ over inducing functions $\boldsymbol{u}_f$ and $\boldsymbol{u}_g$. For $q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}_f, \boldsymbol{\Sigma}_f)$ and $q(\boldsymbol{g}) = \mathcal{N}(\boldsymbol{g}|\boldsymbol{m}_g, \boldsymbol{\Sigma}_g)$ follows

$$\boldsymbol{m}_f = \boldsymbol{K}_{\boldsymbol{f}\boldsymbol{u}_f} \boldsymbol{K}_{\boldsymbol{u}_f\boldsymbol{u}_f}^{-1} \boldsymbol{\mu}_f, \tag{4.13}$$

$$\boldsymbol{\Sigma}_f = \boldsymbol{K}_{\boldsymbol{f}\boldsymbol{f}} + \boldsymbol{K}_{\boldsymbol{f}\boldsymbol{u}_f} \boldsymbol{K}_{\boldsymbol{u}_f\boldsymbol{u}_f}^{-1} (\boldsymbol{S}_f - \boldsymbol{K}_{\boldsymbol{u}_f\boldsymbol{u}_f}) \boldsymbol{K}_{\boldsymbol{u}_f\boldsymbol{u}_f}^{-1} \boldsymbol{K}_{\boldsymbol{u}_f\boldsymbol{f}}, \tag{4.14}$$

$$\boldsymbol{m}_g = \boldsymbol{K}_{\boldsymbol{g}\boldsymbol{u}_g} \boldsymbol{K}_{\boldsymbol{u}_g\boldsymbol{u}_g}^{-1} \boldsymbol{\mu}_g, \tag{4.15}$$

$$\boldsymbol{\Sigma}_g = \boldsymbol{K}_{\boldsymbol{g}\boldsymbol{g}} + \boldsymbol{K}_{\boldsymbol{g}\boldsymbol{u}_g} \boldsymbol{K}_{\boldsymbol{u}_g\boldsymbol{u}_g}^{-1} (\boldsymbol{S}_g - \boldsymbol{K}_{\boldsymbol{u}_g\boldsymbol{u}_g}) \boldsymbol{K}_{\boldsymbol{u}_g\boldsymbol{u}_g}^{-1} \boldsymbol{K}_{\boldsymbol{u}_g\boldsymbol{g}}. \tag{4.16}$$

Training the model is then realized by minimizing $-\mathcal{L}$ with respect to the variational parameters $\boldsymbol{\mu}_{f,g}$ and $\boldsymbol{S}_{f,g}$ as well as the hyperparameters in the covariance matrices $\boldsymbol{K}_{**}$.

4.2.4 Initialization of Hyperparameters

Due to the dependence on specific inducing point locations and hyperparameter guesses, the objective function of the lower bound (eq. 4.12) is prone to local minima. A major influencing factor is the initial guess of the hyperparameters of the temporal kernel k_t (eq. 4.10), which are the periods γ_i and variances $\sigma^2_{\mathrm{t},i}$. As shown in [TR18], initializing these parameters with frequency-analysis-based methods can reduce optimization time and the quality of the approximation. CoPA-Map includes a method to obtain the distinct temporal periods of a spatial domain based on non-uniform frequency analysis and a subsequent clustering step. The proposed initialization method (Algorithm 4.1) builds on the idea [KFSD17] of transferring the time-dependent activities at different locations into the frequency spectrum and making an approximation via a FOURIER series with a reduced number of components. Figure 4.5 shows a schematic overview of the main steps of the algorithm.

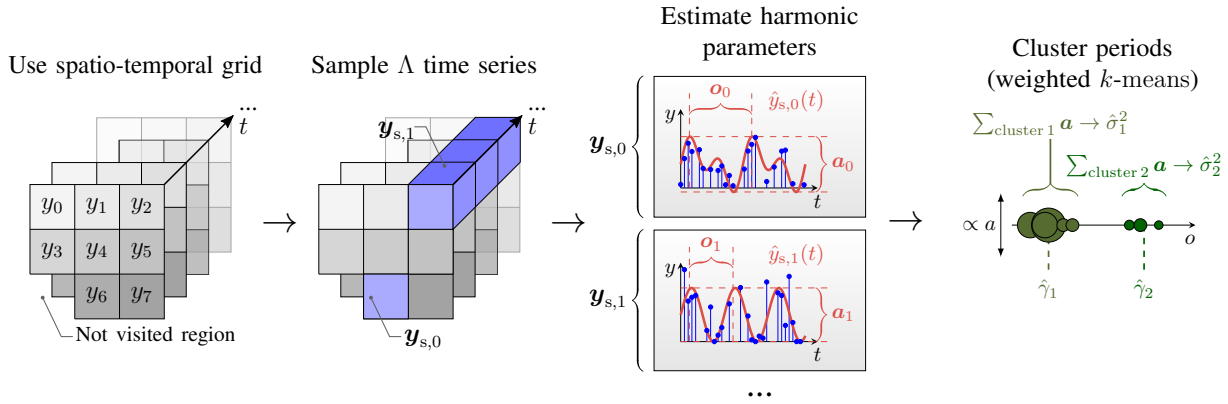


Figure 4.5: Periodic hyperparameter initialization routine.

The required time series for each spatial region (i.e., cell with shape $r_\mathrm{s} \times r_\mathrm{s}$) results by squashing the cells $\mathbb{S}_i$ of the spatio-temporal grid along the temporal dimension, leading to a time series of length J with values $\boldsymbol{y}_\mathrm{s} = (y_{\mathrm{s},0}, \cdots, y_{\mathrm{s},J-1}) \subset \boldsymbol{y}$ sampled at time-points $\boldsymbol{t}_\mathrm{s} = (t_{\mathrm{s},0}, \cdots, t_{\mathrm{s},J-1})$ for each spatial cell. A subset $\mathbb{T} = \big\{(\boldsymbol{t}_{\mathrm{s},0}, \boldsymbol{y}_{\mathrm{s},0}), \cdots, (\boldsymbol{t}_{\mathrm{s},\Lambda-1}, \boldsymbol{y}_{\mathrm{s},\Lambda-1})\big\}$, containing a predefined number of Λ time series, is then taken by roulette wheel selection, where each time series is given a weight of its total counts over all time steps. Hence, spatial regions with high activity are more likely to be selected, but regions with lower activity are not excluded altogether. Due to the robot's movement, certain regions might not be visited for some time, resulting in a non-equidistant distribution of the time samples $\boldsymbol{t}_\mathrm{s}$. Therefore, the conversion of each time series to the frequency domain is made by the non-uniform discrete FOURIER transform [DR93]

$$\mathrm{NUDFT}(\boldsymbol{t}_\mathrm{s}, \boldsymbol{y}_\mathrm{s}, \boldsymbol{o}): \quad \xi_k = \sum_{j=0}^{J-1} y_{\mathrm{s},j} \exp\left(-i\frac{2\pi}{T o_k} t_{\mathrm{s},j}\right), \quad 0 \leq k \leq J-1, \tag{4.17}$$

with signal duration T and candidate period $o_k \in \boldsymbol{o}$. The set of candidate periods $\boldsymbol{o}$ is defined to contain equally spaced periods within an interval (e.g., between one hour and seven days). The signal is iteratively reconstructed with a reduced number b of the most prominent frequency components as

$$\Psi(\boldsymbol{\xi}_b, \boldsymbol{o}_b): \quad \hat{y}_{\mathrm{s},j} = |\xi_0| + \sum_{u=1}^{b} |\xi_u| \cos\left(\frac{2\pi}{o_u} t_{\mathrm{s},j} + \arg(\xi_u)\right) \tag{4.18}$$

for each region and compared to the original signal via five-fold cross validation (Algorithm 4.1, lines 2 to 9). The algorithm requires an upper limit $\psi_{\max}$ of periods to check and a scaling factor $\sigma^2_{\max}$ as the maximum variance.

The total number of periods ψ of the whole domain is then calculated as the mean of all domain-specific numbers of periods (line 13). The period's respective values are calculated by weighted k-means clustering, with the complex magnitudes as weights (line 14). Weighting ensures that locations with a recurring number of people are more influential than cells with less activity.

Algorithm 4.1: Initialize the hyperparameters of the periodic kernel k_{t} (cf. eq. 4.10)

Input : Set containing Λ time series $\mathbb{T}$, set of candidate periods $\boldsymbol{o}$
Output : Number of periods ψ, set of estimated periods $\hat{\gamma}_{1:\psi}$, set of variances (weights) for each period $\hat{\sigma}^2_{1:\psi}$
Parameter : Upper limit for number of periods $\psi_{\max}$, maximum variance $\sigma^2_{\max}$

1 **for** $l = 0$ to $\Lambda - 1$ **do**
2 Repeat lines 3 – 9 as cross validation for $i = 1..5$
3 Split $\boldsymbol{y}_{\mathrm{s},l}$ into contiguous train/test sets $\boldsymbol{y}^{\mathrm{tr}}_{\mathrm{s}}/\boldsymbol{y}^{\mathrm{ts}}_{\mathrm{s}}$
4 $\boldsymbol{\xi}_i \leftarrow \mathrm{NUDFT}(\boldsymbol{t}^{\mathrm{tr}}_{\mathrm{s}}, \boldsymbol{y}^{\mathrm{tr}}_{\mathrm{s}}, \boldsymbol{o})$ `// eq. 4.17. Complex components in freq. domain`
5 **for** $b = 0$ to $\psi_{\max}$ **do**
6 $\boldsymbol{\xi}_{i,b} \leftarrow b$ most prominent complex numbers in $\boldsymbol{\xi}_i$ w.r.t. magnitude
7 $\boldsymbol{o}_{i,b} \leftarrow$ Set of periods, corresponding to $\boldsymbol{\xi}_{i,b}$
8 $\hat{\boldsymbol{y}}_b \leftarrow \Psi(\boldsymbol{\xi}_{i,b}, \boldsymbol{o}_{i,b})$ `// eq. 4.18. Reconstruct reduced signal`
9 $e_{i,b} \leftarrow \mathrm{RMSE}(\hat{\boldsymbol{y}}_b, \boldsymbol{y}^{\mathrm{ts}}_{\mathrm{s}})$
10 $i_l, b_l \leftarrow \arg\min_{i,b}(e_{1,0}, ..., e_{5,\psi_{\max}})$
11 $\boldsymbol{a}_l \leftarrow$ Element-wise magnitudes of $\boldsymbol{\xi}_{i_l,b_l}$
12 $\boldsymbol{o}_l \leftarrow$ Periods, corresponding to $\boldsymbol{\xi}_{i_l,b_l}$
13 $\psi \leftarrow \lfloor \mathrm{Mean}(\{b_0, \cdots, b_{\Lambda-1}\}) \rfloor$
14 $\hat{\gamma}_{1:\psi} \leftarrow$ Obtain k-means centroids with $k = \psi$ using $\boldsymbol{o}_{0:\Lambda-1}$ with weights $\boldsymbol{a}_{0:\Lambda-1}$
15 $\hat{\sigma}^2_{1:\psi} \leftarrow$ Sum weights $\boldsymbol{a}_{0:\Lambda-1}$ in clusters and normalize to $\left[0, \sigma^2_{\max}\right]$

Another factor influencing the model quality is the positioning of the inducing points $\boldsymbol{Z}$, as this directly affects the approximation of the covariance matrix $\boldsymbol{K}_{\boldsymbol{ff}}$. Although the inducing points are treated as hyperparameters and are modified during optimization, proper initialization shortens the time to find satisfactory solutions. Based on a user-defined ratio $\alpha_{\mathrm{ind}} \in (0, 1]$, the number of inducing points is selected as $m = \lfloor \alpha_{\mathrm{ind}} n \rfloor$. Their (spatio-temporal) location is then determined

via k-means clustering ($k = m$) of the spatio-temporal training inputs $\boldsymbol{X}$, where each input point is weighted by its observation time Δ_i. By weighting the inputs, the initial inducing points $\hat{\boldsymbol{Z}}$ are placed primarily at locations that have been observed for extended periods, providing more reliable data.

4.2.5 Model Optimization and Inference

Given the training inputs $\boldsymbol{X}$, observations $\boldsymbol{y}$, and initial hyperparameter configuration, the introduced model (eq. 4.7) is optimized based on the variational lower bound (eq. 4.12). Subsequently, it may be queried for new data inputs $\boldsymbol{X}^*$. The procedure of both steps is described in the following.

Model Optimization

By introducing latent inducing locations $\boldsymbol{Z}$, cubic scaling regarding the number n of training inputs is avoided during optimization (cf. section 4.2.1). When the covariance matrices of the variational distributions $\boldsymbol{S}_f, \boldsymbol{S}_g$ are defined by CHOLESKY decompositions $\boldsymbol{S}_f = L_{\boldsymbol{f}} L_{\boldsymbol{f}}^{\mathrm{T}}$ and $\boldsymbol{S}_g = L_{\boldsymbol{g}} L_{\boldsymbol{g}}^{\mathrm{T}}$, optimizing the lower bound $\mathcal{L}$ (eq. 4.12) scales with $\mathcal{O}(nm^2 + 2nm)$ [SHVL16]. Therefore, especially for $m \ll n$, model optimization is significantly sped up compared to the standard GPR case. In the present case, this can be realized by choosing $\alpha_{\mathrm{ind}} \ll 1$.

Model optimization is then executed for three types of parameters:

1. Variational parameters for $q(\boldsymbol{u}_f) = \mathcal{N}\left(\boldsymbol{u}_f | \boldsymbol{\mu}_f, \boldsymbol{S}_f\right)$ and $q(\boldsymbol{u}_g) = \mathcal{N}\left(\boldsymbol{u}_g | \boldsymbol{\mu}_g, \boldsymbol{S}_g\right)$,
2. the length-scale, variance, and periodicity hyperparameters of the kernels k_f and k_g, and
3. the location of inducing inputs $\boldsymbol{Z}$.

Two different optimization techniques based on stochastic gradient descent (SGD) are utilized for the different parameter types. The variational parameters are optimized using the natural-gradient method since the inherent minimization of KL-divergence integrates well with the variational framework and leads to fast convergence [SEH18]. The kernel hyperparameters and inducing point locations are optimized with the Adam optimizer [KB14]. Steps of both optimizers are executed alternately, with the learning rate decaying linearly for the first 100 steps. The n integrals as part of the lower bound $\mathcal{L}$ are solved by two-dimensional HERMITE-GAUSS quadratures. As the methods utilize SGD, the optimization is separated into mini-batches for increased efficiency.

Predicting with the Model

After maximization of the variational lower bound and optimization of hyperparameters, predictions with the model can be made via its predictive distribution. For arbitrary new data inputs $\boldsymbol{X}^* = \{\boldsymbol{x}_i^*\}_{i=1}^{n^*}$ the predictive distribution is given as

$$p\left(\boldsymbol{y}_i^* \mid \boldsymbol{y}_i, \boldsymbol{x}_i\right) = \int p\left(\boldsymbol{y}_i^* \mid \boldsymbol{f}_i^*, \boldsymbol{g}_i^*\right) q\left(\boldsymbol{f}_i^*\right) q\left(\boldsymbol{g}_i^*\right) \mathrm{d}\boldsymbol{f}_i^* \, \mathrm{d}\boldsymbol{g}_i^*. \tag{4.19}$$

This analytically intractable integral can be computed using HERMITE-GAUSS quadrature to obtain the predictive mean $m_y(\boldsymbol{x}_i^*)$ and variance $\sigma_y^2(\boldsymbol{x}_i^*)$. The specific values of the predictive mean depend on the chosen spatial resolution r_s and temporal resolution τ of the input grid. For a subset $\boldsymbol{X}' \subset \boldsymbol{X}^*$ of finite extend (e.g., the FOV of the robot and a given duration), the expected number of people can then be calculated as a point estimate $\frac{1}{r_\mathrm{s}^2 \tau} \int m_y \, \mathrm{d}\boldsymbol{X}'$. Model outputs of the mean m_y and standard deviation σ_y on an exemplary dataset (same as for Figure 4.2.b) for two points in time are shown in Figure 4.6. The predictive uncertainty σ_y increases outside the visible areas and in regions with high variability in human activity. Consequently, in addition to indicating predictive uncertainty, this suggests areas where further data collection might be useful.

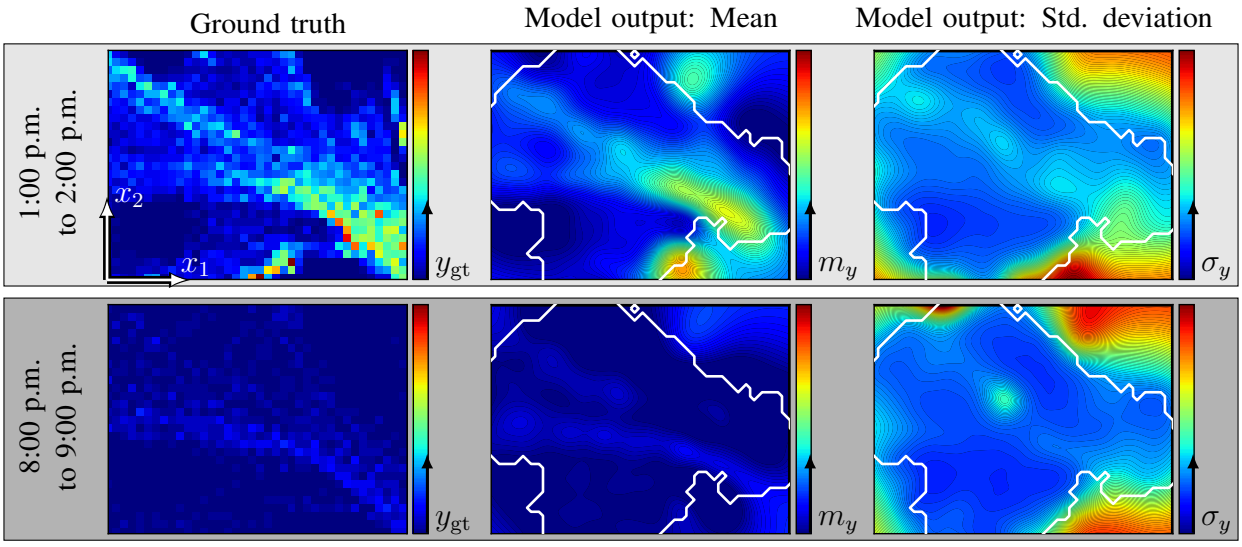


Figure 4.6: Model output compared to ground truth values of human occurrence rate. Only the areas within the white boundaries were visible and used for training.

4.3 Results

CoPA-Map is evaluated on two real-world datasets (section 4.3.1) in comparison to different state-of-the-art methods (section 4.3.2). Besides the evaluation of the hyperparameter initialization

routine in section 4.3.3 and the predictive error in section 4.3.4, one focus is on the quantitative assessment of the model as a basis for navigation tasks (section 4.3.5). All experiments are performed with the same parameterization: number of sampled time series $\Lambda = 10$, maximum number of periods $\psi_{\max} = 10$, maximum variance of each periodic kernel component $\sigma^2_{\max} = 0.95$, and ratio between inducing points and data points $\alpha_{\text{ind}} = 0.02$. The value for α_{ind} is empirically chosen to obtain a good balance between computational speed and prediction quality on the evaluated datasets. The initialization routine (Algorithm 4.1) was done with a fixed grid resolution of $5\,\text{m} \times 5\,\text{m} \times 60\,\text{min}$, whereas the grid resolution resulting in $\boldsymbol{X}$ was varied for different experiments (square spatial shape, respectively specified as $r_\text{s} \times \tau$). The method is implemented in Python based on the GPflow library [GWN+17] to perform the training and inference GPU-based.

4.3.1 Datasets

The model is evaluated on two publicly available long-term datasets of real-world pedestrian detections. Both datasets represent typical human-centered environments but vary in activity patterns and the average number of pedestrians. Occupancy maps of both environments and exemplary detections $\boldsymbol{D}_{\text{ped}}$ are shown in Figure 4.7 with density visualization.

ATC Dataset [BKIM13]: This dataset contains measurements of tracked pedestrians in a shopping center in Osaka, Japan, covering an area of ca. $900\,\text{m}^2$. Data collection was performed with several permanently mounted 3D range sensors every week on Wednesdays and Sundays, resulting in 92 days. The data is downsampled to a detection rate of 0.5 Hz, resulting in an average of about 1,700 entries per square meter and day. For evaluation, a subset of ten Wednesdays for training and four days for testing is used.

Office Dataset [MCD21]: The second dataset contains tracks of people based on measurements by a single stationary 3D Lidar sensor in an office environment of the University of Lincoln, England, covering an area of ca. $85\,\text{m}^2$ with averagely about 300 entries per square meter and day. The dataset consists of 22 consecutive days, of which ten weekdays are used for training and five weekdays for testing.

Since both datasets contain measurements by stationary sensors, data acquisition by a moving robotic system must be simulated. Nine robot trajectories per dataset are manually defined with an average motion speed of $0.5\,\text{m}\,\text{s}^{-1}$ and stationary intermediate stops. Only the measurements within the FOV of the simulated robot are processed, which is again (cf. section 2.1.2, section 4.2.2), defined by a circle with a fixed radius. The circle is pruned based on the known occupancy maps of the environments to filter out pedestrians obstructed by static obstacles, as depicted in Figure 4.7 with exemplary observation durations Δ_i.

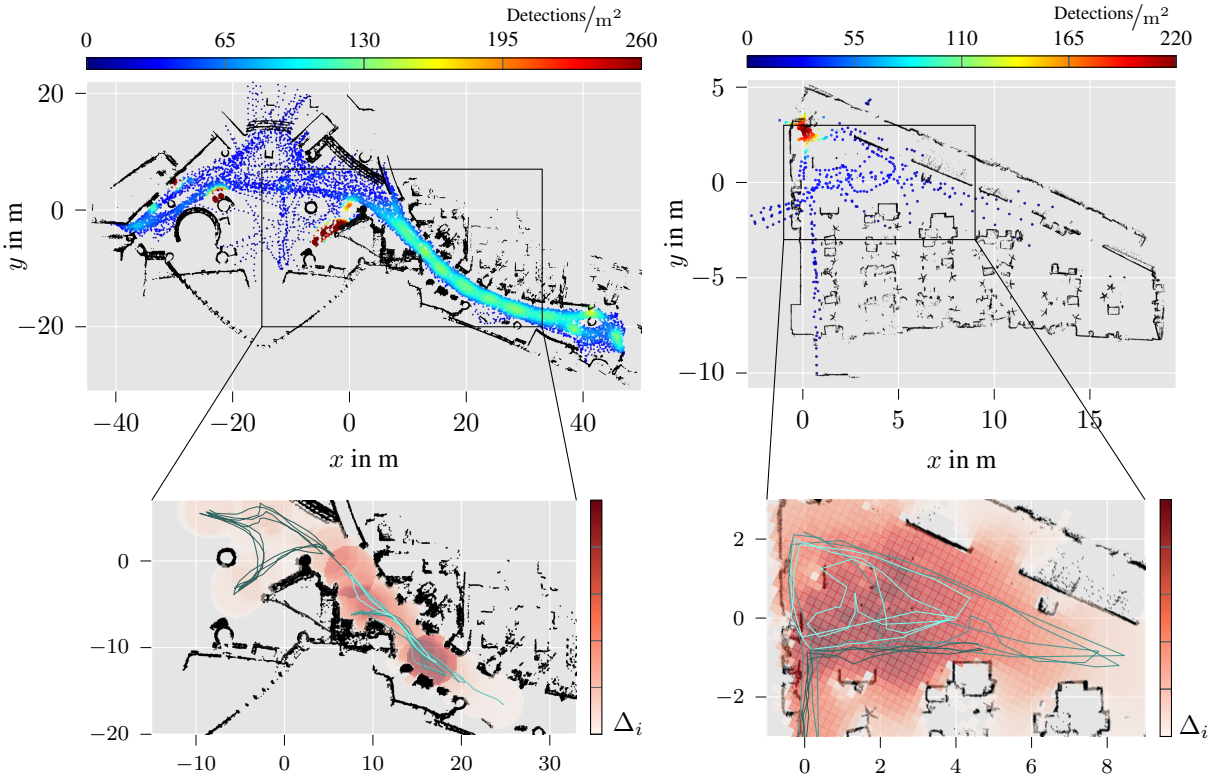


Figure 4.7: Upper images: Occupancy maps and exemplary detections in a 15 min time window during midday. The color-coded density is given by kernel density estimation with a GAUSSIAN kernel. Lower images: Simulated robot paths and exemplary observation time Δ_i for cumulated paths.

4.3.2 Evaluation Metrics and Baselines

Three criteria are used to measure the predictive quality of the model. The first criterion is normalized root mean square error

$$\mathrm{NRMSE} = \sqrt{\frac{1}{\bar{y}_{\mathrm{gt}}^2\, n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} (\hat{y}_i - y_{\mathrm{gt},i})^2}, \tag{4.20}$$

between model predictions $\hat{y}_i$ and ground truth $y_{\mathrm{gt},i}$ and normalized by the mean test data value $\bar{y}_{\mathrm{gt}}$. Normalization is necessary due to the varying length and area coverage of the different robot paths.

Ground truth is obtained similarly to the training data $\boldsymbol{y}$ as a quotient of people count and observation duration (cf. section 4.2.2), but ignoring the occupancy map and setting the observation durations $\Delta_i = \tau$. Thus, it represents the people count $c_{\mathrm{gt},i}$ during test time within the cells of the

spatio-temporal grid visited during training, using the same spatial and temporal resolution r_s and τ, respectively. The second criterion is the Chi-square distance

$$\chi^2\text{−distance} = \sum_{i=1}^{n_\mathrm{test}} \frac{(\hat{y}_i - y_{\mathrm{gt},i})^2}{(\hat{y}_i + y_{\mathrm{gt},i})}, \tag{4.21}$$

which is a measure for histogram comparison, where larger values indicate less accurate predictions. Normalized root mean square error and Chi-square distance can be regarded as the standard metrics when comparing human-activity or flow models [JWHK16; MCD21; VMS+19; VYE+20]. However, these criteria have limited explanatory value in terms of the model's usefulness, e.g., in supporting unobstructed navigation or task planning. Vintr et al. [VYE+20] propose new criteria to evaluate these models based on their ability to support human-aware navigation.

The main idea of the benchmark is to better score models that avoid disturbing humans by performing robot movements outside their immediate walking paths. The third criterion considers a number of n_nav imaginary navigation scenarios in which a robot is supposed to navigate between a set of destinations at different points in time. The navigation path is planned based on the output of each model via a two-dimensional graph search, with higher activity indicating higher path costs. All resulting paths are then ranked in ascending order by their total cost and the *service disturbance*

$$E(\lfloor n_\mathrm{nav}\rho \rfloor) = \sum_{k=1}^{\lfloor n_\mathrm{nav}\rho \rfloor} e_k \tag{4.22}$$

is defined as a sum of robot-human encounters e_k during test time. The encounters e_k are the person detections that occur within a radius of 1 m around the robot as it simulatively travels the path at a speed of $0.5\,\mathrm{m\,s^{-1}}$. The value $\rho \in [0, 1]$ is referred to as the *servicing ratio* and defines the relative number of navigation actions to be performed. A lower servicing ratio gives the robot more freedom to discard paths with a high cost, e.g., when the number of expected people is large.

Besides CoPA-Map, the following methods are compared in the evaluation.

Maximum-likelihood (ML) model: Calculates the mean of all observed rates in each cell. As a result, the rates are assumed to be constant over time.

FreMEn-Additional Amplitude Model (Fr-AAM) [JWHK16] is a state-of-the-art approach that models human activity as an inhomogeneous POISSON process through a spatial grid with a continuous-time rate function. For each cell of the grid, a spectral analysis based on the FreMEn method [KFSD17] is repeatedly performed to obtain the most influential spectral components, from which the predictive signal is then reconstructed. The method uses the training count data c_i and observation durations Δ_i separately to infer the POISSON rate with a Gamma-distributed prior. The model is trained via five-fold cross validation as it chooses the number of predictive components based on test data.

Warped-Hypertime (WHyTe) [VMS+19] is a state-of-the-art approach for continuous activity and flow modeling. It is based on a frequency analysis by the FreMEn method and subsequent projection into a circular space. As training data, it directly uses people detections $\boldsymbol{D}_{\mathrm{ped}}$ and outputs the probability of occurrence given an input point. For this reason, the model output is not directly compared to the quantitative value $c_{\mathrm{gt},i}$, but is only included in the evaluation of service disturbance. As the method uses a pre-defined number of clusters, models with up to seven clusters are trained separately, and only the variant with the best result is included. WHyTe is also capable of estimating movement direction and velocity, which is not used in the present evaluation to ensure direct comparability to the other models.

Homoscedastic GAUSSIAN *process* (GP-Hom) model: Consists of the same modules as CoPA-Map but is realized as a LOG-GAUSSIAN-COX process. Instead of a heteroscedastic likelihood, the method uses a homoscedastic POISSON likelihood for inference using a single latent function. The latent function is transformed with an exponential function to output only positive values required for the POISSON distribution rate parameter.

4.3.3 Validating Hyperparameter Initialization

The importance of proper initialization of the hyperparameters of the temporal kernel k_{t} is demonstrated by training different periodic kernels and an RBF kernel for comparison. Besides the proposed initialization method (cf. section 4.2.4), ten randomly initialized periodic kernels are used with variances chosen randomly in $(0, 1)$ and one to two random periods as multiples of 30 minutes and smaller than 30 hours. Data input is taken from an 185 m^2 area of the ATC dataset in 50 cm $\times$ 60 min resolution.

The initialization procedure results in two periods of twelve and six hours with variances of 0.9 and 0.42, respectively. Figure 4.8 shows the negative log-likelihood (NLL) loss during training and the RMSE relative to the ML model on four independent test days. Due to the high variance of the training data, NLL shows slight variation for the periodicity parameter. However, suitable initial parameters of k_{t} lead to better extrapolations and shorter optimization time, which is reflected in the evolution of the RMSE with respect to optimization steps. Completely neglecting periodicities (RBF kernel) leads to unsatisfactory prediction results since long-term changes cannot be captured, and the length scale parameter of the kernel limits the prediction horizon.

4.3.4 Spatio-Temporal Prediction

The predictive quality of CoPA-Map is evaluated in two different scenarios: a *static* and a *moving* robot. The *static* scenario assumes a permanently motionless robot for five separately considered positions and leads to a constant observation time of $\Delta_i = \tau$ for every cell. For the *moving* case, nine different paths are created with varying spatial coverage of the robot's FOV (between 40 m^2–70 m^2 for Office and 100 m^2–200 m^2 for ATC datasets) and varying waiting times along the

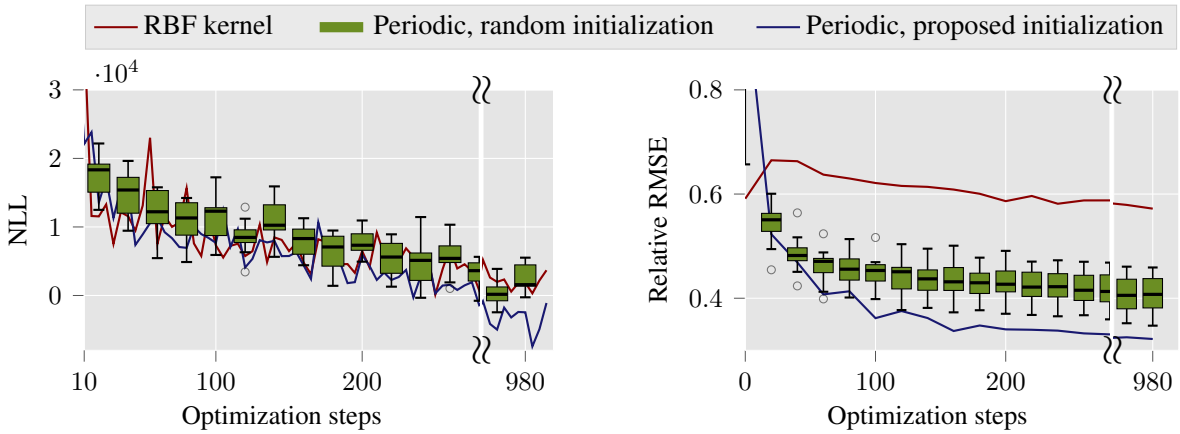


Figure 4.8: NLL and relative RMSE for an RBF kernel, a periodic kernel with the proposed initialization routine, and periodic kernels with randomly initialized parameters. ([SS22] ©2022 IEEE)

paths. The paths cover different cases, where some locations may constantly be in the robot's FOV, and others may be visited only a few times a day. Figures 4.9 and 4.10 show the results for NRMSE and χ^2-distance. Because of the dependence on the chosen spatial and temporal resolution (r_s and τ), four different combinations are shown.

The results of the *static* case (Figure 4.9) illustrate dependency of the prediction quality on the specific chosen location. While CoPA-Map performs better on the Office dataset, outliers in the ATC dataset show that some locations are better represented by discrete models or the homogeneous GP. These are mainly locations with a small area, where people stay for extended periods (e.g., queues in front of restaurants).

In the *moving* case (Figure 4.10), the advantage of CoPA-Map's heteroscedastic modeling becomes clear, as singularly occurring high target values (e.g., due to brief observation durations) are given less weight during training. Discrete models such as Fr-AAM strongly approximate areas with high numbers of people but suffer when people appear in slightly different locations in the test data.

The path with the largest area coverage results in 147,500 input points and 2,860 inducing points at the smallest resolution (ATC, 50 cm × 30 min). In this case, training took 28 minutes to converge (*Nvidia* GTX1070, i7-8700 CPU, 16 GB RAM). Thus, a training duration of this magnitude allows the training to be repeated periodically with current data, e.g., while the mobile robot is charging.

4.3.5 Service Disturbance

Two scenarios investigate whether the model can actively avoid areas with high human activity during navigation. The *hallway* scenario involves a strong flow of people along a corridor, and the *shops* scenario represents an area in front of different stores, where people movement splits and queues occur (both on the ATC dataset). The directional edges of the cost graph for navigation result from the model outputs in the MOORE neighborhood of each cell. The navigation scenarios

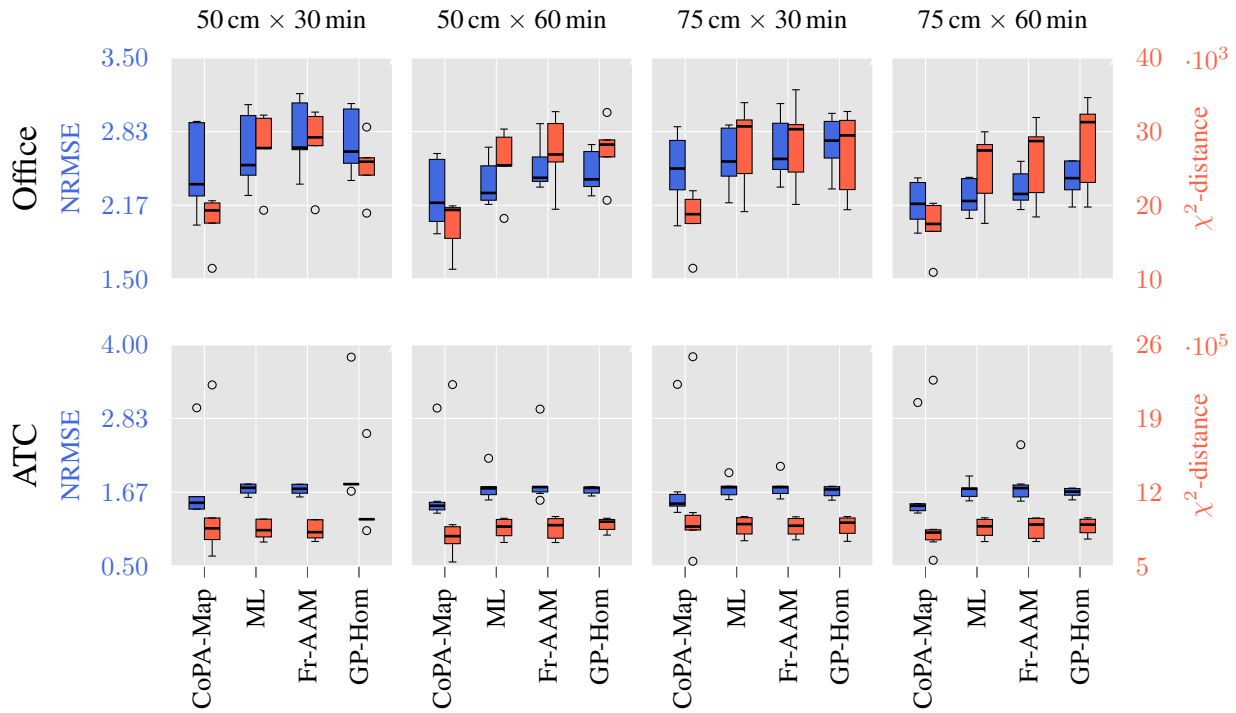


Figure 4.9: Evaluation of the *static* case.

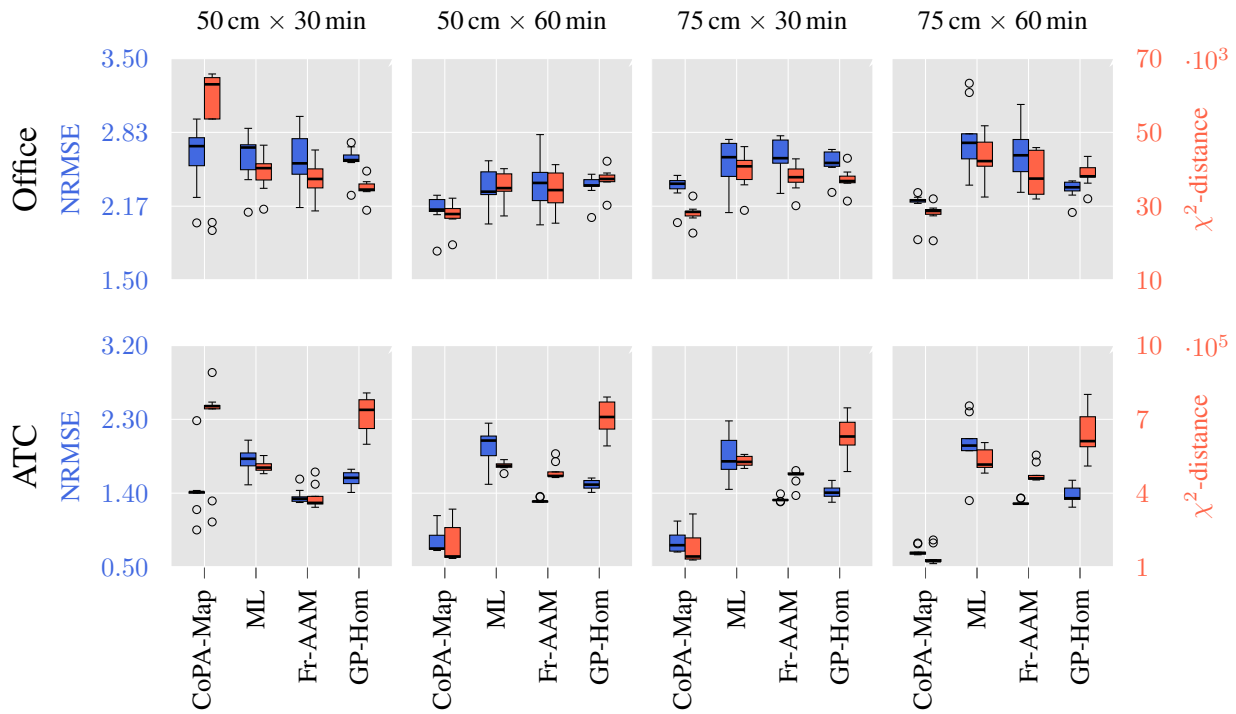
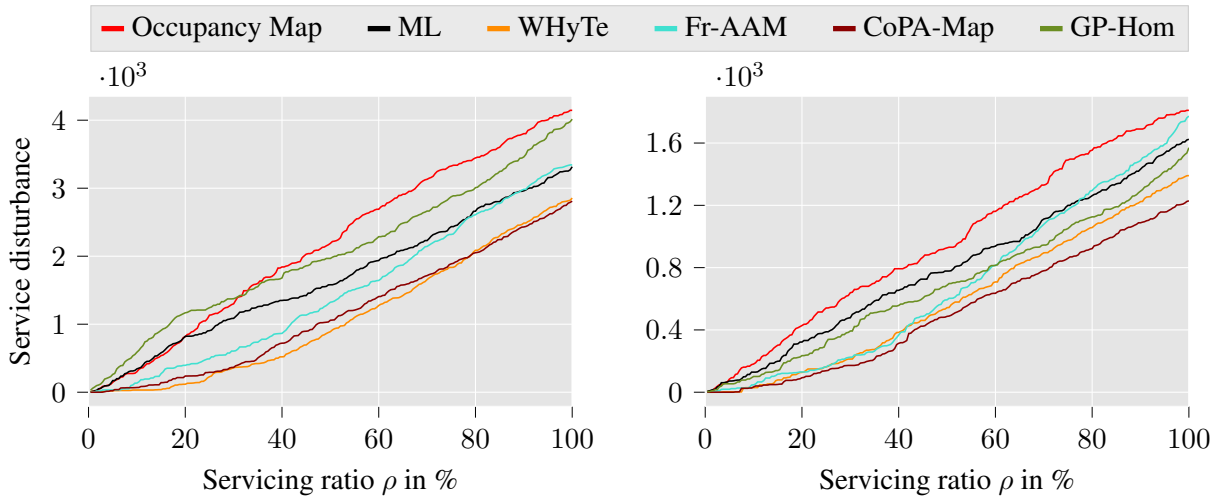


Figure 4.10: Evaluation of the *moving* case.

are inspired by a security or logistics robot procedure, where the robot must visit four predefined locations (A→B→C→D). Each scenario includes four patrols per hour between 9 a.m. and 9 p.m. for four days, leading to $n_{\text{nav}} = 240$ patrols per scenario. The paths for the scenarios are computed by Dijkstra's algorithm [Dij59] based on the cost graph resulting from each model.

As a baseline, the *Occupancy Map* model indicates how many encounters would occur if the robot always chose the shortest route. Figure 4.11 shows the cumulated number of encounters (service disturbance) over the servicing ratio ρ and Figure 4.12 gives exemplary model outputs and navigational paths for two times of day with high and low people traffic.

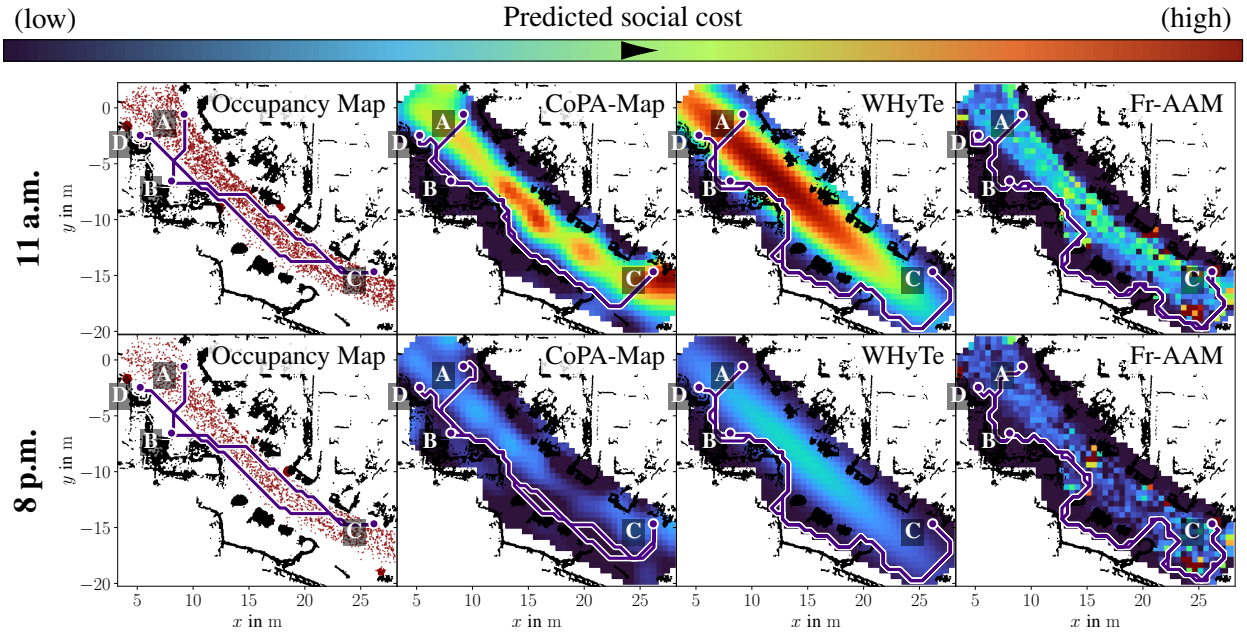


(a) *Hallway* scenario

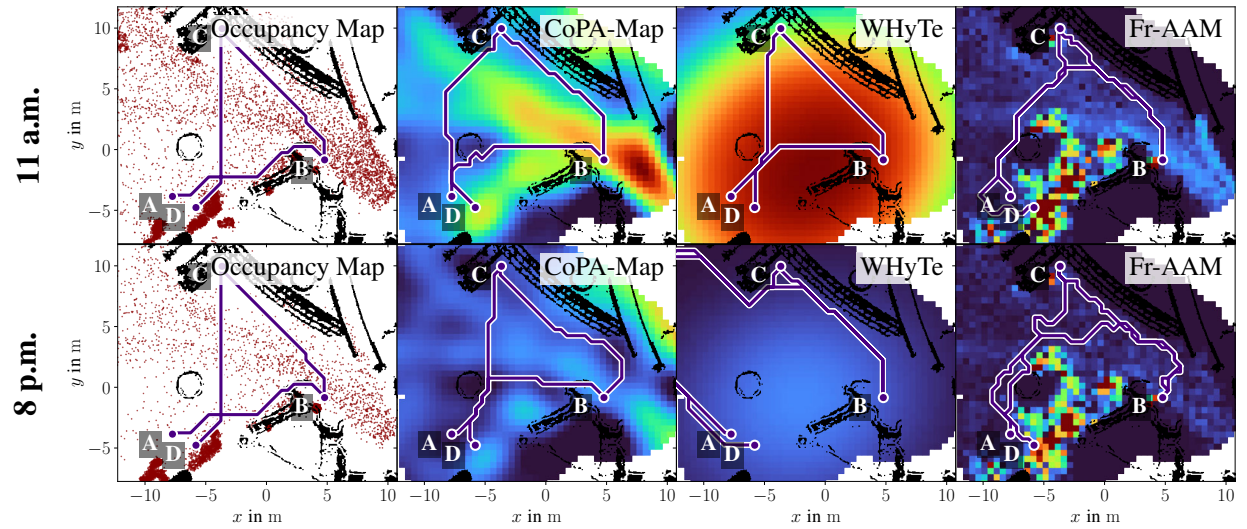
(b) *Shops* scenario

Figure 4.11: Service disturbance (cumulated encounters) for two navigation scenarios on the ATC dataset (lower is better). ([SS22] ©2022 IEEE)

Spatially continuous models (CoPA-Map, WHyTe) capture the modality of human activity in the *hallway* scenario significantly better than discrete models (e.g., Fr-AAM). Discrete models often lead to sinuous paths, which increases the number of human encounters. CoPA-Map and WHyTe perform well, particularly for small servicing ratios ($< 40\,\%$) because navigation tasks are performed only at times when few people are expected, such as in the morning and evening hours. Compared to WHyTe, CoPA-Map has advantages when pedestrian activity varies multimodally, e.g. with a constant people flow at one location and stationary stays at another, as evident in the *shops* scenario. WHyTe directly incorporates the detections $\boldsymbol{D}_{\text{ped}}$ without accounting for varying observation times, resulting in underrepresentation in the prediction in areas with short stays and thus fewer detections. For a servicing ratio of $\rho = 1$ (all paths must be driven) CoPA-Map leads to ca. 32 % fewer encounters over all paths compared to the Occupancy Map model for both the *hallway* and *shops* scenarios.



(a) *Hallway* scenario



(b) *Shops* scenario

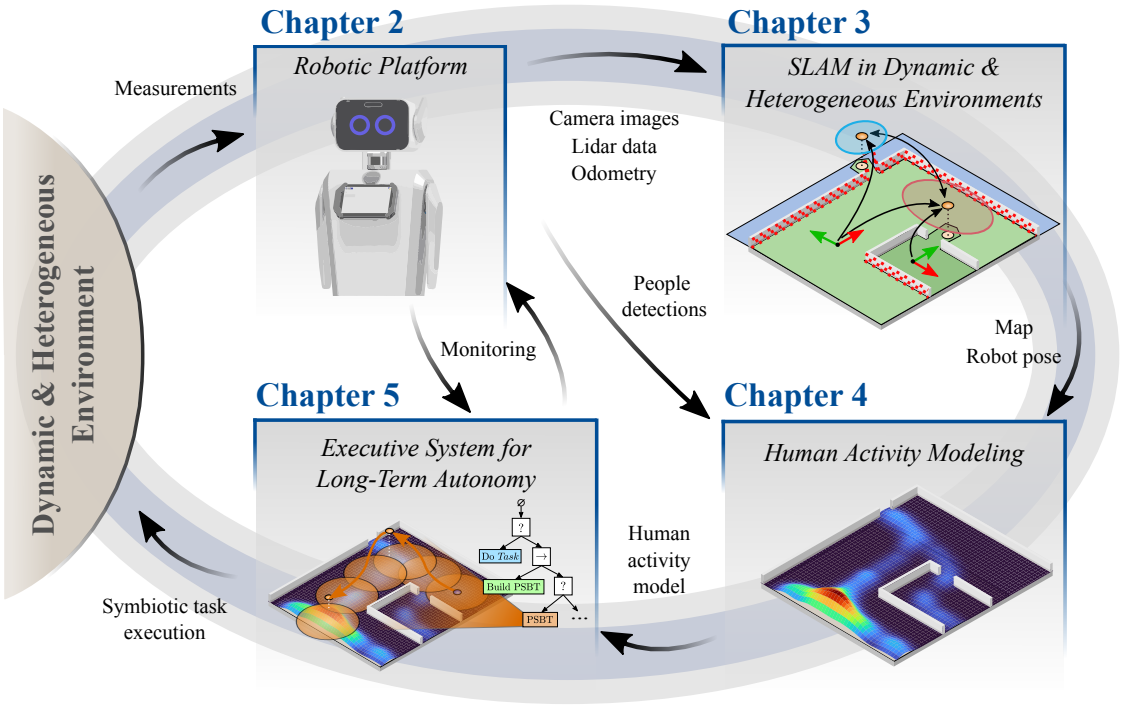
Figure 4.12: Exemplary model predictions and resulting paths (purple) from the service disturbance experiment. The social cost is scaled to the respective maximum model output. Actual pedestrian data from a 15 min time window is shown as red dots in the occupancy map subfigures. Model output outside the FOV is not displayed. Models requiring a grid representation are trained with resolution 0.5 m × 60 min.

4.4 Conclusion

This chapter introduces CoPA-Map, a non-parametric method for spatio-temporally continuous modeling of human activity. By considering heterogeneous data distributions resulting from varying dwell times as part of a heteroscedastic likelihood function, the method applies to data collected by mobile robots. Two real-world datasets show that CoPA-Map outperforms the state of the art in predictive accuracy. The predictive distribution is obtained by variational inference, which allows model training with multi-week data within minutes.

An experiment that utilizes the novel *service disturbance* metric shows that CoPA-Map allows path planning and scheduling outside areas heavily frequented by people, which actively avoids their disturbance. Compared to commonly used occupancy map models, significantly fewer encounters with humans occurred. This also emphasizes the reverse case: If people are to be proactively searched for or should be approached (e.g., for info terminal services), using such a model can increase the encounter probability.

5 Executive System for Long-Term Autonomy



Learned long-term patterns of an environment may be used to forecast human movement dynamics, as shown in the last chapter. This knowledge, inter alia, helps to productively plan paths through areas with fewer people, which avoids disturbances and increases the acceptance of service robots. However, planning paths is only one of many skills a long-term-autonomous mobile robot must frequently perform.

Autonomous robots must deliberately decide on and control their actions to achieve their tasks, which is realized by an executive system. To account for dynamics in the environment and tasks, it must not only plan actions but also reactively adapt the robot's behavior when problems arise. However, the robot's physical capabilities limit possible recovery strategies in failure cases. Many service robots do not have physical manipulators due to cost and complexity, primarily when focusing on tasks for human-robot interaction. In heterogeneous environments, this prevents them from entirely using the environment, e.g., because closed doors cannot be opened or elevators are not operated autonomously. To overcome this, integrating the knowledge of environment dynamics into the acting system can improve the robot's LTA capability in terms of reliability and robustness. Given the potential locations of people, the robot could then search and actively ask humans for help in case of failure.

Starting from a general task-switching mechanism that controls the robot's behavior, this chapter introduces a monitoring system that provides fallback behaviors and recoveries to improve the robot's performance for LTA. It is based on the modular and reactive task-switching formalism behavior trees (BTs), which allow for a stateless and extendable implementation. Firstly, an overview of the state of the art is given (section 5.1), and the fundamentals of the BT formalism are presented (section 5.2). After introducing the generally usable system-monitoring method in section 5.3, it is extended to allow for an efficient people search. Model knowledge about the spatio-temporal occurrence of people is integrated into the BT and used productively in section 5.4.

The first part of this chapter was partly published at the peer-reviewed "European Conference on Mobile Robots" (ECMR) 2021 [SWSS21]. The content beginning with section 5.4 was published at the peer-reviewed "IEEE International Conference on Robotics and Automation" (ICRA) 2021 [SLPS21].

5.1 Related Work

To achieve autonomy, service robots may have to act *deliberately*, which means performing actions that are motivated by some intended objective and aimed and achieving these objectives [IG17]. In the following section 5.1.1, a brief introduction to autonomous deliberation and its major subdomains *planning* and *acting* is given. Subsequently, strategies to mitigate failures by monitoring LTA system components are presented in section 5.1.2. Finally, a summary and critical discussion are made in section 5.1.3.

5.1.1 Autonomous Deliberation

Service robots face dynamic environments, semantically rich tasks, and human interactions, requiring them to act autonomously and deliberately. These demands are in stark contrast to robots in fixed and controlled, i.e., industrial, settings.

Although there is no irrefutable definition of *deliberation*, its two significant aspects are commonly referred to in the literature as *planning* and *acting* [GNT04; IG17]. It should be noted, however, that boundaries between these functions are fluid and depend on specific tasks, implementations, and utilized methods.

Planning and Acting

Based on an input and predictive model of the environment, task planning creates an output plan expressed in some representation. Acting then refines the planned actions into specific commands and reacts to external events. Whereas planning can be viewed as an open-loop search, acting must be a closed-loop process. Executive systems that include both functionalities are part of almost

every LTA system. For example, planning approaches are used to schedule tasks arrangements for the Opportunity rover [BJMR05], delivery or patrolling applications [MLH15] or in logistics systems to cope with varying orders and resources [BK21].

A planning system generally needs to consider spatial and temporal variations to allow for concurrent activities and synchronization. Since environments are dynamic, probabilistic planning techniques may address the uncertainty in knowledge and environment variability. MARKOV decision processes (MDPs) are a classical framework in this regard, which can be extended to partially observable systems (POMDP). Successful implementations of this technique include applications for navigation planning [FT07], multi-tasking SLAM [GP10] or people search [TA11]. However, MDP-based models suffer from the assumption that actions are applicable in every state, restrictive model formulation, and specific tailoring of optimization problems [GNT16].

Whereas task planning creates a trajectory in an abstract action space and may be domain-independent, acting needs to refine the planned actions into specific commands and react to (unforeseen) events. Therefore, the acting system needs to be both reactive and modular. If a system can be divided into separate building blocks and rearranged (i.e. when it is modular), this reinforces independent development, testing, and reusability. One standard approach for acting systems has long been the finite state machine (FSM), which models the task execution by a set of states, transitions, and events [Mea55]. FSMs were used to successfully control LTA deployments of service robots over multiple days [MMWG11] which lead the way to the widely used framework *SMACH* [BC10], tour guide robots [WC18] or physical therapy of the elderly [HKG+16]. The main drawbacks of FSMs are their complex maintainability, non-trivial scalability, and reusability, which are particularly important for LTA applications. Although extensions to hierarchical finite state machines (HFSMs) [Har87] exist to increase modularity, these formalisms are still hard to maintain due to the need for redefinition of transitions when states are added or removed. behavior trees (BTs) were introduced as a tool to improve on these aspects [CÖ18].

Originating in the computer-game industry, BTs found attention in the robotics community in recent years [CN21; ISS+20]. Due to their tree structure and standardized task interfaces, modularity and reactiveness are ensured. Additionally, they inherently support task hierarchies and can be automatically synthesized by a planner [CAÖ19; MME+21; LPS+21; SIN+22]. These characteristics meet the current challenges stated by the planning community that deliberation should be organized hierarchically and be expandable online [GNT16]. Successful examples of BT implementation include the navigation and planning frameworks for ROS 2 [MMWG20; MGMR21], the *Boston Dynamics'* Spot SDK [Bos22] to model the robot's mission or non-expert programming via the *CoSTAR* framework [PHJ+17].

5.1.2 Failure Mitigation

An executive control system that plans and acts on action sequences and responds to opportunities and failures is an essential part of an LTA architecture [KHD+18]. In particular, variable and

dynamic environments and temporal changes may lead to unforeseen situations. Therefore, an important goal is the prevention of behavioral loops and providing recovery mechanisms, which is realized by a *monitoring* system. Monitoring is closely intertwined with acting [IG17], as recoveries may include specifically executed behaviors and even the involvement of humans in problem-solving.

Monitoring Systems

The key responsibilities of execution monitoring systems are detecting discrepancies between predictions and observations, diagnosing their cause, and recovering from them [GNT16]. Monitoring supervises and, if necessary, modifies the executed commands of acting subsystems. Modification may happen on the sensory-motor level via control-theory by comparing the system and component behavior with a nominal target [Ant14], by statistical methods, or outlier rejection techniques [KK18a]. Concerning LTA this has been explored particularly for areas outside of service robotics, such as underwater [XYZ17] or extraterrestrial applications [LC04].

Besides fault detection and diagnosis on the control level, the complex interplay of different software components in LTA robots can lead to errors preventing their continuous deployment. This has been especially highlighted in large-scale research initiatives with service robot deployments over several weeks, namely, the *STRANDS* [HBJ+17], *CoBot* [BV16], and Willow Garage *Office Marathon* [MMWG11] projects. One key strategy of the STRANDS executive system for delivering long-term software robustness is continuously monitoring task and navigation execution, restarting components on demand, and triggering recovery behaviors. It follows the widely employed [HBJ+17; MMWG11; WC18] design principle that software components of mobile robots must not be flawless but instead should tolerate and even enforce restarts in case of failures [BJPM16]. Frequent restarts and running as few processes as possible also lowers the probability of memory leaks, internal deadlocks, and invalid state [MMWG11].

A recurring motif for fault recovery in this context is the involvement of humans as a last resort, e.g., by sending emails, requesting teleoperation, or asking bystanders for help. As argued by [MMWG11; BV16], maximum robustness may not be achieved by full autonomy or complete human control alone but by combining both.

Symbiotic Autonomy

Overcoming the limitations of a robotic system by actively involving humans is referred to as *symbiotic autonomy* and has been considered in various contexts. Malfunctions can be mitigated by forewarning the users [LKF+10] or purposefully utilizing human collaboration in autonomous plans [NI14]. This may also be included already in the planning stage to plan for task collaboration [ACM+06] or contingency [Fra18]. An essential distinction of human-centered help is how knowledgeable potential helpers are. System experts can usually only be requested for teleoperation

and are not available in the immediate environment. Humans in the robot's work environment are present in higher numbers but can only be involved in tasks that can be done easily and quickly [BV16].

When a situation occurs where the robot needs help, this must first be identified, e.g., by detecting a closed door [SNTO19] or an elevator [LSG19]. Asking humans for help requires locating them first, which may be based on greedy search [VG13], hidden MARKOV models [BN14], or periodic GAUSSIAN mixture models [KKM+15]. In [TA11] people are searched based on a model of the environment, by employing an individually defined MDP with lattice-like movement primitives.

The search during symbiotic autonomous tasks requires that people are not only found but then accompany the robot to the location where assistance is needed. Since people are only willing to travel a limited distance to help [RV12], this imposes an additional constraint on the search locations. Most work in this context assumes that help is always available at the immediate help location, e.g., by supervisors [TKL+14] or bystanders [WIT+10]. Only few authors consider proactively searching and finding people to fulfill a task that the robot cannot achieve alone: Rosenthal et al. [RVD12] show that navigating the environment to search for humans could decrease the time until a potential helper is found. They employ an A*-based planner to decide where to seek help in an office building based on the location of offices and availability of the person. However, their method is only evaluated with static locations, using occupancy sensors installed in offices, and does not apply to dynamically created locations based on people detections.

5.1.3 Conclusion

For service robots to act autonomously (in the long term), they need the ability to *deliberate*. Deliberation includes task planning, reactively acting on the synthesized plan, and monitoring its proper execution. Uncertainties resulting from dynamic environments can be addressed by probabilistic techniques, such as MDPs or POMDPs. However, their model formulation is restrictive and problem-dependent, complicating the implementation for robots that must execute various tasks. The actual task execution is the responsibility of the acting system, which is often implemented using FSM. The drawbacks of this technique concerning reactiveness, modularity, and expandability are increasingly addressed by employing BTs. Due to their statelessness, BTs are also suitable for monitoring systems to react deterministically to errors. However, this has not yet been implemented in the state of research.

Besides autonomous recovery behaviors, monitoring systems of LTA robots may involve humans, i.e., supervisors or bystanders. Since help is not always available at the robot's immediate location, deciding whether and where to search for people may be necessary. In the state of research, it is usually assumed that help is directly available or that potential locations of helpers are static. In other cases, only simulations are used for evaluation [TA11; RV12]. Hence, the application of *dynamically created models of person occurrence* to symbiotic autonomy and a *generally applicable* action description to find people are open problems. BTs are particularly suitable for

this purpose since they can be synthesized stochastically and have a human-readable structure. Compared to the otherwise used MDP or A*-based planners, this results in structures that can be extended easily by additional task elements.

5.2 Preliminaries

This section introduces the methodology of behavior trees (BTs) and the subsequent extension to stochastic behavior trees (SBTs). The following descriptions only cover the aspects that are fundamentally necessary for understanding the developed methods and do not claim completeness. For a comprehensive introduction to BTs, the reader is referred to [ISS+20] for applications, [CN21] for implementations in robotics, and [CÖ18] for the general methodology.

5.2.1 Behavior Trees

A behavior tree (BT) is a directed rooted tree consisting of internal nodes for control flow and leaf nodes for action execution or condition evaluation [CÖ18]. Pairs of adjacent nodes are denoted as *parent* (outgoing) and *child* (incoming). The node without parents is called the root node, which periodically sends an enabling signal (*tick*) through the tree, which is then propagated according to the policies of different control flow nodes. A node is executed if, and only if, it receives ticks. It then immediately returns one of three states: *running* (*R*, execution ongoing), *success* (*S*, goal achieved), or *failure* (*F*). In the classical formulation of BTs, there are four types of control flow nodes (sequence, fallback, parallel, decorator) and two types of leaf nodes (action, condition), which are shortly introduced in the following.

The *sequence node* executes Algorithm 5.1 and routes the tick from left to right until a child returns either *F* or *R* as status, which is then returned to the parent of the node. Only if all children return *S* the sequence node returns *S*. It is denoted by a "$\rightarrow$" symbol, as shown in Figure 5.1.

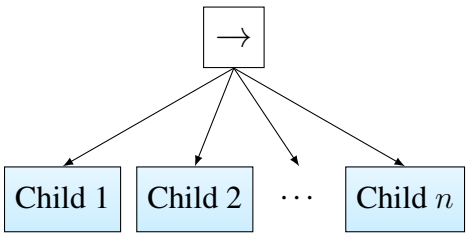


Figure 5.1: Graphical representation of a sequence node with n children.

Algorithm 5.1: Pseudocode of a sequence node with n children.

```
for i ← 1 to n do
    childStatus ← Tick(child(i))
    if childStatus = R then
        return R
    else if childStatus = F then
        return F
return S
```

The *fallback node* executes Algorithm 5.2 and routes the tick from left to right until a child returns either *S* or *R* as status, which is then again returned to the parent of the fallback node. Only if all children return *F* the fallback node returns *F*. It is denoted by a "?" symbol, as shown in Figure 5.2.

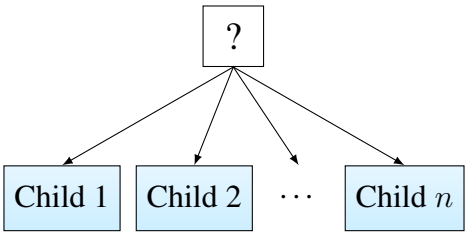


Figure 5.2: Graphical representation of a fallback node with n children.

Algorithm 5.2: Pseudocode of a fallback node with n children.

```
for i ← 1 to n do
    childStatus ← Tick(child(i))
    if childStatus = R then
        return R
    else if childStatus = S then
        return S
return F
```

An *action node* executes a command when it is ticked, and it returns *R* during execution. At completion, it returns either *S* if it succeeded or *F* if it failed. A *condition node* checks a specified expression and returns *S* if it is satisfied and *F* if not. The condition node can not return *R*. A *decorator node* is a node with a single child that modifies the output of its child node according to a predefined policy. Decorator nodes can, e.g., be used to invert the output of a node ("¬") or stop the tick propagation after a predefined time limit. The graphical representation of the decorator node is depicted in Figure 5.3 together with the condition and action node. The labels describe the verified condition or the performed action, respectively. Assemblies of different control flow and leaf nodes may also be grouped as a single BT. This tree can be viewed as a subtree, whose graphical representation is also shown in Figure 5.3. Various extensions of this basic set of node types exist, e.g., nodes with memory or for asynchronous or parallel execution [CÖ18; CN21], which are not covered in this work. An exemplary BT describing a recharging behavior of an autonomous robot is shown in Figure 5.4. The charging behavior executes a docking, charging, and undocking procedure if specific conditions based on the robot's battery level are met. By adding this behavior to a fallback node, arbitrary tasks can be extended by this charging functionality, which will automatically interrupt the task execution if the robot needs to be charged. Thus, this example emphasizes the reactiveness and modularity of the BT-framework.

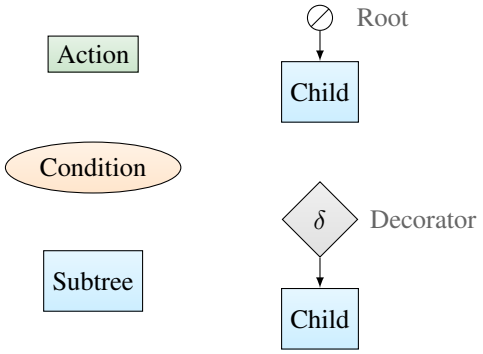


Figure 5.3: Graphical representation of an action, condition, subtree, decorator, and root node.

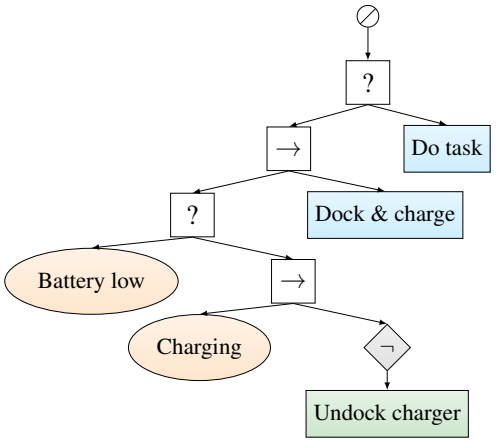


Figure 5.4: Exemplary BT describing a recharging behavior.

5.2.2 Stochastic Behavior Trees

One drawback of the original formulation of BTs is the fixed execution order of fallback and sequence nodes, which must be defined manually and is often non-trivial to choose. To overcome this problem, SBTs are introduced in [CMÖ14]. In an SBT, each action is extended by a tuple $A_{\mathrm{sbt}} : \langle p_{\mathrm{s}}(t), p_{\mathrm{f}}(t), \mu, \nu \rangle$ and each condition by a tuple $C_{\mathrm{sbt}} : \langle p_{\mathrm{s}}(t), p_{\mathrm{f}}(t) \rangle$, where $p_{\mathrm{s}}(t)$ and $p_{\mathrm{f}}(t)$ are the probabilities to succeed and fail respectively, at any given time t. The time to succeed and time to fail are random variables with exponential distribution and rates μ and ν, respectively. The variables and probability distributions of the tuples can be defined individually for each action and condition, e.g., based on expert or model knowledge. Interconnection of the stochastic nodes then allows to specify success and failure probabilities for entire BTs and to prioritize them. To receive an indication of the success and failure probabilities $p_{\mathrm{s,T}}(t)$ and $p_{\mathrm{f,T}}(t)$ of a complete BT, the inner flow of an SBT is described as a discrete-time MARKOV chain (DTMC) with states $\mathbb{S}$. A DTMC is suitable to model state changes with given transition probabilities in a system without regarding state history. The total probabilities follow by summing up the probabilities of being in one of the DTMC success/failure states $\mathbb{S}_{\mathrm{S}}$ and $\mathbb{S}_{\mathrm{F}}$ as

$$p_{\mathrm{s,T}}(t) = \sum_{i:s_i \in \mathbb{S}_{\mathrm{S}}} \pi_i(t), \quad p_{\mathrm{f,T}}(t) = \sum_{i:s_i \in \mathbb{S}_{\mathrm{F}}} \pi_i(t). \tag{5.1}$$

The probability vector $\boldsymbol{\pi}(t)$ is obtained by solving the CAUCHY problem

$$\dot{\boldsymbol{\pi}}(t) = \boldsymbol{Q}(t)\boldsymbol{\pi}(t), \quad \boldsymbol{\pi}(0) = \boldsymbol{\pi}_0, \tag{5.2}$$

with $\boldsymbol{Q}$ as the infinitesimal generator matrix of the DTMC and initial probability vector $\boldsymbol{\pi}_0$ assumed to be known a priori. With the definition of transient states as $\mathbb{S}_{\mathrm{T}} = (\mathbb{S} \setminus \mathbb{S}_{\mathrm{S}}) \setminus \mathbb{S}_{\mathrm{F}}$, the success rate μ_{T} of the tree is calculated as

$$\mu_{\mathrm{T}} = \mathrm{avg}\left(\frac{\sum_{i=1}^{|\mathbb{S}_{\mathrm{S}}|} u_{i1}^{\mathrm{S}}(\kappa) \log\left(h_{i1}^{\mathrm{S}}(\kappa)\right)}{\sum_{i=1}^{|\mathbb{S}_{\mathrm{S}}|} u_{i1}^{\mathrm{S}}(\kappa)} \right)^{-1} \tag{5.3}$$

with $\mathrm{avg}(\cdot)$ as the average function over time and κ as a time step. The matrices $\boldsymbol{H}^{\mathrm{S}}(\kappa, \mathbb{A}_{\mathrm{sbt}}, \mathbb{C}_{\mathrm{sbt}}) \in \mathbb{R}^{|\mathbb{S}_{\mathrm{S}}| \times |\mathbb{S}_{\mathrm{T}}|}$ and $\boldsymbol{U}^{\mathrm{S}}(\kappa, \mathbb{A}_{\mathrm{sbt}}, \mathbb{C}_{\mathrm{sbt}}) \in \mathbb{R}^{|\mathbb{S}_{\mathrm{S}}| \times |\mathbb{S}_{\mathrm{T}}|}$ depend on the transit times, the number of steps between transient states and success states, and the success and failure probabilities and rates of all actions $\mathbb{A}_{\mathrm{sbt}}$ and conditions $\mathbb{C}_{\mathrm{sbt}}$. The sets $\mathbb{A}_{\mathrm{sbt}}$ and $\mathbb{C}_{\mathrm{sbt}}$ contain all individual actions A_{sbt} and conditions C_{sbt} of the tree. The failure rate of the tree ν_{T} follows likewise based on the failure states $\mathbb{S}_{\mathrm{F}}$ as

$$\nu_{\mathrm{T}} = \mathrm{avg}\left(\frac{\sum_{i=1}^{|\mathbb{S}_{\mathrm{F}}|} u_{i1}^{\mathrm{F}}(\kappa) \log\left(h_{i1}^{\mathrm{F}}(\kappa)\right)}{\sum_{i=1}^{|\mathbb{S}_{\mathrm{F}}|} u_{i1}^{\mathrm{F}}(\kappa)} \right)^{-1} \tag{5.4}$$

with matrices $\boldsymbol{H}^{\mathrm{F}}(\kappa, \mathbb{A}_{\mathrm{sbt}}, \mathbb{C}_{\mathrm{sbt}}) \in \mathbb{R}^{|\mathbb{S}_{\mathrm{F}}| \times |\mathbb{S}_{\mathrm{T}}|}$ and $\boldsymbol{U}^{\mathrm{F}}(\kappa, \mathbb{A}_{\mathrm{sbt}}, \mathbb{C}_{\mathrm{sbt}}) \in \mathbb{R}^{|\mathbb{S}_{\mathrm{F}}| \times |\mathbb{S}_{\mathrm{T}}|}$ [CMÖ14].

5.3 Reactive Task Control and Monitoring System

Assuming a task-switching mechanism based on BTs, this section and the next section introduce a new method to provide monitoring and recovery functionalities for technical and task errors. The task execution BT may realize arbitrary behaviors, synthesized by a planner or specified manually (see also Figure 2.4 for exemplary behaviors of the Sobi robot). The proposed monitoring system acts as a reactive extension, which is not included in the planning stage of the respective tasks but is activated by binary conditions in the acting stage. It will then block the task execution until the problematic situation is resolved. The system consists of two main parts, depicted in Figure 5.5. The first part is system monitoring, which describes the analysis of technical functions that may be fixed automatically or by remote supervision in a failure case. The second part considers faults that require manual help at the immediate location by bystanders and will be presented in section 5.4

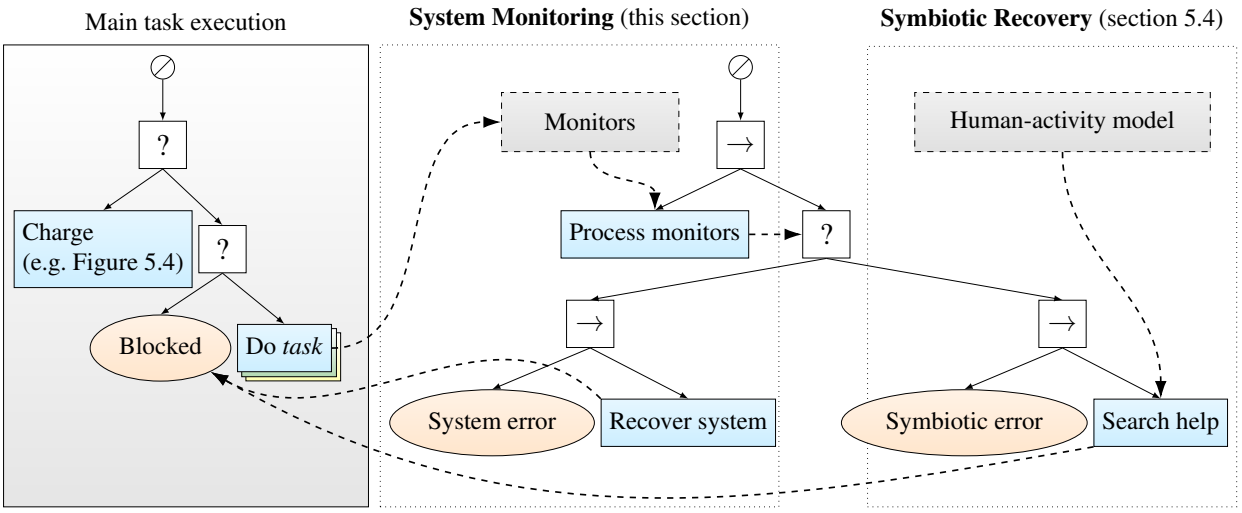


Figure 5.5: Extension of a BT-controlled task by system monitoring and symbiotic recoveries. Dashed connections indicate information flow. The task execution BT may realize, for example, a patrolling or guiding task with charging when needed.

5.3.1 System Monitoring

The system monitoring consists of separate applications that monitor hardware- and ROS framework parameters and a BT-based arbiter. Although BTs are mainly used for sequential control of autonomous agents, their advantages in reactivity and modularity combined with intuitive modeling are also applicable for monitoring purposes. Therefore, due to their statelessness and reactive structure, it is possible to react immediately to errors that occur, without explicit state transition modeling, as in the case of finite state machines. Before executing recovery behaviors, the system temporarily blocks the main task execution by setting a condition that is reverted after the error is corrected. Hardware-wise, the system monitoring continuously checks the CPU, random-access memory (RAM) and network load and additionally the time differences based on NTP if different

host computers are used as part of the robot. ROS nodes are continuously checked for if pings are received and if actual topic publishing rates are within a tolerance band. Additionally, specific monitors check whether there is a valid loop closure in the localization system and if there are error cases in the navigation system. Each monitor includes an individual warning and error range, the activation of which serves as an input to the BT-based arbiter that responds deterministically to the different error cases. The structure of the monitoring system is shown in Fig. 5.6.

System entities (i.e. programs or ROS topics) to monitor are organized in configurations for different use cases, such as regular operation, charging, or mapping. Configurations may be switched autonomously or manually, with all currently unneeded programs from other configurations terminating after the change. Monitor-specific detection signals determine whether an error is present and all error messages are aggregated and fed into the BT. Table 5.1 summarizes these signals together with the associated recovery reaction.

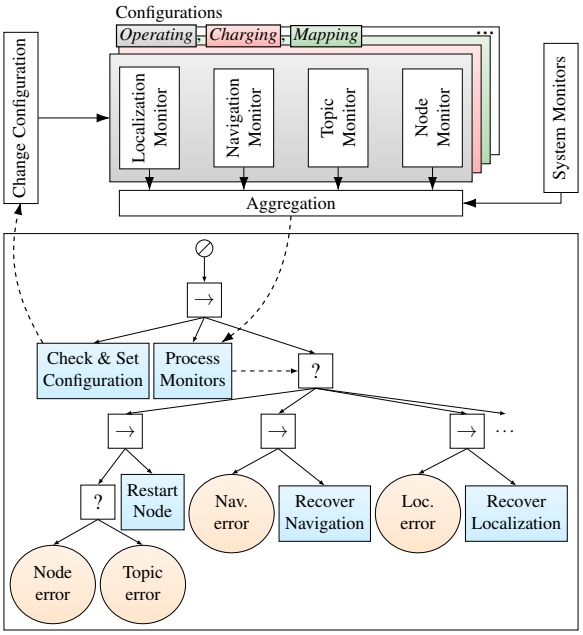


Figure 5.6: Structure of the monitoring framework.

Name	Detection	Reaction
Node monitor	Node not pingable	Restart node
Topic monitor	Publish frequency not in tolerance band	Restart publishing node
CPU, RAM, NTP, Network Monitor	Not in tolerance band	Send message
Navigation monitor	No global path found and *MoveBase* recoveries failed	1. Wait and retry 2. Move backwards 3. Ask Supervisor
Localization monitor	No loop closure detected	1. Slow rotate 2. Restart localization 3. Slow rotate 4. Ask Supervisor

Table 5.1: Overview of the utilized monitors with detection signal and reaction.

Similar to [MMWG11], the approach is that maximum robustness may not be achieved by full autonomy alone but by planned interventions by supervisors in case of failures. Therefore, the last resort for navigation and localization errors is a predefined request to a list of supervisors, in which the robot sends an instant message with a URL. The linked website then allows for teleoperation based on the camera views, or the current position on the map can be specified. This simple system intervention usually takes less than a minute of the operator's time but prevents a total failure of the system. Once the problem is fixed, the operator confirms this, and an all-clear is sent to the other supervisors.

5.3.2 Results

The system monitoring is validated in a 16-day deployment of the mobile robot Sobi, introduced in chapter 2. In addition to the effectiveness of monitoring, the evaluation demonstrates the LTA capabilities of Sobi and its components, such as localization (cf. section 3.2.1), navigation or docking (cf. section 2.1.3).

The robot's task is to permanently patrol one office floor during office hours (9 a.m. – 5 p.m.) on weekdays. Part of the floor is a corridor connected to several offices, workshops, laboratories, and an entrance area. The metric and topological maps of the environment are shown in Figure 5.7. Outside office hours, and as soon as the battery level falls below a specific threshold, Sobi autonomously approaches the charging station and performs the charging process.

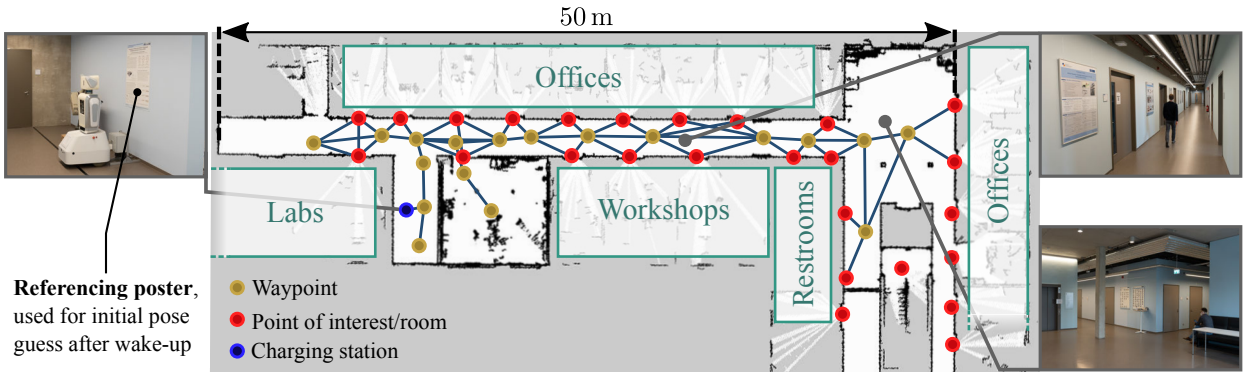


Figure 5.7: Environment used for validation. Dots indicate nodes, and blue lines indicate edges of the topological map. The robot continuously patrols between the shown waypoints randomly.

Covid-19-related hygiene regulations are in place during the evaluation in the year 2021, requiring employees and visitors to wear face masks inside the building. Therefore, visual mask recognition is implemented as part of the patrol task. In addition to automated verbal requests to comply with hygiene regulations, the number of detected faces is a measure of the environmental dynamics since crowded environments pose a more significant challenge for LTA systems. The detection is realized in a frequency of 0.5 Hz as a binary classification via the YOLO framework [RF18] with 600 images containing both *mask* and *no mask* classes (Average precision on 100 validation images: 92.9 % for the *mask* and 82.34 % for the *no mask* class).

Table 5.2 summarizes the number of detections together with typical LTA metrics and Table 5.3 gives an overview of the performed recovery behaviors. As in [HBJ+17], a recovery behavior is considered successful if no further recovery of the same category needs to be performed within one minute. An additional requirement for navigation recoveries is that no recovery must be performed within a circle of 1 m radius within this time.

In total, the robot was undocked from the charging station for 71.2 h, of which 65.4 h were spent in motion. The latter indicates the duration the robot performed its patrol task and was not in an error or recovery state. Relative to an 8-hour workday, this yields an autonomy percentage $A\%_{(8\,\mathrm{h})}$ of

Metric	Value
Timespan	12 days
Mean time operating per day t_o	5.9 h
Mean time moving per day t_m	5.5 h
$A\%_{(8\,\mathrm{h})} = {}^{t_\mathrm{m}}/_{8\,\mathrm{h}}$	69 %
$A\%_{(t_\mathrm{o})} = {}^{t_\mathrm{m}}/_{t_\mathrm{o}}$	93 %
Traveled distance	66.6 km
Mean TSL	56.7 h
Max TSL	90.6 h
Detected faces with mask	983
Detected faces w/o mask	101
Successful docking attempts	27/28

Table 5.2: Metrics of the deployment. Weekend days are excluded.

Type	Reaction	#	success
ROS	Restart node	32	62.5 %
Navigation	Wait and retry	194	78.8 %
	Move backwards	41	87.8 %
	Ask supervisor for teleoperation	5	100 %
Localization	Slow rotate	76	84.2 %
	Restart localization & rotate again	12	66.6 %
	Ask supervisor for relocalization	4	100 %

Table 5.3: Executed recovery behaviors sorted by category.

69 %, which is an indicator of the percentage of available time that the robot uses to provide its services (patrolling and mask checking). In this case, the parameter is mainly influenced by the average additional charging time of 2 h during the day. With respect to the time the robot was active (not charging), the autonomy percentage $A\%_{(t_\mathrm{o})}$ is 93 %, which emphasizes the high availability of the system.

As evident in Table 5.3, most occurring problems can already be solved by waiting and going back (in case of navigation errors) or slow rotation and restarts (in case of localization errors). A typical quantity for evaluating overall system robustness is the total system lifetime (TSL), which specifies the time interval between interventions by supervisors that were not explicitly requested by the robot in the event of a failure. Four of such interventions were necessary during the evaluation period, resulting in the TSL values shown in Table 5.2.

5.4 Incorporating Symbiotic Autonomy

The results of the previous section indicate that system errors may often be fixed automatically or by teleoperation, if necessary. However, these solution strategies are limited to problems that do not require physical intervention. Many problems, such as blocked paths or closed doors, require the direct help of people on site and the integration of symbiotically autonomous processes. Therefore, this section introduces an approach to extend a BT by a search for help based on a human-activity model through a combination of waiting and searching behaviors. It is assumed that finding people faster will lead to faster problem-solving. Further steps of the symbiotic process, such as verbally asking for help and solving the original problem, are strongly case-dependent and are not considered part of this work.

The goal of the BT extension is to set up a sequence of actions to maximize the probability of meeting a person. To create the tree, wait and search actions are defined (cf. section 5.4.2) and intelligently assembled to form a BT (cf. section 5.4.3) that performs waiting and searching at different locations. Underlying the synthesis of these actions is a model that gives a spatio-temporal indication of human activity and is required to follow a specific form, which is introduced in the following section.

5.4.1 Integrating Models of Human Activity

The human-activity model is supposed to provide a temporally dependent estimation of the time until a person arrives at a specific location $\boldsymbol{x}_\mathrm{s} \in \mathbb{R}^2$. As commonly done [Ibe13], this *interarrival time* is assumed to be of exponential distribution, which is entirely defined by a rate function $\lambda\left(\boldsymbol{x}_\mathrm{s}, t\right)$. For stationary and independent increments with exponentially distributed interarrival times the corresponding stochastic process modeling the count of people is the POISSON process. With Poisson-distributed random variable $N(X)$ (with spatio-temporal region $X \subseteq D \times \mathbb{R}$, where $D \subset \mathbb{R}^2$ is a spatial region), the probability of $N(X)$ being equal to a count c is given by

$$p\left(N(X) = c\right) = \frac{(\lambda(t)t)^c}{c!} e^{-\lambda(t)t} \quad \text{with} \quad c = 0, 1, 2, ..., \tag{5.5}$$

where

$$\lambda(t) = \int_D \lambda\left(\boldsymbol{x}_\mathrm{s}, t\right) \mathrm{d}\boldsymbol{x}_\mathrm{s} \tag{5.6}$$

and counts are regarded as the sum of single, independent people occurrences. As learning the full continuous rate function $\lambda\left(\boldsymbol{x}_\mathrm{s}, t\right)$ is an expensive process, it is often approximated by a 2D grid representation [LDA11; TA11]

$$G: \quad \lambda\left(\boldsymbol{x}_\mathrm{s}, t\right) \approx \sum_{i=1}^{m_\mathrm{grid}} \sum_{j=1}^{o_\mathrm{grid}} \lambda_{ij\tau} \mathbf{1}_{ij\tau}(\boldsymbol{x}_\mathrm{s}) \tag{5.7}$$

with $G : \mathbb{R}^{m_\mathrm{grid} \times o_\mathrm{grid}} \rightarrow \mathbb{R}$, indicator function $\mathbf{1}_{ij\tau}(\boldsymbol{x})$ and $\lambda_{ij\tau}$ as the constant rate of a piece-wise homogeneous Poisson process, valid in a time interval $[t_\tau, t_{\tau+1})$. Learning the constant rates $\lambda_{ij\tau}$ may then be achieved by Bayesian inference with Gamma-distributed prior $\lambda_\tau \sim \Gamma\left(\lambda_\tau; \alpha_\tau, \beta_\tau\right)$ with α_τ directly resulting from people detections and β_τ by incrementation [LDA11].

As introduced in chapter 4, a spatio-temporally continuous rate function can also be obtained by GAUSSIAN process regression (GPR). The model may either be realized as a LOG-GAUSSIAN-COX Process (cf. section 4.3.2) or with a normally distributed likelihood, as done with the proposed method CoPA-Map. The normal distribution as the limiting distribution of the POISSON distribution

[GK20] approximates it for $\lambda \to \infty$. Following the notation from section 4.2.5, the expected value and variance of the rate result as

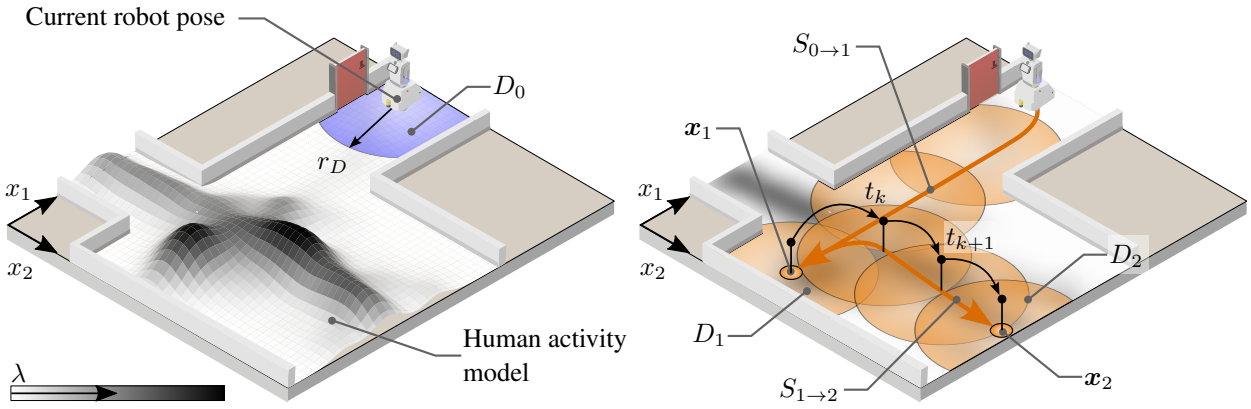
$$\mathrm{E}\left[N(X)\right] = \mathrm{Var}\left[N(X)\right] = \lambda(t) \approx \int_D m_y\left(\boldsymbol{x}_\mathrm{s}, t\right) \mathrm{d}\boldsymbol{x}_\mathrm{s}. \tag{5.8}$$

In practice, this approximation may already be used for values $\lambda \gtrapprox 10$ if continuity correction is performed [GK20]. Here, values of this magnitude can be achieved by selecting an appropriately large size for region X. Further introduced methods of this chapter are independent of the specific formulation of the rate function λ. Regarding the notation, a continuous function $\lambda\left(\boldsymbol{x}_\mathrm{s}, t\right)$ is assumed.

5.4.2 Definition of Atomic Actions

Based on a model of human activity, the goal is to set up a sequence of actions to maximize the probability of meeting a person, or in other words, to determine if and where the robot should search for or wait for people. These behaviors are realized by stochastic BT actions, i.e., by defining nodes of an SBT (cf. section 5.2.2). To create the tree, the atomic actions $\mathcal{W}_{\mathrm{A},i}$ (*Wait at place* $\boldsymbol{x}_i$) and $\mathcal{S}_{\mathrm{A},i\to j}$ (*Search from place* $\boldsymbol{x}_i$ *to place* $\boldsymbol{x}_j$) are defined, where a *place* $\boldsymbol{x}_i \in \mathbb{R}^2$ refers to a point on the map of the environment. As the robot perceives the environment only partially, the detection area D defines the robot's field of view. Following sections 2.1.2 and 4.2.2, this area can assumed to be a circle with radius r_D. Figure 5.8 illustrates the wait and search actions.

It is assumed that there is a number of n_pl known places the robot could move to and wait at, including the robot's position $\boldsymbol{x}_0$, which is the location where help is needed (section 5.4.3 shows how n_pl and the corresponding places are determined). To decide between different behaviors (i.e. behavior trees), the tuple $A_\mathrm{sbt} : \langle p_\mathrm{s}(t), p_\mathrm{f}(t), \mu, \nu \rangle$ must first be defined for each type of action.



(a) Wait action (b) Search action

Figure 5.8: Exemplary arrangement of different search locations $\boldsymbol{x}_i$ and paths $S_{i\to j}$ in case a robot needs to enter a room. Detection areas with detection radius r_D are denoted as D_i.

The wait action $\mathcal{W}_{\mathrm{A},i}$ returns *S* when a person is found and *F* when a maximum time has been reached. The exponentially distributed success rate μ_{w} then results directly from the property that the interarrival times of people are exponentially distributed. When the robot starts waiting at a time t_0 at a location $\boldsymbol{x}_i$, the probability density function of the exponential distribution

$$f(t;\mu_{\mathrm{w}}) = \begin{cases} \mu_{\mathrm{w}} e^{-\mu_{\mathrm{w}} t} & t \geq t_0 \\ 0 & t < t_0 \end{cases}, \quad \text{with} \quad \mu_{\mathrm{w}} = \int_{D_i} \lambda\left(\boldsymbol{x}_{\mathrm{s}}, t = t_0\right) \mathrm{d}\boldsymbol{x}_{\mathrm{s}} \tag{5.9}$$

describes the waiting time. This assumes that the rate λ can be considered constant while waiting. The corresponding cumulative distribution function

$$p_{\mathrm{s,w}}(t;\mu_{\mathrm{w}}) = \begin{cases} 1 - e^{-\mu_{\mathrm{w}} t} & t \geq t_0 \\ 0 & t < t_0 \end{cases} \tag{5.10}$$

gives the probability of meeting a person. By manually specifying the desired confidence $p'_{\mathrm{s}} > p_{\mathrm{s,w}}(\mu_{\mathrm{w}}^{-1};\mu_{\mathrm{w}})$, this equation can be rearranged to estimate the expected waiting time T' until the next person appears. If no person appears after T', the wait action is considered as failed, resulting in the mean time to fail:

$$\nu_{\mathrm{w}}^{-1} = T' = -\log(1 - p'_{\mathrm{s}})\, \mu_{\mathrm{w}}^{-1}. \tag{5.11}$$

Because the wait action will never fail before T' has passed, the failure probability is defined as

$$p_{\mathrm{f,w}}(t;\mu_{\mathrm{w}}) = \begin{cases} 1 - p'_{\mathrm{s}} & t \geq t_0 + T' \\ 0 & t < t_0 + T' \end{cases}. \tag{5.12}$$

For proactive search, a search path is defined as

$$S_{i \to j} : \langle \boldsymbol{x}_i, \boldsymbol{x}_j, \mathbb{G}, l_{\mathrm{sp}}, \bar{v}_{\mathrm{sp}} \rangle, \quad i, j \in \{0, 1, ..., n_{\mathrm{pl}}\}, \quad i \neq j, \tag{5.13}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the start and end places, $\mathbb{G} \subset \mathbb{R}^2$ is the geometric description, l_{sp} is the length of the path and $\bar{v}_{\mathrm{sp}}$ the average velocity of the robot while driving on the path ("sp" denotes *search path*.). $\mathbb{G}$ and l_{sp} result directly from a geometric path planner (such as the A* algorithm on an occupancy grid map), and $\bar{v}_{\mathrm{sp}}$ can either be determined empirically or based on the settings of a local path planner. Similar to the wait action, the search action $\mathcal{S}_{\mathrm{A},i \to j} = A_{\mathrm{sbt}} \cup \mathcal{S}_{i \to j}$ returns *S* as soon as a person is found and *F* if the whole path was driven without finding anyone. Additionally, this action can also return *F* if the navigation execution fails, e.g., due to an obstructed goal. While the robot moves on the path, it observes different areas of the human-activity model, each for an individual time. This imposes a time dependency on the success rate

$$\mu_{\mathrm{sp}}(t) = \int_{D(t)} \lambda\left(\boldsymbol{x}_{\mathrm{s}}, t = t_0\right) \mathrm{d}\boldsymbol{x}_{\mathrm{s}}. \tag{5.14}$$

The rate is calculated by discretizing the path with $t_k = k\,\Delta t$ and calculating the corresponding rate $\mu_{\mathrm{sp},k}$ for each point in time. Figure 5.8.b illustrates this as an example of different paths in an environment.

The minimal time until finding a person on the path results from the sum of the time t_k and the expected arrival time (as in eq. 5.11) to

$$\mu_{\mathrm{sp,tot}}^{-1} = \underset{t_k \in [t_0, t_0 + l_{\mathrm{sp}}/\bar{v}_{\mathrm{sp}}]}{\arg\min} \left(t_k - \log(1 - p'_{\mathrm{s}})\, \mu_{\mathrm{sp},k}^{-1} \right). \tag{5.15}$$

Here, p'_{s} again is a confidence value, and t_0 is the time when the search is started. The success probability $p_{\mathrm{s,sp}}(t)$ of finding a person on the path follows as the converse probability of not having found anyone until time t to

$$p_{\mathrm{s,sp}}(t; \mu_{\mathrm{sp}}) = 1 - \exp \int_{t_0}^{t} \mu_{\mathrm{sp}}(\tilde{t}) \mathrm{d}\tilde{t}. \tag{5.16}$$

If nobody has been found until the goal is reached or the navigation fails, the action *search person* is considered as failed. The fail rate ν_{sp} is approximated by

$$\nu_{\mathrm{sp}} = \frac{\bar{v}_{\mathrm{sp}}}{l_{\mathrm{sp}}} + \frac{\bar{v}_{\mathrm{sp}}}{l_{\mathrm{fail}}}, \tag{5.17}$$

where l_{fail} is an expected distance the robot can move until a navigation failure occurs, which is also assumed to be exponentially distributed. This value can, for instance, be estimated by observation. Before the proactive search action is finished, the action can only fail due to the navigation and only after it has been finished due to not finding a person. The probability $p_{\mathrm{f,sp}}(t; \nu_{\mathrm{sp}})$ to fail is, therefore, defined as a piece-wise function:

$$p_{\mathrm{f,sp}}(t; \nu_{\mathrm{sp}}) = \begin{cases} 1 - p_{\mathrm{s,sp}}(\nu_{\mathrm{sp}}^{-1}; \mu_{\mathrm{sp}}), & t \geq \nu_{\mathrm{sp}}^{-1} \\ 1 - \exp(-\frac{\bar{v}_{\mathrm{sp}}}{l_{\mathrm{fail}}}\, t), & t < \nu_{\mathrm{sp}}^{-1} \end{cases}. \tag{5.18}$$

Introducing l_{fail} into eq. 5.17 increases the probability of failure for longer search paths, giving preference to shorter paths with an otherwise equal chance of finding a person. For eq. 5.18 to be applicable, the condition

$$p_{\mathrm{s,sp}}(\nu_{\mathrm{sp}}^{-1}) \leq \exp(-\frac{l_{\mathrm{sp}}}{l_{\mathrm{fail}} + l_{\mathrm{sp}}}) \tag{5.19}$$

must hold due to the law of total probability. This is fulfilled for $l_{\mathrm{sp}} \ll l_{\mathrm{fail}}$, which is the case for the present application since all longer paths can be discarded to avoid searching far away from the help location.

5.4.3 Stochastic Behavior-Tree Synthesis

Based on the wait and search actions, an action rule must now be found that maximizes the probability of finding a person, considering the return time to the help location. Therefore, the *person search behavior tree* (PSBT) is defined as a sequence of actions that should be executed when the robot faces a task that cannot be solved by itself. For this, a fallback behavior is defined, with a general form as shown in Figure 5.9.

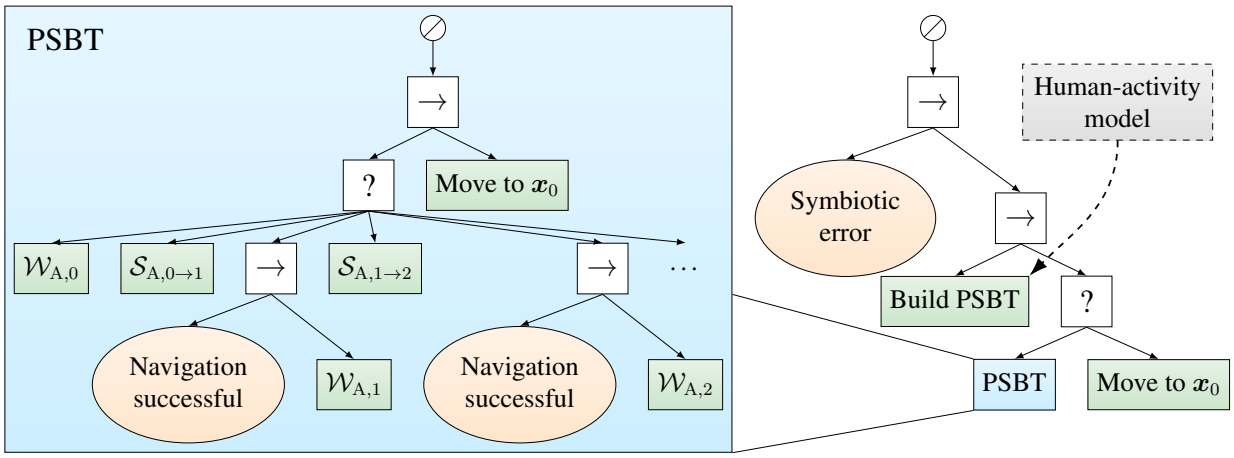


Figure 5.9: General form of the PSBT. This tree is executed when the robot needs human help (Symbiotic error $\leftarrow \mathrm{True}$), for example, when a door must be opened and the robot has no manipulators.

The tree contains a $\mathrm{Move\ to}\ \boldsymbol{x}_0$ action, which describes the movement back to the start location and is interpreted as a search action that can only fail due to the navigation. To create the PSBT, $\lambda\left(\boldsymbol{x}_\mathrm{s}, t\right)$ is spatially discretized and a number of n_pl cells is sampled, where the probability p of a cell to be selected is $p \propto \int_\mathrm{cell} \lambda\left(\boldsymbol{x}_\mathrm{s}, t = t_0\right) \mathrm{d}\boldsymbol{x}_\mathrm{s}$.

If sampled cells are close to each other (the EUCLIDEAN distance is smaller than the radius r_D of the FOV D), only the cell with the larger rate λ_cell is kept. Also, only cells with variance and distance to the robot below-specified thresholds can be sampled. Subsequently, all wait actions $\mathcal{W}_{\mathrm{A},i}$ and search paths $\mathcal{S}_{\mathrm{A},i\to j}$ $(\forall i, j \in \{0, 1, ..., n_\mathrm{pl}\},\ i \neq j)$ are calculated. The search order is set by means of an open traveling salesman problem (OTSP) with places $\boldsymbol{x}_{0\ldots n_\mathrm{pl}}$ as nodes and the inverse of the failure rates $\nu^{-1}_{\mathrm{s},i\to j}$ (expected times to fail) as costs for the corresponding graph. This is required to reduce the complexity $\mathcal{O}(n_\mathrm{pl}!)$ of investigating all possible sequences to search all places $\boldsymbol{x}_{1\ldots n_\mathrm{pl}}$ The OTSP is solved by genetic algorithm [Dav10]. The next step is the stochastic analysis of all possible BTs to obtain an estimate of the success probability $p_{\mathrm{s,T}}(t)$ and success rate μ_T of the tree (according to eq. 5.1 and eq. 5.3) as the basis for decision-making. Each tree's flow is decomposed as a DTMC by solving the CAUCHY problem in eq. 5.2 with the infinitesimal generator matrix $\boldsymbol{Q}(\mathbb{A}_\mathrm{sbt,sp}, \mathbb{A}_\mathrm{sbt,w})$ until a pre-defined look-ahead time t_max, with all search and wait actions collected in $\mathbb{A}_\mathrm{sbt,sp}$ and $\mathbb{A}_\mathrm{sbt,w}$, respectively. The decomposition is calculated for every possible case, i.e., driving to specific places (or not), waiting there (or not), and finally returning

to the help location, leading to a total number of $3^{n_{\text{pl}}} + 1$ executions. The preferred tree is then chosen as the tree with maximum $p_{\text{s,T}}(t_{\max})$.

5.4.4 Results

The experimental evaluation is divided into two parts: firstly, the performance evaluation of PSBT based on the human-activity model, and secondly, a comparison of the time to find people under real-life conditions with other methods. The ground floor of a multi-story 50 m by 25 m university building serves as the environment for evaluation, as depicted in Figure 5.10. The area includes lecture halls, a cafeteria, several entrances, and sitting areas. For the real-world experiments, the mobile social robot Sobi is used with the people perception and tracking pipeline presented in section 2.1.3. A human-activity model is trained as a grid model (cf. eq. 5.7) with data from two working days by moving the robot throughout the day to different places on the entire floor so that the same locations are observed at different times of day. Only people tracks with a minimum observed time of 3 s in a range of 5 m are used for training, resulting in a total number of 18,362 tracks. This relatively high number resides in the fact that the robot partly loses the same person for a short time and then re-detects them as a new track. It is also considered a new track if a person moves more than one meter or is detected for more than 20 s. The following parameters were chosen for the experiments: $m_{\text{grid}} = 50$, $o_{\text{grid}} = 25$ (grid cell resolution of 1 m), detection radius $r_D = 2\,\text{m}$, confidence $p'_{\text{s}} = 0.9$, path discretization $\Delta t = 1\,\text{s}$, expected distance $l_{\text{fail}} = 100\,\text{m}$, average velocity $\bar{v}_{\text{sp}} = 0.5\,\text{m}\,\text{s}^{-1}$, sampled goals $n_{\text{pl}} = 6$, DTMC look-ahead time $t_{\max} = 200\,\text{s}$.

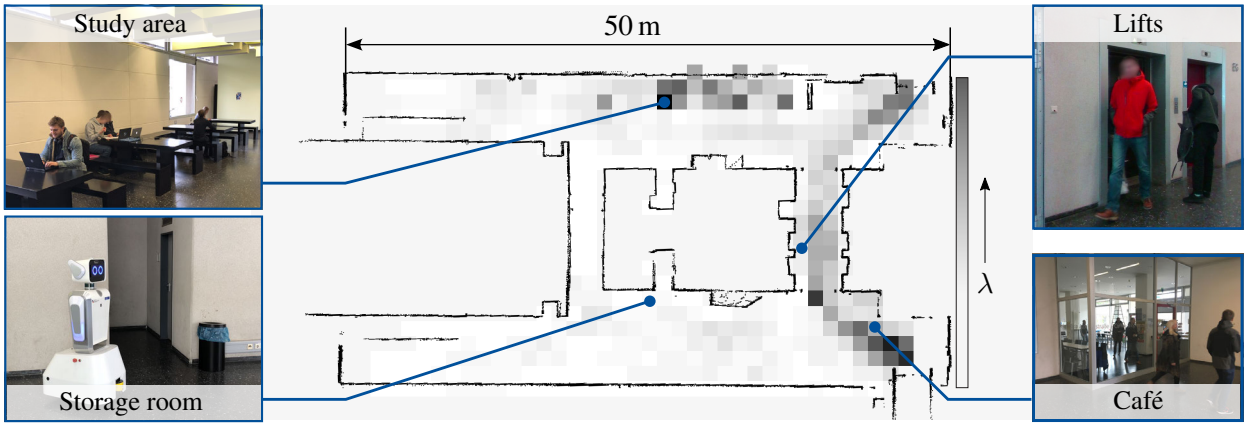


Figure 5.10: Occupancy map and human-activity model of the evaluation environment.

Model-based Evaluation

person search behavior tree (PSBT) is compared with five other strategies:

Greedy planning to the global maximum (GM) heuristically plans to the location with maximum occurrence rate λ and waits there.

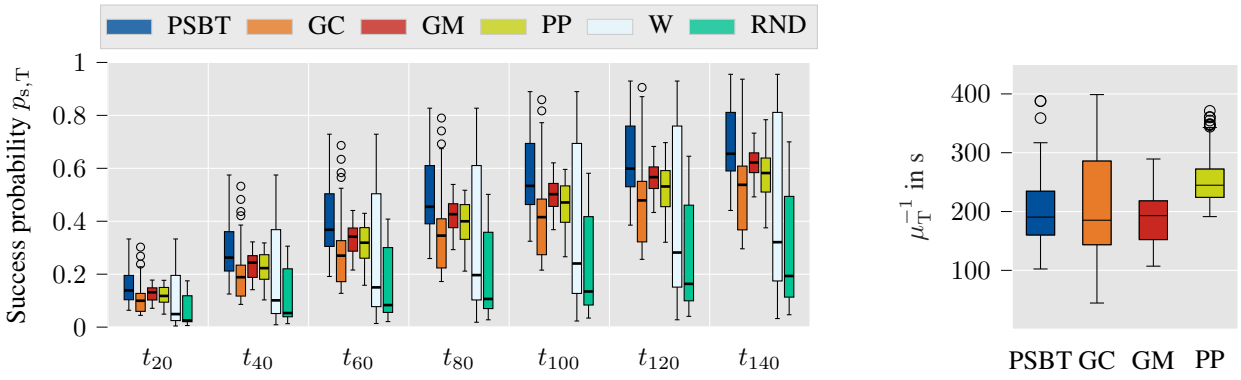
Greedy planning to a close maximum (GC) is similar to GM, but plans to the closest cell out of the n_λ cells with highest rate and waits there ($n_\lambda = 50$ in this evaluation).

Uniform random sampling of goals (RND) samples the same number of goals n_{pl} like PSBT without regarding the rates of the corresponding cells and randomly waits at the locations.

Wait at the help location (W) is the trivial approach as described.

Purely proactive (PP) samples the goals in the same manner as PSBT, but never waits at sampled locations in order to only search proactively.

Firstly, the different methods are compared offline based on the trained human-activity model. Five hundred accessible poses are sampled randomly on the occupancy grid map of the building as starting poses for the search for helpers. Then, the probability of success $p_{\mathrm{s,T}}(t_k)$ is calculated using the methods presented in section 5.4.2. The comparison is shown in Figure 5.11.a for seven points in time (every 20 s).



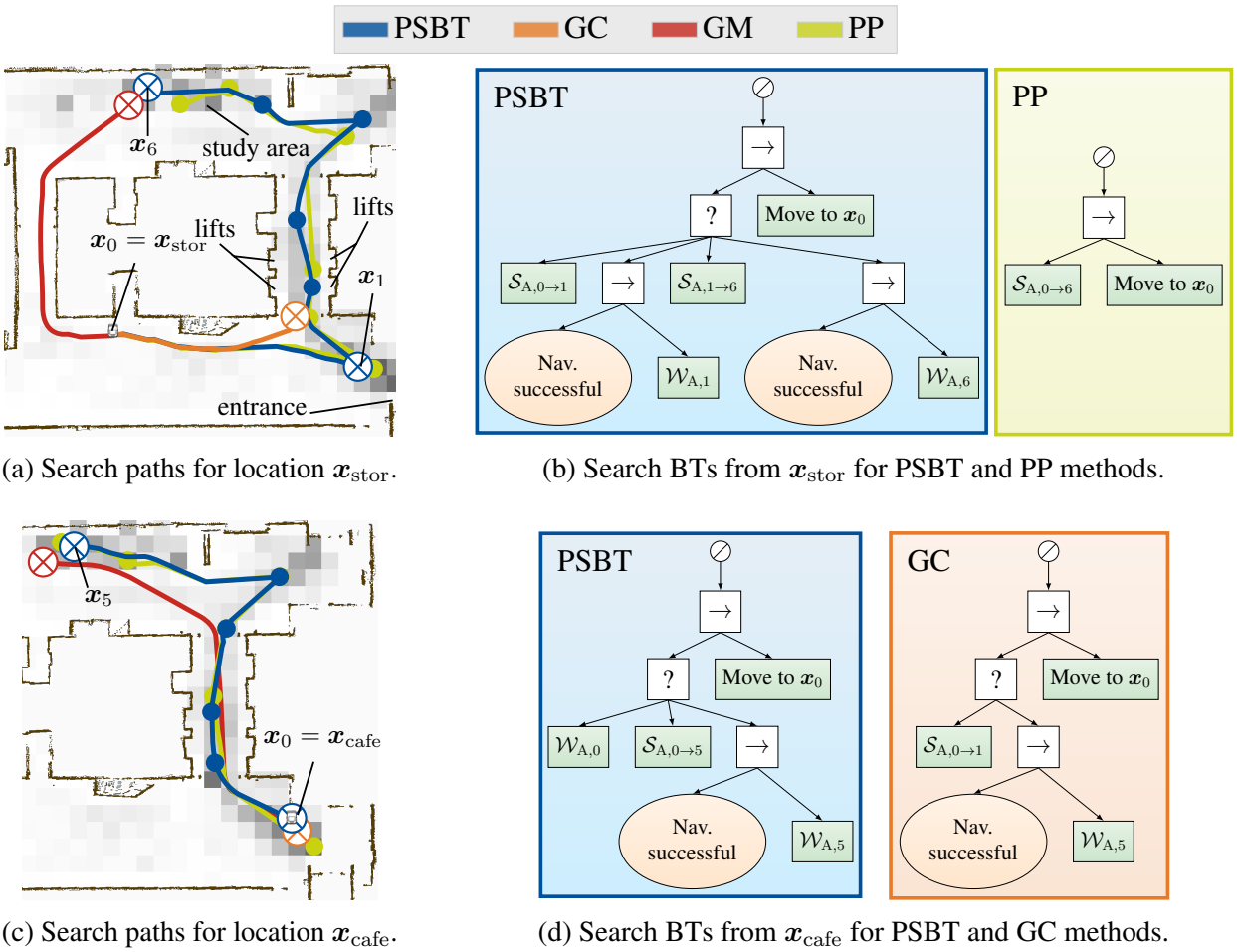
(a) Success probability distribution of the BT root nodes at seven points in time for 500 randomly sampled starting poses on the map.

(b) Expected time to success μ_{T}^{-1}.

Figure 5.11: Metrics for model-based evaluation

Although the observable variance shows a dependency on the starting pose, PSBT outperforms the other methods, leading to an on average 5.7 % higher probability of success after 140 s than the following best method (GM). The PSBT is always at least as good as waiting at the help location (W) because the stochastic analysis also explicitly considers this case and can execute it accordingly. Figure 5.11.b shows the expected time to success μ_{T}^{-1} for the four best methods. The time μ_{T}^{-1} refers to the expected time to find a person and then return to the help location. Here, the expected time for the PSBT is ca. 11 s higher on average than for GM, which is due to the property that

the DTMC decomposition yields more conservative estimates of the success rate for an increasing number of fallback child nodes. Additionally, the resulting probability distribution $p_{\mathrm{s,T}}(t_k)$ from the DTMC is not exponential, so it is not precisely correlated to μ_{T}^{-1}. The performance of the GC method is strongly location dependent, which is reflected in its spread of data for μ_{T}^{-1}. Figure 5.12 shows two exemplary help locations in front of doors and the corresponding BTs. The location $\boldsymbol{x}_{\mathrm{stor}}$ (cf. Figures 5.12.a and 5.12.b) is in front of a door to a storage room without a large number of people coming and going. While the greedy strategies aim for the closest (or global) maxima, the PSBT strategy moves to the nearby building entrance ($\boldsymbol{x}_1$), waits there, and then proceeds to move along the lifts to the global maximum (a sitting area). The second start location is one of the entrances to the cafeteria $\boldsymbol{x}_{\mathrm{cafe}}$ (cf. Figures 5.12.c and 5.12.d), which is located in an area with a higher volume of people. Here, the PSBT strategy first waits on site before proactively searching for people, while the greedy strategies again aim for the maxima.



(a) Search paths for location $\boldsymbol{x}_{\mathrm{stor}}$.

(b) Search BTs from $\boldsymbol{x}_{\mathrm{stor}}$ for PSBT and PP methods.

(c) Search paths for location $\boldsymbol{x}_{\mathrm{cafe}}$.

(d) Search BTs from $\boldsymbol{x}_{\mathrm{cafe}}$ for PSBT and GC methods.

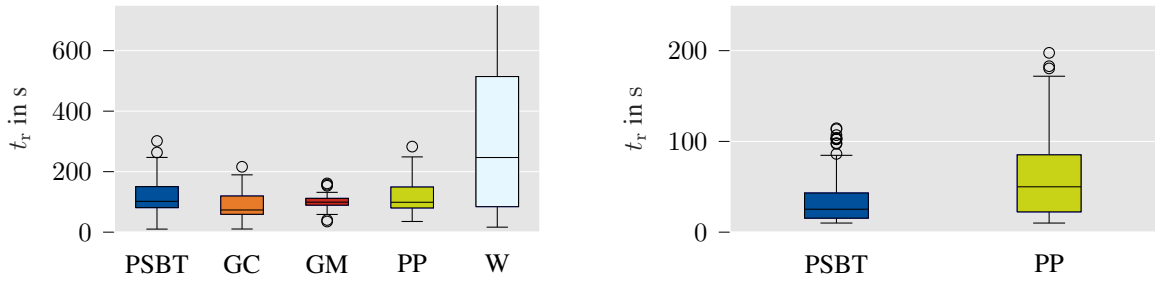
Figure 5.12: Exemplary search paths and corresponding BTs of PSBT and selected methods for comparison.

Real-World Experiments

Additionally, real-world experiments are conducted to demonstrate the efficiency of the presented approach. The aforementioned methods are tested over five working days for the two different start locations from Figure 5.12. The same constant human-activity model is used for all methods, which are tested alternatingly to compensate for biasing effects (e.g. events in the lecture halls or the time of day).

A single run includes online planning according to the corresponding method and the subsequent search for people. Instead of approaching the sampled goals directly, the closest reachable pose in a radius of 2 m around the goal is chosen, allowing the sensors to be oriented towards the goal. A person is considered as found if a detected track $\boldsymbol{d}_{\mathrm{ped},k}$ of a person is within the detection radius r_D in front of the robot for a period of at least 3 s. After a successful search, the robot returns to the starting position, and the required time for the whole run t_r is saved. A run is considered failed when the search tree returns F, i.e., when all search and wait actions were executed without finding anyone. Successful searches based on false-positive person detections are removed manually from the evaluation.

The waiting time for all methods is determined according to eq. 5.11 based on the confidence value p'_s. Figure 5.13.a shows the results for $\boldsymbol{x}_\mathrm{stor}$ as the starting location. All proactive methods perform significantly better than waiting at the help location (W). Table 5.4 summarizes the mean time $\bar{t}_\mathrm{r}$ with standard deviation, which was calculated for the successful runs only.



(a) Start location $\boldsymbol{x}_\mathrm{stor}$ (b) Start location $\boldsymbol{x}_\mathrm{cafe}$

Figure 5.13: Run time t_r for successful searches.

Although the greedy methods can be faster, PSBT-based search is successful in substantially more cases. Since the greedy methods only approach one maximum and remain there, the success depends strongly on whether people can be recognized at this specific location. For example, the global maximum in this environment was a resting area, often occupied by people in slightly different spots, which are then difficult to detect when using a static waiting location. This is also reflected in the high success rate of the PP method, as it is more likely to detect a person directly in front of the robot during the proactive search. A low success rate is also problematic in that it takes time to determine that all actions have failed (here, always more than 200 s). Therefore, PSBT would still be faster on average compared to GC/GM. An essential advantage of the PSBT method

		Experimental results			Model estimation	
Place	Method	Trials	% successful	$\bar{t}_\mathrm{r}$ in s	$\bar{\mu}_\mathrm{r}^{-1}$ in s	$\bar{p}_{\mathrm{s,T}}(t_\mathrm{max})$
$\boldsymbol{x}_\mathrm{stor}$	PSBT	86	**98.8 %**	120.5±61.2	139.5±8.7	0.87±0.01
	PP	93	88.2 %	115.4±52.3	150.8±7.0	0.85±0.01
	GM	86	62.8 %	101.1±22.4	121.5±0.0	0.81±0.00
	GC	90	65.6 %	**89.1**±49.7	108.1±0.0	0.81±0.00
$\boldsymbol{x}_\mathrm{cafe}$	PSBT	112	**91.1 %**	**35.5**±28.2	49.9±0.0	0.99±0.00
	PP	121	90.1 %	63.0±48.9	126.8±6.0	0.86±0.01
Total	PSBT	198	94.4 %	–	–	–

Table 5.4: Results of the experiments with averagely estimated expected time to success $\bar{\mu}_\mathrm{r}^{-1}$ and averagely estimated success probability $\bar{p}_{\mathrm{s,T}}(t_\mathrm{max})$. Quantitative results are given as mean ± std. deviation.

can be seen for highly frequented areas like $\boldsymbol{x}_\mathrm{cafe}$ (see Figure 5.13.b). Here the two best methods from $\boldsymbol{x}_\mathrm{stor}$ are compared to emphasize the difference between exclusively proactive search (PP) and intermittent waiting (PSBT). The exclusively proactive PP method takes longer on average to find a person and return to the start location since PSBT correctly decides that it is more worthwhile to wait on-site. Although not evaluated, it should be noted that the greedy strategy GC would lead to similar results as PSBT here (see, e.g., the waiting location of GC in Figure 5.12.b).

Based on the expected time to success μ_r^{-1} and success probability $p_{\mathrm{s,T}}(t_\mathrm{max})$ the model gives an estimate of the performance of the different methods and the necessary time to find a person, thus providing a good estimation for decision making. It provides more conservative estimates for the duration since cells with high variance are ignored in the calculation. Furthermore, the assumed average velocity of the robot $\bar{v}_\mathrm{sp}$ during the movement was often exceeded in the test runs. The calculation time of the PSBT averaged 4.8 s for all test runs (Intel i7-7700T CPU), which allows for online execution. To summarize, in 198 trial runs, the proposed method PSBT can produce suitable predictions of the time until success and found people in 94 % of all cases, which is a higher rate than all other compared methods.

5.5 Conclusion

This chapter introduces a reactive task-control and monitoring system for long-term-autonomous robots. Using behavior trees (BTs), a modular system extends arbitrary task-control mechanisms by setting starting and stopping conditions and monitoring system variables. Since the reliability of a robot's software components is essential for long-term-autonomous applications, executed programs and basic skills, such as localization and navigation, are monitored and recovered in prioritized order. The system is evaluated in a continuous 16-day deployment of a service robot, and typical LTA metrics are recorded. The results show that the automated failure recovery strategies lead to uninterrupted autonomy for multiple days.

For failure cases that can neither be solved autonomously nor by remote supervision, an approach is introduced to involve humans in problem-solving proactively. With PSBT, a method is presented to find potential helpers in problematic situations, such as blocked passages or closed doors. PSBT automatically synthesizes a BT based on an activity model that indicates the occurrence rate of people in an environment. Especially compared to purely proactive search, or not searching at all, the search time can be significantly reduced by actively deciding for or against waiting at the help location. Again, the BT framework enables a modular formulation of the search problem and easy future expandability to include further actions, such as approaching and talking to people or accompanied drives to the help location. This distinguishes PSBT from state-of-the-art methods, which usually employ tailored low-level cost functions that are hard to extend with actions of a different type. One limiting feature of PSBT is its $\mathcal{O}(3^{n_{\mathrm{pl}}})$ complexity, constraining its application to a few individual search locations. Although this is not necessarily a disadvantage in practice (symbiotic search is only reasonable in close vicinity), heuristics or approximations could be investigated to reduce complexity in large-scale environments.

6 Conclusion and Future Work

Technological advances have enabled robotic systems to make their way into many new areas over the past decades. They can be used in increasingly complex environments for extended periods. Service robots that perform simple cleaning tasks at home, make hotel deliveries, or provide information services in shopping malls are now part of many people's everyday lives. To be helpful and to relieve humans of tasks, these robots require a degree of autonomy. However, achieving reliable autonomy over long periods is a significant challenge regarding the robot's hardware and software components. With the assumption of static environments, many methods exist in the state of the art in areas like localization, environment modeling, navigation, and task planning. In real applications with multiple moving people and objects, changes over time, and varying environmental influences, such an assumption is no longer valid. Therefore, intelligent robots that dynamically modify and expand their internal models, learn continuously, and adapt their behavior are a highly relevant research topic.

6.1 Summary

This thesis proposes several approaches to improve the long-term autonomy of service robots. In contrast to many works from the state-of-the-art, a holistic research effort with experiments based on real-world data and an actual robot is chosen for this purpose. Besides introducing a new service robot system, methods for simultaneous localization and mapping, modeling of human-induced dynamics, and subsequent task execution are presented.

The basis for experimental investigations is the service robot Sobi, designed for applications inside and outside buildings. These varying characteristics are considered in the component selection and design, enabling robust sensory perception of environmental features. One focus is the person-detection pipeline that uses multi-modal sensor information. Experimental results demonstrate its suitability for further processing in terms of precision and accuracy.

The idea of using multi-modal sensor information is then carried forward to improve the robustness and accuracy of localization and mapping. Research question 1 (cf. section 1.2) is answered in summary by extending (graph-based) SLAM methods to further utilize multi-modal sensor data. Initially, the V-SLAM method RTAB-Map is modified to include 3D Lidar data, as significantly more environmental features and a larger field of view are taken into account. The combined classification and registration routine allows improved mapping and localization in dynamic environments. A map-management system efficiently captures heterogeneous environments by

partitioning, specifically parameterizing, and graphically linking them. Resultingly, improvements in terms of map quality are observed.

Based on the assumption of robust self-localization and a known map of the static areas of the environment, chapter 4 then introduces the continuous pedestrian activity map (CoPA-Map). As a regression model, the method allows spatio-temporal prediction of person occurrence in a continuous representation. Consistent with research question 2, the regression is performed based on long-term observations of humans by a mobile robot. As an answer to this research question, it can be concluded that a heteroscedastic likelihood function can effectively represent the heterogeneous distribution of detected people resulting from moving robots. Besides classical error measures on the prediction quality, it is shown that CoPA-Map is especially suitable for effective path planning.

Building on the idea of using human-activity models for effective path planning, an executive system for LTA is introduced to actively search for people in the event of faults. First, it is shown for the Behavior-Tree-based system that targeted recovery behaviors enable long-term-autonomous performance using the robot Sobi as an example. The last research question 3 is then addressed by synthesizing a behavior tree for efficient person search in case of problems based on a human-activity model. It shows that people can be found faster than with other methods and that an interpretable and extensible task description structure emerges.

6.2 Future Research Directions

The various approaches presented in robot design, SLAM, activity modeling, and task control represent contributions to diverse areas of long-term autonomy. This diversity raises further research questions in different categories summarized below.

6.2.1 Robotic Platform and SLAM

Long-term autonomy in mixed environments. Long-term tests with the service robot Sobi are presented in section 5.3.2 for an indoor area. Although the robot was also designed for outdoor areas and methodologically evaluated to some extent (cf. section 3.2.3, section 3.3.4), long-term investigation of the autonomous system in these areas is an open problem. Varying weather and temperature influences are particularly relevant. Extending the open source system [Stu22] with additional sensors is conceivable. For instance, humidity and air pressure sensors could help in task planning to avoid rain and to determine the current level in a building, respectively.

Absolute positioning. All presented localization and mapping techniques are based on relative self-pose estimation via the robot's perceptive sensors. Although this is a common approach, it requires making certain assumptions in chapter 3. The search area for loop closures is generally

defined based on the odometry's uncertainty estimate, which is not feasible when an existing map is extended. Additionally incorporating absolute position information would allow to limit the search area globally, reduce errors in strongly repetitive areas, and select maps spatially. Depending on the robot's deployment area, WiFi mapping [AD19], Bluetooth beacons [GFFM18], or global positioning system (GPS) are suitable options.

Dynamic map modification. In this work, an increase in the robustness and accuracy of SLAM methods is generated primarily by a strong focus on multi-modal sensor properties to create a robust map representation. For example, the fact that laser scans improve localization regardless of ambient light is primarily inherent to the technology and not only to the method used. As presented as part of related works (cf. section 3.1.2), map representations may be modified and extended in a time-dependent manner. When the environment changes, already mapped areas could be exchanged, subdivided [HDFF16], or recorded with a new configuration. Until now, the individual configurations are manually parameterized to predefined conditions (short-range, long-range areas). Parameterizing them automatically via feedback of map consistency and localization quality could further reduce the need for expert knowledge.

6.2.2 Spatio-Temporal Human-Activity Map

Adding direction and speed. CoPA-Map is created as a single-output GAUSSIAN process model that indicates human activity in terms of people occurrence rate. As seen in the last chapter 5, a scalar-valued model is sufficient to find people efficiently. However, adding other output variables, most importantly predicted mean velocity and direction of motion, offers advantages [VMS+19; MCD21]. Mobile robot systems could thus adapt their path planning to the preferred walking direction and act cautiously when there is a rapid flow of people. One approach to incorporate these outputs into CoPA-Map could be to keep heterogeneous inputs and utilize multi-output GAUSSIAN processes [MAA18]. However, the multi-modal characteristics of motion directions could be challenging to capture with a single GP model. In this case, combining different GP experts or Bayesian committee machines are potential research directions [LOSC20].

More prior information. An essential part of CoPA-Map is the introduction of prior information like periodicities and smooth spatial interrelationships as part of the GP framework. The implemented kernels are stationary, i.e., the covariance between input points depends solely on their Euclidean distance. However, people's movement is strongly non-stationary due to obstacles, pre-defined walking paths, or grouping effects. Additional prior information, e.g., a map of the static environment (walls), could be used to improve prediction quality further. Currently, CoPA-Map utilizes occupancy map representations only to calculate observation durations as part of the training dataset. These maps could also be incorporated to encode that people usually move away from obstacles (and not through walls). Implementation could be realized by constraining the

GP model spatially [SK19] or by a combination with pre-trained models that output predominant directions given the map [VKK22].

Sparse data in large-scale. For the evaluation of CoPA-Map, different simulated paths are used, which cover a maximum area of $100\,m^2$–$200\,m^2$ (cf. section 4.3.4). Two properties mainly limit the application in larger environments. First, increasing data would require more advanced approximations of the GAUSSIAN process regression. In addition, very infrequent visits to certain areas would make temporal reconstruction (frequency analysis) difficult. A combination with the map management system (cf. section 3.3) could be explored for the first aspect. Separate GAUSSIAN processes could be investigated using Product-of-Expert, or Mixture-of-Experts approaches [LOSC20]. One relevant question in the context of multi-layer map arrangements would be whether the input resolution can be reduced to such an extent that single predictions are made, e.g., for entire rooms. This would also reduce the need for the robot to visit all individual, small-spaced, locations frequently.

6.2.3 Executive System for Long-Term Autonomy

Completing the symbiotic process. The method PSBT is introduced to efficiently search for people as part of an LTA monitoring system. One assumption is that finding people faster will lead to faster problem-solving. In further work, this assumption should be softened by implementing further steps of actively asking for help. This includes approaching a person, verbally asking for help, moving together back to the help location, and recognizing successful problem-solving. There are many overlaps to the research area of human-robot interaction that could be investigated, e.g., in the form of social acceptance studies. In this regard, it should be noted that the influence of the robot itself on the environment has yet to be considered in the methods presented. People behave differently when a robot is present than when it is absent (wow-effect [FDO+20]), influencing human-activity modeling and people search.

Integrating planning and scheduling. The presented Behavior-Tree-based monitoring system acts reactively, inter alia, by utilizing a model of human activity. In long-term operation, it may be desirable to actively extend this model, e.g. by employing CoPA-Map's predictive uncertainty estimation. The timing of exploration activities requires a combination with a task scheduling system since the robot must continue to perform its services. Additionally, the model of human activity could be used in problem cases and for active task planning. For example, an info-terminal robot could offer its services when there is a high volume of people and make deliveries at other times. Current research implies the suitability of planning-based synthesis of behavior trees for this automated task generation [LPS+21; SIN+22].

A Lidar-based Loop Closure Detection: Parameters for Evaluation

Parameter	Description	Indoor	Outdoor	KITTI
	Loop search			
$p_{\min}$	Loop detection threshold	52.4 %	52.9 %	52.4 %
$r_{\min}$	Minimum loop search radius	3 m	3 m	7.5 m
β	Scale factor for uncertainty-dependent search radius	0.25	0.25	0 (n.a.)
	Registration			
l_{vox}	Voxel cube length for filtering pre-registration	0.03 m	0.03 m	0.2 m
r_{lim}	Maximum EUCLIDEAN distance from sensor origin for filtering pre-registration	30 m	30 m	30 m
i_{lim}	Minimum point intensity for filtering pre-registration	5	5	0 (n.a.)
z_{lim}	Minimum point height for filtering pre-registration	0.3 m	0.3 m	1 m
$n_{p,\max}$	Maximum number of points after heuristic down-sampling	7,000	7,000	7,000
$t_{\max}$	Maximum translation for registration	3 m	3 m	10 m
	Verification			
$\alpha_{\min}$	Minimum ratio between local nodes and nodes in WM to conduct local search	0.5	0.5	0.5
n_{start}	Number of successful loop closures before changing to local search	3	3	3
n_{ver}	Number of verification nodes in neighborhood for global search	2	2	2
r_{ver}	Radius of neighborhood for global search	5 m	5 m	5 m

Bibliography

During the work at the Institute of Mechatronic Systems, an ongoing publication of individual subtopics of this dissertation took place. In total five international conference papers [SNTO19; SWTO19; SLPS21; SWSS21; SS22] were published as first author. Although this is the author's own content, references to these publications are neither omitted nor marked separately in the text. The same applies to published contributions as co-author [ESNO20; HSLS21] and the referenced supervised student theses [Pet19; Ben21; Har20; Wes21].

The following figures use modified and unmodified graphical resources from [Vec23]: Figure 1.3, Figure 3.5, Figure 3.11.a, Figure 4.2.a, Figure 5.8.

[ACM+06] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila. "Toward Human-Aware Robot Task Planning." In: *AAAI spring symposium: to boldly go where no human-robot team has gone before*. 2006, pp. 39–46.

[AD19] C. Adhivarahan and K. Dantu. "WISDOM: WIreless Sensing-assisted Distributed Online Mapping". In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8026–8033.

[Ant14] G. Antonelli. "Fault detection/tolerance strategies for AUVs and ROVs". In: *Underwater Robots*. Springer, 2014, pp. 101–116.

[APSL08] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó. "The SLAM problem: a survey". In: *Artificial Intelligence Research and Development* (2008), pp. 363–371.

[BAYG17] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. "Simultaneous localization and mapping: A survey of current trends in autonomous driving". In: *IEEE Transactions on Intelligent Vehicles* 2.3 (2017), pp. 194–220.

[BC10] J. Bohren and S. Cousins. "The SMACH High-Level Executive [ROS News]". In: *IEEE Robotics & Automation Magazine* 17.4 (2010), pp. 18–20.

[BCF+98] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. "The Interactive Museum Tour-Guide Robot". In: *AAAI Conference on Artificial Intelligence*. 1998, pp. 11–18.

[BD05] P. Biber and T. Duckett. "Dynamic Maps for Long-Term Operation of Mobile Service Robots". In: *Robotics: Science and Systems*. 2005.

[BD06] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II". In: *IEEE Robotics Automation Magazine* 13.3 (2006), pp. 108–117.

[BETV08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-up robust features (SURF)”. In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359.

[BFG08] J.-L. Blanco, J.-A. Fernández-Madrigal, and J. Gonzalez. “Toward a unified Bayesian approach to hybrid metric–topological SLAM”. In: *IEEE Transactions on Robotics* 24.2 (2008), pp. 259–270.

[BJL+16] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele. “Room segmentation: Survey, implementation, and analysis”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1019–1026.

[BJMR05] J. L. Bresina, A. K. Jónsson, P. H. Morris, and K. Rajan. “Activity Planning for the Mars Exploration Rovers.” In: *ICAPS*. Vol. 2005. 2005, pp. 40–49.

[BJPM16] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy. *Site reliability engineering: How Google runs production systems*. " O’Reilly Media, Inc.", 2016.

[BK21] A. Bolu and Ö. Korçak. “Adaptive task planning for multi-robot smart warehouse”. In: *IEEE Access* 9 (2021), pp. 27346–27358.

[BKIM13] D. Bri, T. Kanda, T. Ikeda, and T. Miyashita. “Person tracking in large public spaces using 3-D range sensors”. In: *IEEE Transactions on Human-Machine Systems* 43.6 (2013), pp. 522–534.

[BKR07] E. Brunskill, T. Kollar, and N. Roy. “Topological mapping using spectral clustering and classification”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2007, pp. 3491–3496.

[BM92] P. Besl and N. D. McKay. “A method for registration of 3-D shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.

[BN14] P. Bovbel and G. Nejat. “Casper: An Assistive Kitchen Robot to Promote Aging in Place”. In: *Journal of Med Devices* 8.3 (July 2014).

[BNLT04] M. Bosse, P. Newman, J. Leonard, and S. Teller. “Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework”. In: *The International Journal of Robotics Research* 23.12 (2004), pp. 1113–1139.

[Bos22] Boston Dynamics Inc. *Boston Dynamics Spot SDK*. Online. Accessed: 28.09.2022. 2022. URL: `https://dev.bostondynamics.com`.

[BS08] K. Bernardin and R. Stiefelhagen. “Evaluating multiple object tracking performance: the clear mot metrics”. In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[BS16] I. Bogoslavskyi and C. Stachniss. “Fast range image-based segmentation of sparse 3D laser scans for online operation”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 163–169.

[BV16] J. Biswas and M. Veloso. “The 1,000-km Challenge: Insights and Quantitative and Qualitative Results”. In: *IEEE Intelligent Systems* 31.3 (2016), pp. 86–96.

[BZ10] M. Bosse and R. Zlot. “Place Recognition Using Regional Point Descriptors for 3D Mapping”. In: *Field and Service Robotics*. Vol. 62. Springer Tracts in Advanced Robotics. Springer-Verlag Berlin Heidelberg, 2010, pp. 195–204.

[CACB12] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona. “A linear approximation for graph-based simultaneous localization and mapping”. In: *Robotics: Science and Systems*. Vol. 7. 2012, pp. 41–48.

[CAÖ19] M. Colledanchise, D. Almeida, and P. Ögren. “Towards Blended Reactive Planning and Acting using Behavior Trees”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8839–8845.

[CCC+16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[CE13] N. Carlevaris-Bianco and R. M. Eustice. “Generic factor-based node marginalization and edge sparsification for pose-graph SLAM”. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 5748–5755.

[CH10] S.-Y. Chung and H.-P. Huang. “A Mobile Robot that Understands Pedestrian Spatial Behaviors”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 5861–5866.

[CLSF10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. “Brief: Binary robust independent elementary features”. In: *European Conference on Computer Vision*. Springer. 2010, pp. 778–792.

[CMHC17] E. Chinellato, K. V. Mardia, D. C. Hogg, and A. G. Cohn. “An incremental von Mises mixture framework for modelling human activity streaming data”. In: *Proceedings ITISE* 2017 (2017).

[CMÖ14] M. Colledanchise, A. Marzinotto, and P. Ögren. “Performance analysis of stochastic behavior trees”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 3265–3272.

[CN13] W. Churchill and P. Newman. “Experience-based navigation for long-term localisation”. In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1645–1661.

[CN21] M. Colledanchise and L. Natale. “On the implementation of behavior trees in robotics”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5929–5936.

[CÖ18] M. Colledanchise and P. Ögren. *Behavior Trees in Robotics and Al: An Introduction*. CRC Press, 2018.

[Con22] C. Contreras. *FedEx Shuts Down Roxo Same Day Delivery Service as Amazon Scales Back Scout Program*. Online. Accessed: 17.11.2022. 2022. URL: https://www.robotics247.com/article/fedex_shuts_down_robotic_same_day_delivery_service_amazon_scales_back_scout_program.

[Cou92] R. C. Coulter. *Implementation of the Pure Pursuit Path Tracking Algorithm*. Tech. rep. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[CSS87] P. Cheeseman, R. Smith, and M. Self. “A stochastic map for uncertain spatial relationships”. In: *4th international symposium on robotic research*. MIT Press Cambridge. 1987, pp. 467–474.

[CTL+15] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng, and Y. Chong. “Sensor technologies and simultaneous localization and mapping (SLAM)”. In: *Procedia Computer Science* 76 (2015), pp. 174–179.

[CTW+08] K.-H. Chiang, S.-H. Tseng, Y.-H. Wu, G.-H. Li, C.-P. Lam, and L.-C. Fu. “Multisensor-based Outdoor Tour Guide Robot NTU-I”. In: *SICE Annual Conference* (2008), pp. 1425–1430.

[Dav10] D. Davendra. *Traveling Salesman Problem: Theory and Applications*. InTech, 2010.

[DB06] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110.

[DBH19] F. D. Duchetto, P. Baxter, and M. Hanheide. “Lindsey the Tour Guide Robot - Usage Patterns in a Museum Long-Term Deployment”. In: *28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2019, pp. 1–8.

[DCD+18] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena. “SegMap: 3d segment mapping using data-driven descriptors”. In: *arXiv preprint* (2018).

[DCD11] F. Dayoub, G. Cielniak, and T. Duckett. “Long-term experiments with an adaptive spherical view representation for navigation in changing environments”. In: *Robotics and Autonomous Systems* 59.5 (2011), pp. 285–295.

[Dij59] E. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[DR93] A. Dutt and V. Rokhlin. “Fast Fourier transforms for nonequispaced data”. In: *SIAM Journal on Scientific computing* 14.6 (1993), pp. 1368–1393.

[Duv14] D. Duvenaud. “Automatic model construction with Gaussian processes”. PhD thesis. University of Cambridge, 2014.

[EHS+14] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. “3-D Mapping With an RGB-D Camera”. In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 177–187.

[EKJ07] S. Ekvall, D. Kragic, and P. Jensfelt. “Object detection and mapping for service robot tasks”. In: *Robotica* 25.2 (2007), pp. 175–187.

[ENT05] C. Estrada, J. Neira, and J. D. Tardós. “Hierarchical SLAM: Real-time accurate mapping of large environments”. In: *IEEE Transactions on Robotics* 21.4 (2005), pp. 588–596.

[ESC15] J. Engel, J. Stückler, and D. Cremers. “Large-scale direct SLAM with stereo cameras”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 1935–1942.

[ESNO20] S. F. G. Ehlers, M. Stuede, K. Nuelle, and T. Ortmaier. “Map Management Approach for SLAM in Large-Scale Indoor and Outdoor Areas”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9652–9658.

[ESR09] D. Ellis, E. Sommerlade, and I. Reid. “Modelling pedestrian trajectory patterns with Gaussian processes”. In: *IEEE ICCV Workshops* (2009), pp. 1229–1234.

[FB81] M. A. Fischler and R. C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[FBT97] D. Fox, W. Burgard, and S. Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[FDO+20] L. Fuentes-Moraleda, P. Diaz-Perez, A. Orea-Giner, A. Munoz-Mazon, and T. Villace-Molinero. “Interaction between hotel service robots and humans: A hotel-specific Service Robot Acceptance Model (sRAM)”. In: *Tourism Management Perspectives* 36 (2020), p. 100751.

[Fra18] I. A. Franch. “Human-help in automated planning under uncertainty”. PhD thesis. Instituto de Matematica e Estatistica, Universidade de Sao Paulo, 2018.

[FRR15] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha. “Visual simultaneous localization and mapping: A survey”. In: *Artificial Intelligence Review* 43.1 (2015), pp. 55–81.

[FS97] Y. Freund and R. E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[FT07] A. Foka and P. Trahanias. “Real-time hierarchical POMDPs for autonomous robot navigation”. In: *Robotics and Autonomous Systems* 55.7 (2007), pp. 561–571.

[FZG+16] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. “SVO: Semidirect visual odometry for monocular and multicamera systems”. In: *IEEE Transactions on Robotics* 33.2 (2016), pp. 249–265.

[GBS+09] H.-M. Gross, H. Boehme, C. Schroeter, S. Mueller, A. Koenig, E. Einhorn, C. Martin, M. Merten, and A. Bley. “TOOMAS: Interactive Shopping Guide Robots in Everyday Use - Final Implementation and Experiences from Long-term Field Trials”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2009), pp. 2005–2012.

[GCRN09] K. Granström, J. Callmer, F. Ramos, and J. Nieto. “Learning to detect loop closure from range data”. In: *2009 IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 15–22.

[GFFM18] D. Giovanelli, E. Farella, D. Fontanelli, and D. Macii. “Bluetooth-based indoor positioning through ToF and RSSI data fusion”. In: *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2018, pp. 1–8.

[GK20] S. Gupta and V. Kapoor. *Fundamentals of mathematical statistics*. Sultan Chand & Sons, 2020.

[GKSB10] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. “A tutorial on graph-based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[GNT04] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Elsevier, 2004.

[GNT16] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[GO15] E. Garcia-Fidalgo and A. Ortiz. “Vision-based topological mapping and localization methods: A survey”. In: *Robotics and Autonomous Systems* 64 (2015), pp. 1–20.

[GOLR16] W. N. Greene, K. Ok, P. Lommel, and N. Roy. “Multi-level mapping: Real-time dense monocular slam”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 833–840.

[GP10] A. Guez and J. Pineau. “Multi-tasking SLAM”. In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 377–384.

[GS10] K. Granström and T. B. Schön. “Learning to close the loop from 3D point clouds”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 2089–2095.

[GSB07] G. Grisetti, C. Stachniss, and W. Burgard. “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters”. In: *IEEE Transactions on Robotics* 23.1 (2007), pp. 34–46.

[GVD18] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake. “Gaussian processes autonomous mapping and exploration for range-sensing mobile robots”. In: *Autonomous Robots* 42.2 (2018), pp. 273–290.

[GWN+17] A. G. de G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. “GPflow: A Gaussian Process Library using TensorFlow”. In: *Journal of Machine Learning Research* 18.40 (2017), pp. 1–6.

[Har87] D. Harel. “Statecharts: A visual formalism for complex systems”. In: *Science of computer programming* 8.3 (1987), pp. 231–274.

[HBJ+17] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, et al. “The STRANDS Project: Long-Term Autonomy in Everyday Environments”. In: *IEEE Robotics & Automation Magazine* 24.3 (2017), pp. 146–156.

[HDFF16] C.-h. Hua, L.-h. Dou, H. Fang, and H. Fu. “A Novel Algorithm for SLAM in Dynamic Environments Using Landscape Theory of Aggregation”. In: *Journal of Central South University* 23.10 (2016), pp. 2587–2594.

[HHK17] M. Hanheide, D. Hebesberger, and T. Krajník. “The When, Where, and How: An Adaptive Robotic Info-Terminal for Care Home Residents - A Long-Term Study”. In: *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2017, pp. 341–349.

[HHS+20] Z. Huang, A. Hasan, K. Shin, R. Li, and K. Driggs-Campbell. “Long-term pedestrian trajectory prediction using mutable intention filter and warp LSTM”. In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 542–549.

[HIT+15] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. “Registration with the point cloud library: A modular framework for aligning in 3-D”. In: *IEEE Robotics & Automation Magazine* 22.4 (2015), pp. 110–124.

[HKG+16] D. Hebesberger, T. Koertner, C. Gisinger, J. Pripfl, and C. Dondrup. “Lessons learned from the deployment of a long-term autonomous robot as companion in physical therapy for older adults with dementia a mixed methods study”. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2016, pp. 27–34.

[HKM+21] J. Holland, L. Kingston, C. McCarthy, E. Armstrong, P. ODwyer, F. Merz, and M. McConnell. “Service Robots in the Healthcare Sector”. In: *Robotics* 10.1 (2021), p. 47.

[HKRA16] W. Hess, D. Kohler, H. Rapp, and D. Andor. “Real-time loop closure in 2D LIDAR SLAM”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.

[HNR68] P. Hart, N. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[Hoo84] W. E. Hoover. “Algorithms for confidence circles and ellipses”. In: (1984).

[HSKV19] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze. “An Empirical Evaluation of Ten Depth Cameras: Bias, Precision, Lateral Noise, Different Lighting Conditions and Materials, and Multiple Sensor Setups in Indoor Environments”. In: *IEEE Robotics Automation Magazine* 26.1 (2019), pp. 67–77.

[HSLS21] T.-L. Habich, M. Stuede, M. Labbé, and S. Spindeldreier. “Have I been here before? Learning to Close the Loop with LiDAR Data in Graph-Based SLAM”. In: *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2021.

[HTS15] A. Harmat, M. Trentini, and I. Sharf. “Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments”. In: *Journal of Intelligent & Robotic Systems* 78.2 (2015), pp. 291–317.

[HZL19] B. Huang, J. Zhao, and J. Liu. “A survey of simultaneous localization and mapping”. In: *arXiv preprint arXiv:1909.05214* (2019).

[Ibe13] O. Ibe. *Markov processes for stochastic modeling*. Newnes, 2013.

[IG17] F. Ingrand and M. Ghallab. “Deliberation for autonomous robots: A survey”. In: *Artificial Intelligence* 247 (2017), pp. 10–44.

[ISS+20] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith. “A Survey of Behavior Trees in Robotics and AI”. In: *arXiv preprint arXiv:2005.05842* (2020).

[JH18] S. John and J. Hensman. “Large-scale Cox process inference using variational Fourier features”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2362–2370.

[JKFL13] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard. “Temporally scalable visual SLAM using a reduced pose graph”. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 54–61.

[Jon+18] M. de Jong et al. “Towards a Robust Interactive and Learning Social Robot”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 883–891.

[JWHK16] F. Jovan, J. Wyatt, N. Hawes, and T. Krajník. "A Poisson-Spectral Model for Modelling Temporal Patterns in Human Data Observed by a Robot". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4013–4018.

[KB09] K. Konolige and J. Bowman. "Towards lifelong visual maps". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2009, pp. 1156–1163.

[KB14] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[KCAK17] P. Kim, B. Coltin, O. Alexandrov, and H. J. Kim. "Robust visual localization in changing lighting conditions". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 5447–5452.

[KFC+14] T. Krajnik, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett. "Spectral analysis for long-term robotic mapping". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 3706–3711.

[KFSD17] T. Krajnik, J. P. Fentanes, J. M. Santos, and T. Duckett. "FreMEn: Frequency Map Enhancement for Long-Term Mobile Robot Autonomy in Changing Environments". In: *IEEE Transactions on Robotics* 33 (2017), pp. 964–977.

[KGK+10] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. "Efficient sparse pose adjustment for 2D mapping". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 22–29.

[KGS+11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 3607–3613.

[KHD+18] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník. "Artificial Intelligence for Long-Term Robot Autonomy: A Survey". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4023–4030.

[KJR+12] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. "iSAM2: Incremental smoothing and mapping using the Bayes tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[KK13] S. Kim and J. Kim. "Continuous occupancy maps using overlapping local Gaussian processes". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 4709–4714.

[KK18a] E. Khalastchi and M. Kalech. "On Fault Detection and Diagnosis in Robotic Systems". In: *ACM Computing Surveys (CSUR)* 51.1 (2018), p. 9.

[KK18b] G. Kim and A. Kim. “Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[KKM+15] T. Krajník, M. Kulich, L. Mudrová, R. Ambrus, and T. Duckett. “Where’s waldo at time t? using spatio-temporal models for mobile robot search”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2140–2146.

[KLE11] K. Kim, D. Lee, and I. Essa. “Gaussian process regression flow for analysis of motion trajectories”. In: *2011 International Conference on Computer Vision*. 2011, pp. 1164–1171.

[KMKI20] D. Karunarathne, Y. Morales, T. Kanda, and H. Ishiguro. “Understanding a public environment via continuous robot observations”. In: *Robotics and Autonomous Systems* 126 (2020), p. 103443.

[KMM11] K. Konolige, E. Marder-Eppstein, and B. Marthi. “Navigation in hybrid metric-topological maps”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 3041–3047.

[KMS+17] T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal. “Enabling Flow Awareness for Mobile Robots in Partially Observable Environments”. In: *IEEE RA-L* 2.2 (2017), pp. 1093–1100.

[KMW+11] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura. “Pedestrian recognition using high-definition LIDAR”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)* (2011), pp. 405–410.

[KPK19] G. Kim, B. Park, and A. Kim. “1-Day Learning, 1-Year Localization: Long-Term LiDAR Localization Using Scan Context Image”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1948–1955.

[KPR+15] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr, and D. Murray. “Very high frame rate volumetric integration of depth images on mobile devices”. In: *IEEE transactions on visualization and computer graphics* 21.11 (2015), pp. 1241–1250.

[KZI+21] M. U. Khan, S. A. A. Zaidi, A. Ishtiaq, S. U. R. Bukhari, S. Samer, and A. Farman. “A comparative survey of lidar-slam and lidar based sensor technologies”. In: *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*. 2021, pp. 1–8.

[LBLA16] T. Linder, S. Breuers, B. Leibe, and K. O. Arras. “On multi-modal people tracking from mobile platforms in very crowded and dynamic environments”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 5512–5519.

[LC04] H. Liu and G. M. Coghill. “A model-based approach to robot fault diagnosis”. In: *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer. 2004, pp. 137–150.

[LDA11] M. Luber, G. Diego Tipaldi, and K. O. Arras. “Place-dependent people tracking”. In: *The International Journal of Robotics Research* 30.3 (2011), pp. 280–293.

[LGA15] T. Linder, F. Girrbach, and K. O. Arras. “Towards a robust people tracking framework for service robots in crowded, dynamic environments”. In: *Assistance and Service Robotics Workshop (ASROB-15) at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2015.

[LGQ+18] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen. “Autonomous aerial navigation using monocular visual-inertial fusion”. In: *Journal of Field Robotics* 35.1 (2018), pp. 23–51.

[LKF+10] M. K. Lee, S. Kiesler, J. Forlizzi, S. Srinivasa, and P. Rybski. “Gracefully mitigating breakdowns in robotic services”. In: *5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2010), pp. 203–210.

[LM11] M. Labbé and F. Michaud. “Memory management for real-time appearance-based loop closure detection”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011, pp. 1271–1276.

[LM13] M. Labbé and F. Michaud. “Appearance-based loop closure detection for online large-scale and long-term operation”. In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745.

[LM18] M. Labbé and F. Michaud. “Long-term online multi-session graph-based SPLAM with memory management”. In: *Autonomous Robots* 42.6 (2018), pp. 1133–1150.

[LM19] M. Labbé and F. Michaud. “RTABMap as an opensource lidar and visual simultaneous localization and mapping library for largescale and longterm online operation”. In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446.

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[LMR+19] M. Luperto, J. Monroy, J. R. Ruiz-Sarmiento, F.-A. Moreno, N. Basilico, J. Gonzalez-Jimenez, and N. A. Borghese. “Towards Long-Term Deployment of a Mobile Robot for at-Home Ambient Assisted Living of the Elderly”. In: *European Conference on Mobile Robots (ECMR)* (2019), pp. 1–6.

[LOSC20] H. Liu, Y.-S. Ong, X. Shen, and J. Cai. “When Gaussian process meets big data: a review of scalable GPs”. English. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (Jan. 2020), pp. 4405–4423.

[Low04] D. G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[LPS+21] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy. “Reactive task and motion planning under temporal logic specifications”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 12618–12624.

[LSG19] J. Liebner, A. Scheidig, and H.-M. Gross. “Now I Need Help! Passing Doors and Using Elevators as an Assistance Requiring Robot”. In: *Social Robotics*. 2019, pp. 527–537.

[LZW+19] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. “L3-net: Towards learning based lidar localization for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6389–6398.

[MAA18] P. Moreno-Muñoz, A. Artés, and M. Alvarez. “Heterogeneous multi-output Gaussian process prediction”. In: *Advances in neural information processing systems* 31 (2018).

[Mac+98] D. J. MacKay et al. “Introduction to Gaussian processes”. In: *NATO ASI series F Computer and Systems Sciences* 168 (1998), pp. 133–166.

[MANL09] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal. “Appearance-based loop detection from 3D laser data using the normal distributions transform”. In: *2009 IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 23–28.

[MCD19] S. Molina, G. Cielniak, and T. Duckett. “Go with the Flow: Exploration and Mapping of Pedestrian Flow Patterns from Partial Observations”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)* (2019), pp. 9725–9731.

[MCD21] S. Molina, G. Cielniak, and T. Duckett. “Robotic exploration for learning human motion patterns”. In: *IEEE Transactions on Robotics* (2021).

[Mea55] G. H. Mealy. “A method for synthesizing sequential circuits”. In: *The Bell System Technical Journal* 34.5 (1955), pp. 1045–1079.

[Mel21] S. M. Mellado. “Exploration and Mapping of Spatio-Temporal Pedestrian FlowPatterns for Mobile Robots”. PhD thesis. 2021.

[MFB11] C. McManus, P. Furgale, and T. D. Barfoot. “Towards appearance-based methods for lidar sensors”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1930–1935.

[MGMR21] F. Martín, J. Ginés, V. Matellán, and F. J. Rodríguez. *PlanSys2: A Planning System Framework for ROS2*. 2021.

[MGS18] N. Mishraa, D. Goyal, and A. D. Sharma. “Issues in existing robotic service in restaurants and hotels”. In: *Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT)*. 2018, pp. 26–27.

[MHDL17] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4628–4635.

[MLH15] L. Mudrova, B. Lacerda, and N. Hawes. “An integrated control framework for long-term autonomy in mobile service robots”. In: *2015 European Conference on Mobile Robots (ECMR)*. 2015, pp. 1–6.

[MME+21] F. Martín, M. Morelli, H. Espinoza, F. J. G. Lera, and V. Matellán. “Optimized Execution of PDDL Plans using Behavior Trees”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 2021, pp. 1596–1598.

[MMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[MMWG11] W. Meeussen, E. Marder-Eppstein, K. Watts, and B. Gerkey. “Long Term Autonomy in Office Environments”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA), Workshop on Long-term Autonomy*. 2011.

[MMWG20] S. Macenski, F. Martín, R. White, and J. Ginés Clavero. “The Marathon 2: A Navigation System”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[MT17] R. Mur-Artal and J. D. Tardós. “ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. “FastSLAM: a factored solution to the simultaneous localization and mapping problem”. In: *Proceedings of the AAAI/IAAI Conference on Artificial Intelligence*. 2002.

[MUN15] C. McManus, B. Upcroft, and P. Newman. “Learning place-dependant features for long-term vision-based localisation”. In: *Autonomous Robots* 39.3 (2015), pp. 363–387.

[MW12] M. J. Milford and G. F. Wyeth. “SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights”. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 1643–1649.

[MYNU20] H. Minoura, R. Yonetani, M. Nishimura, and Y. Ushiku. “Crowd density forecasting by modeling patch-based dynamics”. In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 287–294.

[Nas20] S. Nassauer. *Walmart Scraps Plan to Have Robots Scan Shelves*. Wall Street Journal, Online. Accessed: 10.11.2022. 2020. URL: https://www.wsj.com/articles/walmart-shelves-plan-to-have-robots-scan-shelves-11604345341.

[NBS18] T. Naseer, W. Burgard, and C. Stachniss. "Robust visual localization across seasons". In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 289–302.

[NI14] L. Nardi and L. Iocchi. "Representation and execution of social plans through human-robot collaboration". In: *International Conference on Social Robotics*. Springer. 2014, pp. 266–275.

[NMH10] L. E. Navarro-Serment, C. Mertz, and M. Hebert. "Pedestrian Detection and Tracking Using Three-dimensional LADAR Data". In: *The International Journal of Robotics Research* 29.12 (2010), pp. 1516–1528.

[NN19] T. Nishio and M. Niitsuma. "Environmental Map Building to Describe Walking Dynamics for Determination of Spatial Feature of Walking Activity". In: *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*. 2019, pp. 2315–2320.

[OLT07] E. Olson, J. J. Leonard, and S. J. Teller. "Spatially-Adaptive Learning Rates for Online Incremental SLAM." In: *Robotics: Science and Systems*. Citeseer. 2007.

[OR12] S. T. O'Callaghan and F. T. Ramos. "Gaussian process occupancy maps". In: *The International Journal of Robotics Research* 31.1 (2012), pp. 42–62.

[OSAR11] S. T. O'Callaghan, S. P. N. Singh, A. Alempijevic, and F. T. Ramos. "Learning navigational maps by observing human motion patterns". In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 4333–4340.

[PFC+17] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. J. Berlles. "S-PTAM: Stereo parallel tracking and mapping". In: *Robotics and Autonomous Systems* 93 (2017), pp. 27–42.

[PFS14] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, monocular dense reconstruction in real time". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2609–2616.

[PHJ+17] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager. "CoSTAR: Instructing collaborative robots with behavior trees and vision". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 564–571.

[PKM+17] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras. "Kinodynamic motion planning on Gaussian mixture fields". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6176–6181.

[PMF+16] R. Pinillos, S. Marcos, R. Feliz, E. Zalama, and J. Gómez-García-Bermejo. “Long-term Assessment of a Service Robot in a Hotel Environment”. In: *Robotics and Autonomous Systems* 79 (2016), pp. 40–57.

[RBB09] R. B. Rusu, N. Blodow, and M. Beetz. “Fast point feature histograms (FPFH) for 3D registration”. In: *2009 IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 3212–3217.

[Reu21] Reuters. *SoftBank shrinks robotics business, stops Pepper production*. Online. Accessed: 10.11.2022. 2021. URL: https://tinyurl.com/2hwrdv8a.

[RF18] J. Redmon and A. Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[RMI13] A. Ranganathan, S. Matsumoto, and D. Ilstrup. “Towards illumination invariance for visual localization”. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 3791–3798.

[RMS15] T. Rohling, J. Mack, and D. Schulz. “A fast histogram-based similarity measure for detecting loop closures in 3-D LIDAR data”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 736–741.

[RNS+16] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart. “Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4304–4311.

[RO16] F. Ramos and L. Ott. “Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent”. In: *The International Journal of Robotics Research* 35.14 (2016), pp. 1717–1730.

[Rob18] M. Robotics. *An Important (And Difficult) Announcement*. Online. Accessed: 10.11.2022. 2018. URL: https://www.heykuri.com/blog/important_difficult_announcement/.

[RPH+20] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras. “Human motion trajectory prediction: A survey”. In: *The International Journal of Robotics Research* 39.8 (2020), pp. 895–935.

[RRH+20] A. Ravankar, A. A. Ravankar, Y. Hoshino, M. Watanabe, and Y. Kobayashi. “Safe mobile robot navigation in human-centered environments using a heat map-based path planner”. In: *Artificial Life and Robotics* 25.2 (2020), pp. 264–272.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571.

[RV12] S. Rosenthal and M. Veloso. “Mobile robot planning to seek help with spatially-situated tasks”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[RVD12] S. Rosenthal, M. Veloso, and A. K. Dey. “Is Someone in this Office Available to Help Me?” In: *Journal of Intelligent & Robotic Systems* 66.1-2 (Apr. 2012), pp. 205–221.

[SAL12] J. Saarinen, H. Andreasson, and A. J. Lilienthal. “Independent markov chain occupancy grid maps for representation of dynamic environment”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 3489–3495.

[SB05] C. Stachniss and W. Burgard. “Mobile robot mapping and localization in non-static environments”. In: *AAAI*. 2005, pp. 1324–1329.

[SC86] R. C. Smith and P. Cheeseman. “On the representation and estimation of spatial uncertainty”. In: *The International Journal of Robotics Research* 5.4 (1986), pp. 56–68.

[Sch19] T. Schleifer. *The once-hot robotics startup Anki is shutting down after raising more than 200 million*. Vox, Online. Accessed: 10.11.2022. 2019. URL: https://www.vox.com/2019/4/29/18522966/anki-robot-cozmo-staff-layoffs-robotics-toys-boris-sofman.

[SDF+18] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. “maplab: An open framework for research in visual-inertial mapping and localization”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1418–1425.

[Sec21] I. C. Secretary. *Robotics - Vocabulary*. Standard BS ISO 8373:2021. Geneva, CH: International Organization for Standardization, 2021.

[SEH18] H. Salimbeni, S. Eleftheriadis, and J. Hensman. “Natural gradients in practice: Non-conjugate variational inference in Gaussian process models”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 689–697.

[SHVL16] A. D. Saul, J. Hensman, A. Vehtari, and N. D. Lawrence. “Chained Gaussian Processes”. In: *Proceedings of Machine Learning Research* 51 (2016), pp. 1431–1440.

[SIN+22] J. Styrud, M. Iovino, M. Norrlöf, M. Björkman, and C. Smith. “Combining planning and learning of behavior trees for robotic assembly”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 11511–11517.

[SK19] A. Solin and M. Kok. “Know your boundaries: Constraining Gaussian processes by variational harmonic features”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2193–2202.

[SLPS21] M. Stuede, T. Lerche, M. A. Petersen, and S. Spindeldreier. “Behavior-Tree-Based Person Search for Symbiotic Autonomous Mobile Robot Tasks”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[SLT16] C. Stachniss, J. J. Leonard, and S. Thrun. “Simultaneous localization and mapping”. In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1153–1176.

[SMD12] H. Strasdat, J. M. Montiel, and A. J. Davison. “Visual SLAM: why filter?” In: *Image and Vision Computing* 30.2 (2012), pp. 65–77.

[SME+12] R. Stricker, S. Müller, E. Einhorn, C. Schröter, M. Volkhardt, K. Debes, and H.-M. Gross. “Interactive mobile robots guiding visitors in a university building”. In: *The 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2012, pp. 695–700.

[SNTO19] M. Stuede, K. Nuelle, S. Tappe, and T. Ortmaier. “Door opening and traversal with an industrial cartesian impedance controlled mobile robot”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 966–972.

[SOR16] R. Senanayake, S. O’Callaghan, and F. Ramos. “Predicting spatio-temporal propagation of seasonal influenza using variational Gaussian process regression”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[SR18] R. Senanayake and F. Ramos. “Directional Grid Maps: Modeling Multimodal Angular Uncertainty in Dynamic Environments”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3241–3248.

[SRGB11] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. “Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Ed. by N. M. Amato. Piscataway, NJ: IEEE, 2011, pp. 1249–1255.

[SS22] M. Stuede and M. Schappler. “Non-Parametric Modeling of Spatio-Temporal Human Activity Based on Mobile Robot Observations”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022.

[SSC90] R. Smith, M. Self, and P. Cheeseman. “Estimating uncertain spatial relationships in robotics”. In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.

[Sta21] Statista. *Robotics*. Online. Accessed: 10.11.2022. 2021. URL: https://www.statista.com/study/64326/robotics/.

[Sta22] Statista. *Robotics - Worldwide*. Online. Accessed: 10.11.2022. 2022. URL: https://www.statista.com/outlook/tmo/robotics/worldwide.

[Stu22] M. Stuede. *Sobi: Open source hardware and software documentation*. 2022. URL: https://marvinstuede.github.io/Sobi/.

[SWSS21] M. Stuede, K. Westermann, M. Schappler, and S. Spindeldreier. “Sobi: An Interactive Social Service Robot for Long-Term Autonomy in Open Environments”. In: *10th European Conference on Mobile Robotics (ECMR)*. 2021.

[SWTO19] M. Stuede, J. Wilkening, S. Tappe, and T. Ortmaier. “Voice recognition and processing interface for an interactive guide robot in an university scenario”. In: *2019 19th International Conference on Control, Automation and Systems (ICCAS)*. 2019, pp. 1238–1242.

[SYM+18] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett. “3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5942–5948.

[SZDZ17] H. Su, J. Zhu, Y. Dong, and B. Zhang. “Forecast the Plausible Paths in Crowd Scenes”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 2772–2778.

[TA11] G. D. Tipaldi and K. O. Arras. “I want my coffee hot! Learning to find people under spatio-temporal constraints”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 1217–1222.

[Thr+02] S. Thrun et al. “Robotic mapping: A survey”. In: *Exploring artificial intelligence in the new millennium* 1.1-35 (2002), p. 1.

[Tit09] M. Titsias. “Variational learning of inducing variables in sparse Gaussian processes”. In: *Artificial intelligence and statistics*. PMLR. 2009, pp. 567–574.

[TKL+14] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. “Asking for Help Using Inverse Semantics”. In: *Proceedings of Robotics: Science and Systems*. Berkeley, USA, 2014.

[TL08] M. K. C. Tay and C. Laugier. “Modelling smooth paths using gaussian processes”. In: *Field and Service Robotics*. Springer. 2008, pp. 381–390.

[TMB13] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard. “Lifelong localization in changing environments”. In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1662–1678.

[TR18] A. Tompkins and F. Ramos. “Fourier Feature Approximations for Periodic Kernels in Time-Series Modelling”. In: *AAAI Conference on Artificial Intelligence*. 2018.

[Tri+16] R. Triebel et al. “SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports”. In: *Springer Tracts in Advanced Robotics* (2016), pp. 607–622.

[TSH+18] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. “Semantic match consistency for long-term visual localization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 383–399.

[TVNB17] T. T. Tran, T. Vaquero, G. Nejat, and J. C. Beck. “Robots in Retirement Homes: Applying Off-the-Shelf Planning and Scheduling to a Team of Assistive Robots”. In: *Journal of Artificial Intelligence Research* 58 (2017), pp. 523–590.

[UL18] M. A. Uy and G. H. Lee. “Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4470–4479.

[Van19] B. Vanderborght. “Robotic Dreams, Robotic Realities [From the Editor’s Desk]”. In: *IEEE Robotics & Automation Magazine* 26.1 (2019), pp. 4–5.

[VBR+22] T. Vintr, J. Blaha, M. Rektoris, J. Ulrich, T. Rouek, G. Broughton, Z. Yan, and T. Krajník. “Toward benchmarking of long-term spatio-temporal maps of pedestrian flows for human-aware navigation”. In: *Frontiers in robotics and AI* 9 (2022).

[VCP21] M. Valdez, M. Cook, and S. Potter. “Humans and robots coping with crisis–Starship, Covid-19 and urban robotics in an unpredictable world”. In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2021, pp. 2596–2601.

[Vec23] Vecteezy. *Vecteezy.com*. Online. Accessed: 06.01.2023. 2023. URL: `https://www.vecteezy.com`.

[VG13] M. Volkhardt and H.-M. Gross. “Finding people in home environments with a mobile robot”. In: *European Conference on Mobile Robots (ECMR)*. 2013, pp. 282–287.

[VKK22] F. Verdoja, T. P. Kucner, and V. Kyrki. *Generating people flow from architecture of real unseen environments*. 2022.

[VKS08] K. Van Laerhoven, D. Kilian, and B. Schiele. “Using rhythm awareness in long-term activity recognition”. In: *2008 12th IEEE International Symposium on Wearable Computers*. 2008, pp. 63–66.

[VL10] C. Valgren and A. J. Lilienthal. “SIFT, SURF & seasons: Appearance-based long-term localization in outdoor environments”. In: *Robotics and Autonomous Systems* 58.2 (2010), pp. 149–156.

[VMS+19] T. Vintr, S. Molina, R. Senanayake, G. Broughton, Z. Yan, J. Ulrich, T. P. Kucner, C. S. Swaminathan, F. Majer, M. Stachová, et al. “Time-varying pedestrian flow models for service robots”. In: *European Conference on Mobile Robots (ECMR)*. 2019, pp. 1–7.

[VYDK19] T. Vintr, Z. Yan, T. Duckett, and T. Krajník. “Spatio-temporal representation for long-term anticipation of human presence in service robotics”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2620–2626.

[VYE+20] T. Vintr, Z. Yan, K. Eyisoy, F. Kubi, J. Blaha, J. Ulrich, C. S. Swaminathan, S. Molina, T. P. Kucner, M. Magnusson, et al. “Natural Criteria for Comparison of Pedestrian Flow Forecasting Models”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11197–11204.

[WAJF14] Z. Wang, R. Ambrus, P. Jensfelt, and J. Folkesson. “Modeling motion patterns of dynamic objects by IOHMM”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 1832–1838.

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[WC18] S. Wang and H. I. Christensen. “TritonBot: First Lessons Learned from Deployment of a Long-Term Autonomy Tour Guide Robot”. In: *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2018, pp. 158–165.

[WIT+10] A. Weiss, J. Igelsböck, M. Tscheligi, A. Bauer, K. Kühnlenz, D. Wollherr, and M. Buss. “Robots asking for directions - The willingness of passers-by to support robots”. In: *5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2010, pp. 23–30.

[WN17] D. Withers and P. Newman. “Modelling scene change for large-scale long term laser localisation”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6233–6239.

[WS00] C. Williams and M. Seeger. “Using the Nyström method to speed up kernel machines”. In: *Advances in neural information processing systems* 13 (2000).

[WSB08] K. M. Wurm, C. Stachniss, and W. Burgard. “Coordinated multi-robot exploration using a segmentation of the environment”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 1160–1165.

[XYZ17] X. Xiang, C. Yu, and Q. Zhang. “On intelligent risk analysis and critical decision of underwater robotic vehicle”. In: *Ocean Engineering* 140 (2017), pp. 453–465.

[YDB17] Z. Yan, T. Duckett, and N. Bellotto. “Online learning for human classification in 3D LiDAR-based tracking”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017).

[YSWC20] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers. “D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1281–1292.

[ZLK+18] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. “Places: A 10 Million Image Database for Scene Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1452–1464.

[ZM15] Z. Zhou and D. S. Matteson. “Predicting ambulance demand: A spatio-temporal kernel approach”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 2297–2303.

[ZS14] J. Zhang and S. Singh. “LOAM: Lidar Odometry and Mapping in Real-time”. In: *Robotics: Science and Systems X* (July 2014).

[ZWZ+18] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang. “Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1001–1010.

Related Supervised Student Theses

[Ben21] R. Ben Abdallah. “Evaluation und Fehlermodellierung bild- und tiefenmessbasierter Detektoren zum Personentracking mobiler Robotersysteme [Evaluation and error modeling of image- and depth-based detectors for real-time person tracking of mobile robotic systems]”. (Supervisor: M. Stüde). Project thesis. Hannover, Germany: Institute of Mechatronic Systems, Leibniz University Hannover, 2021.

[Har20] D. Harmel. “Entwicklung eines robusten Andockprozesses an eine passive Ladestation für einen autonomen mobilen Roboter [Robust Indoor Auto-Docking to Passive Charging Station for an Autonomous Mobile Robot]”. (Supervisor: M. Stüde). Project thesis. Hannover, Germany: Institute of Mechatronic Systems, Leibniz University Hannover, 2020.

[Pet19] A. Petersen. “Autonome Navigation und Führung durch einen mobilen Serviceroboter unter Berücksichtigung sozialer Aspekte [Autonomous social navigation and guidance by a mobile service robot]”. (Supervisor: M. Stüde). Master’s thesis. Hannover, Germany: Institute of Mechatronic Systems, Leibniz University Hannover, 2019.

[Wes20] K. Westermann. “Entwicklung einer Ablaufsteuerung für die Benutzerschnittstelle eines humanoiden Serviceroboters [Development of a sequence control for the user interface of a humanoid service robot]”. (Supervisor: M. Stüde). Project thesis. Hannover, Germany: Institute of Mechatronic Systems, Leibniz University Hannover, 2020.

[Wes21] K. Westermann. “Gaussprozessbasierte raum-zeitliche Prädiktion von Personenaufkommen in grossflächigen Umgebungen [Gaussian-process-based spatio-temporal prediction of the number of people in large-area environments]”. (Supervisor: M. Stüde). Master’s thesis. Hannover, Germany: Institute of Mechatronic Systems, Leibniz University Hannover, 2021.

Curriculum Vitae

	Personal
Name	Marvin Stüde
Nationality	German
	Work Experience
since Apr/23	Senior Robotics Engineer at ZF CV Systems Hannover GmbH
Mar/18 - Mar/23	Research Associate at the Institute of Mechatronic Systems, Leibniz University Hannover
	Academic Education
Oct/15 - Feb/18	Mechatronics, M.Sc., Leibniz University Hannover, Specialization: Automation & Industrial Robotics, Service Robots & Autonomous Systems, graduated 26th February 2018 (grade 1,2) with award by the Dr. Jürgen and Irmgard Ulderup Foundation
Jan/17 - Jun/17	Study Abroad at the University of Auckland, Auckland, New Zealand
Oct/11 - Sep/15	Mechatronics, B.Sc., Leibniz University Hannover
	Student Experiences and Internships
Apr/16 - Jan/17	Student Assistant Institute of Mechatronic Systems, Hannover
Oct/14 - Apr/15	Internship Quality Management and Methods, Bosch Rexroth AG, Xi'an, China
Oct/12 - Oct/13	Student Assistant Institute of Algebraic Geometry, Hannover
Mar/12 - Apr/12	Internship Production Department, Ardagh Group, Cuxhaven
	School Education
Aug/08 - Jul/11	Gymnasium, Berufsbildende Schulen Cuxhaven, graduated 23rd June 2011: Abitur (grade 1,4)
Aug/04 - Jul/08	Gymnasium, Lichtenberg Gymnasium Cuxhaven,
Aug/02 - Jul/04	Orientierungsstufe, Geschwister-Scholl-Schule, Altenwalde
Aug/98 - Jul/02	Grundschule, Franzenburger Schule, Altenwalde