HTBLuVA Wiener Neustadt
Höhere Lehranstalt für Informatik

DIPLOMARBEIT

Autonomous universal Mapping and Navigation

Ausgeführt im Schuljahr 2020/21 von:

Themengebiet 1 Lukas Leskovar	5BHIF
Themengebiet 2 Fabian Kleinrad	5BHIF

Betreuer / Betreuerin:

MMag. Dr. Michael Stifter

Wiener Neustadt, am November 25, 2020/21

Abgabevermerk: Übernommen von:

Chapter 1

Eidestattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegeben Quellen und Hilfsmittel verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Wiener Neustadt am November 25, 2020/21

Verfasser / Verfasserinnen:

Lukas Leskovar Fabian Kleinrad

Contents

Chapter 2

Acknowledgement

The authors would like to thank ...

Chapter 3

Kurzfassung

asdf

Chapter 4

Abstract

asdf

Chapter 5

Introduction

Author: Lukas Leskovar

5.1 The Evolution of Robotics

Robotic research has always utilized concepts, processes, and methods of different scientific disciplines such as physics, mathematics, and biology to improve application and aid human needs. Because of this industrial, medical and even agricultural sectors have used technologies and products developed by researchers to improve workflows and alleviate employees from performing exhausting tasks. This relationship ranges back to the early ages of information technology in the 1950s and 1960s in which many developments on production robots and Artificial Intelligence (AI) have been made. Between 1970 and 1990 the public interest in automation and AI has decreased forcing the industry into the so-called AI winter. Despite this recession, research has been continued and the building blocks for another robot boom during the 1990s have been set. Since then the usage of robotic applications has broadened and the industry has proven itself to be a vital aspect of today's economy.

5.2 Robots with human interaction

Nowadays the utilization of robots in workplaces has broadened to almost every branch and is accepted by employees and workers. In countries like Japan, robots are no longer seen as a threat to jobs. Industrial robots are no longer used as simple construction tools, their safety and accuracy have improved so that collaborative robots (Cobots) are capable of working in close cooperation with humans. Surgery robots used in the medical sector not only allow for much more accurate procedures but also enable remote specialists to work on patients without having to be in the same hospital. In developed countries, educational robots are used at school or at home to teach children topics in a playful and interesting way.

[1] Malone, *George Devol: A Life Devoted to Invention, and Robots*

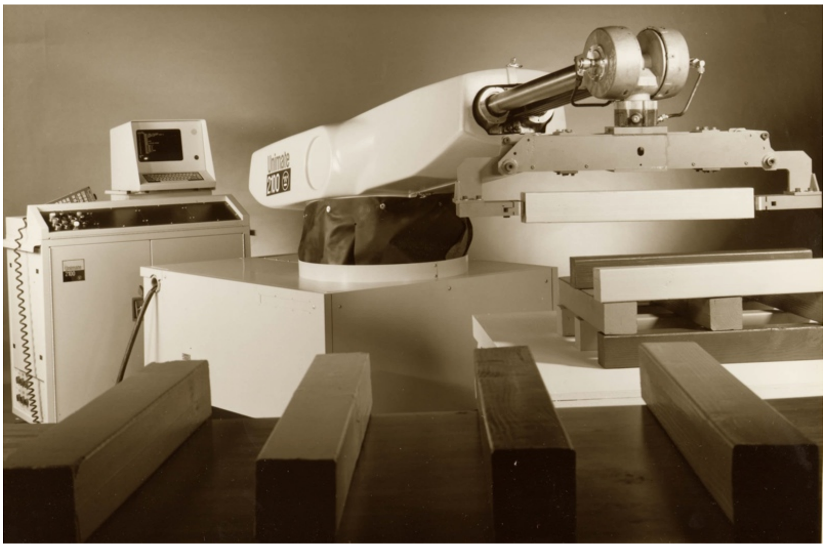

Figure 5.1: Picture of the first industrial robot. The Unimate developed by George Devol and Joseph Engelbert in 1961 was first used for hot die-casting and welding applications.[1]

5.3 Robots in hazardous environments

Robots do not only serve a purpose in a close-to-user work environment, they also ensure human safety by performing dangerous tasks in unsafe surroundings. Remotely controlled robots or drones can be used for inspecting mine shafts, collapsed buildings, pipelines, or overhead power poles. Other applications of such robots are bomb or mine defusion, fire extinction, or avalanche rescue.

5.4 Autonomous robots

Implementing an autonomous robot system is an intricate task that proposes many challenging problems for research or development teams. Autonomy requires a system to continuously work in a dynamic environment without external controlling inputs and utilize perceived information about its surroundings to adapt to environmental change.[2] Despite their complexity in development autonomous systems, mobile or stationary, immensely facilitate the execution of a job for the human user. Such robots can be used to navigate and organize warehouses, constructing parts in an assembly line, or map large areas for comprehensive calculations.

[2]Bekey, *Autonomous robots: from biological inspiration to implementation and control*, Pages 1-2.

Figure 5.2: The Emesent Hovermap mapping a dangerous area inside a mineshaft.[5]

5.5 3D Mapping

5.6 Autonomous 3D Mapping

While 3D Mapping is a well-established and growing economy most applications require human interaction at some point during the mapping process. Products such as the Emesent Hovermap[3] or Exyn Aero[4] utilize the spatial flexibility of drones and high-end light detection and ranging (LiDAR) Sensors to facilitate autonomous mapping in GPS-denied areas without being within sight of the drone pilot. These solutions are used to autonomously map building or explore hazardous environments such as mineshafts in a human-safe manner as seen in Fig. 5.2.

5.7 Goals

The project associated with this thesis aims to implement a 3D Mapping system similar to the aforementioned solutions with limited financial, personnel as well as temporal resources. This goal forces the project team to primarily maintain an open-source approach during development.

The goal of this thesis is to document the challenges encountered and experiences gained by the project team during development to demonstrate how highly sophisticated industrial problems can be solved in a low-budget fashion.

[3] Emesent Hovermap, *Autonomy Level 2 for Emesent Hovermap*.
[4] Exyn Aero, *Exyn Aero - Aerial Mapping Drone*.
[5] Emesent, *Emesent_Hovermap.JPG (JPEG Image, 2018 × 910 pixels)*

5.8 Requirements

In order to declare the project as successful the following criteria have to be implemented:

- The system is capable of generating a 3D Point-cloud of its surroundings
- All required hardware (e.g. sensors, computing boards, etc.) is mounted onto a drone-platform
- All calculations concerning the map-generation are run on a external server in direct communication with the drone
- The drone utilizes the Point-cloud to orientate in its surroundings and navigate one or multiple waypoints
- The drone is capable of avoiding obstacles as it is moving through a unknown environment

The following criteria can be implemented but have no direct correlation to the success of the project:

- A Web-App facilitating the usage of the system and enabling the user to create waypoints, monitor the drone and mapping algorithm as well as evaluate the Point-cloud
- The system is replicated within the gazebo simulator to simplify future development without direct access to its hardware

Chapter 6

Study of Literature

Author:

Chapter 7

System Architecture

Author: Lukas Leskovar
This chapter aims to provide a thorough overview of Autumns system architecture by describing its construction as well as logical and computational structure. To this end, the difficulties faced during development and decisions made that affected the overall project are described.

7.1 First Prototype

The first version of the Autumn drone consisted of three main components:

- A DJI Matrice 100 functioning as the base of the system powering all external components.
- To perform 3D mapping a Stereolabs ZED 1 was mounted onto the lower front part of the drone
- As the main component used to compute 3D mapping and control the drone a NVIDIA Jetson TX2 was mounted onto the drones expansion bay.

This prototype quickly demonstrated which part of the system needed to be improved where as the main issues were the lack of computational power and usage of non optimal hardware. A detailed explanation as well as a multitude of solutions to this problem are discussed in the following sections.

7.2 Processing and power management

In the past decades research has made vast improvements concerning the performance of processors whether they are used as stand-alone microprocessors, microcontrollers, embedded processors or digital signal processors. However these improvements come at the cost of higher power requirements. This trade-off is a major concern in many robotic applications that are powered by batteries or are restricted to low power inputs. Since Autumn is powered only by a drone battery this issue is groundbreaking during the development of this diploma thesis.

The central component of the system is a NVIDIA Jetson TX2 board equipped with a 2GHz NVIDIA Denver2 dual-core and a 2GHz Arm Cortex-A57 quad-core processor.[1]
Using NVIDIA tegrastats[2] the average power consumption of the system at an idle state was measured at 2.7W. However while performing non optimized 3D mapping and navigation algorithms (Chapter 9) with both processors fully utilized and Max-N power mode activated the average power consumption reached 7.9W.[3] Operating the system at such high stress does not only quickly drain the drones battery but also impairs the quality of the resulting 3D map as well as operating the drone.

7.3 Solving processing limitations

When dealing with load heavy computations one way to solve quality issues is to use higher performance hardware, however due to aforementioned power constraints and drone payload requirements this approach is not suitable for this diploma thesis. Another possible solution is to lower the computational load of the processors therefore improving result quality and lowering power consumption. This approach was tested in the following two forms:

- Distributing high power computations to a remote host. This approach lowered the amount of computation on the drone but required to use a high performance computer on site to wirelessly communicate with the drone. Furthermore the wifi-range became another limiting factor since with increasing range the latency increased thus impairing the result quality again.
- The second approach was to disable visual odometry using feature extraction and pattern matching as the most complex part of the algorithm and providing odometry using a much more performant visual inertial odometry algorithm. To this end the drone was equipped with a Stereolabs ZED 2i stereo-camera which is benchmarked against other sensors in section **??**

7.4 Final Product

Following the aforementioned approach the main causes of performance issues where eliminated by replacing non optimized hardware components and distributing non-critical computations such as user interaction or path planning to a remote host as impairing these aspects of the system by latency would not pose as a great problem. With these adjustments the Autumn drones final version consisted of the following altered components:

- As already mentioned the Stereolabs ZED 1 was replaced by its successor the Stereolabs ZED 2i. With its additional sensors and out of the box sensor fusion available it greatly reduced the amount of computations performed on the NVIDIA Jetson TX2.

[1] NVIDIA Corporation, *NVIDIA Jetson Hardware Page*.
[2] NVIDIA Corporation, *tegrastats Utility*.
[3] NVIDIA Corporation, *Power Management for Jetson TX2 Series Devices - Supported Modes and Power Efficiency*.

Figure 7.1: The Autumn drone with the NVIDIA Jetson TX2 on top as well as the Stereolabs ZED 2i mounted onto the drones Gimbal mounting plate using a custom 3D printed frame.

- In order to establish a connection to a remote host and perform computations the Dual Band 2.4GHz and 5GHz Antennas of the NVIDIA Jetson TX2 where utilized.
- A laptop serving as the remote host was added to the system performing non-critical computations.

Chapter 8

Robot Operating System

Author: Lukas Leskovar
This chapters objective is to describe the basic concepts of the Robot Operating System (ROS) utilized by Autumn. The ROS despite its name is a meta-operating system or middleware providing the utility and services often found in robotics frameworks. It enables the composition of distributed systems by utilizing publisher-subscriber communication between different programs of such systems. Furthermore ROS provides a comprehensive set of tools enabling the compilation, operation as well as testing, visualization and debugging of robotic systems. With its vast amount of libraries and huge open-source community providing useful functionality ROS facilitates the development of robotic applications without having to reimplement standardized technology.[1]

8.1 Conceptual Overview

The Robot Operating System can be divided into three conceptual levels each contributing a integral part to the utility of ROS. These different levels are described in the following sections.[2]

File System

The File System Level mainly provides constraints and best practices for creating and structuring packages and their components. ROS provides appropriate tools to facilitate filesystem operations with and within packages.

Computational Graph

The Computational Graph provides crucial functionality to ROS as it refers to the peer-to-peer mesh network of processes (nodes) each providing data to be utilized within the graph by publishing and subscribing to topics. The concepts and technologies powering the computational graph are described in later in this chapter.

[1] Gerkey, *Definition - ROS Answers*.
[2] Open Source Robotics Foundation, *Concepts - ROS Wiki*.

Community

The Community preserves the usability of ROS as new and useful packages and tools are created as well as existing functionality is being maintained.

8.2 Naming

To aid the organization of programs, processes as well as resources ROS provides two naming schemes that are described in the following sections.[3]

Graph Resource Names

Graph Resource Names utilize a hierarchical structure to organize nodes, services, topics or anything else within the computational graph. ROS defines four different types of names:

Package Resource Names

Package Resource Names aim to facilitate the search process of resources at File System Level. These names usually consist of the packages name as well as the path to the desired resource within the package.

8.3 Packages

Software in ROS is organized in packages containing nodes, libraries or any other piece of software providing functionality.
Since packages are the atomic unit of build and release they aim to be a slim as possible my implementing only a limited set of features. In other words packages should be implemented to provide minimal usability without being too large-scaled.[4] This means that each package is developed to work together with other packages to deliver utility as a connected system.
At file-system level packages simply refer to directories. While most subfolders and files within a package depend on its purpose, every package has to contain a package.xml and a CMakeLists.txt providing meta and build information. Packages can be build by utilizing rosbuild or catkin.[5]

Metapackages

Metapackages are specialized packages only containing a package.xml that logically links multiple related packages.[6] They can be used to conveniently install a group of packages simultaneously.

[3] Open Source Robotics Foundation, *Concepts - ROS Wiki*.
[4] Open Source Robotics Foundation, *Packages - ROS Wiki*.
[5] Open Source Robotics Foundation, *Building Packages - ROS Wiki*.
[6] Open Source Robotics Foundation, *Metapackages - ROS Wiki*.

8.4 Nodes

The goal of ROS is to promote code reusability and decoupling of functionality to aid the versatility and usability of the system. Following this guideline every robotic system utilizing ROS consists of a fine-grained graph of processes called nodes. Each node provides computation on a single feature utilizing a ROS client library to communicate with others over a mesh-like peer-to-peer network.[7]

Exemplary for such as system would be one node running a LiDAR sensor, one responsible for localization, one performing motion planning, one controlling motor drivers and motors as well as one node running the robots main control loop.

This architecture allows for a much more fault safe and less complex applications in comparison to monolithic systems.[8] This means that development and debugging are facilitated since errors can be contained within a singular slim node rather than a larger program.

Each node has a node type consisting of the package name it is located and as well as the nodes executable.

8.5 Communication

8.5.1 Messages

Messages are the medium of communication used in topics or services to transport data between nodes.

Message Description

A message is a simple data structure consisting of multiple type fields. These fields can be primitives, arrays, custom types as well as other message types.[9]

The message description language can be used to structure custom messages in *.msg* files contained in the *msg* directory of a package.

Message Types

Message types refer to package resource names consisting of the packages name as well as the name of the messages *.msg* file.

8.5.2 Topics

The core component of communication in ROS are topics. They are unidirectional message streams enabling data transmission by utilizing the publisher-subscriber model Furthermore the decoupling of functionality is facilitated by anonymously connecting nodes as producer and consumer of data. This means neither publisher nor subscriber of the topic need to know each other. While ROS does not limit the amount of publishers and subscribers connected

[7] Open Source Robotics Foundation, *Nodes - ROS Wiki*.
[8] Stephens, *Beginning Software Engineering*, Page 94.
[9] Open Source Robotics Foundation, *Messages - ROS Wiki*.

to a topic, it strictly enforces the usage of the exact message type specified for the topics communication to work properly.

8.5.3 Services

The communication architecture in ROS utilizing the publisher-subscriber model is advantageous in most use-cases, however most distributed systems require remote procedure calls (RPC) which are not supported by default.
With RPCs a client sends a request to a server specifying the procedure to be called and its parameters. While the server executes the procedure the client awaits a reply. Once the procedures results are computed and sent to the client its workflow can be resumed.[10]
Services enable communication over RPC by defining a pair of messages, one for requests and one for replies. Such service can then be attached to a node and called by a client using the service name.[11]

8.6 Master

One of the most important components of ROS is the Master. It tracks publishers and subscribers of topics as well as services and provides registration as well as name resolution to nodes. This means whenever a node wants to publish or subscribe to a specific topic or service it contacts the master first using XML-RPC. When a topic has at leat one subscriber and publisher the Master negotiates between the nodes so a peer-to-peer connection can be established using a Slave API provided by the nodes XML-RPC Server.[12] A simplified version of this procedure can be seen in Fig. 8.1
Besides registration and name resolution, the ROS Master also provides a Parameter Server used for globally storing static system parameters.[13]

8.7 Transform Library

A complex robotic system consists of multiple parts such as sensors, cameras, manipulators, etc. which are each represented as a coordinate frame, where each frame is connected to another frame using joints. When trying to move a specific part or coordinate frame, not only the transform of that single frame but the composite transform of each frame in relation to the target has to be calculated. This is especially important when moving a robotic arm based on sensor readings. In this example a transform between the position of the sensor and the arm needs to be calculated so the motion performed by the arm the motion perceived by the sensor match. Using the ROS Transform Library (tf) these complex calculations can be facilitated. To this end tf keeps track of each coordinate frame in a acyclic relationship tree where tf broadcasters then publish relative pose information and listeners query transforms between two coordinate frames. Because not all pose information in a robotic system is

[10]Srinivasan, *RPC: Remote Procedure Call Protocol Specification Version 2*, Page 3.
[11]Open Source Robotics Foundation, *Services - ROS Wiki*.
[12]Open Source Robotics Foundation, *Master - ROS Wiki*.
[13]Open Source Robotics Foundation, *Parameter Server - ROS Wiki*.

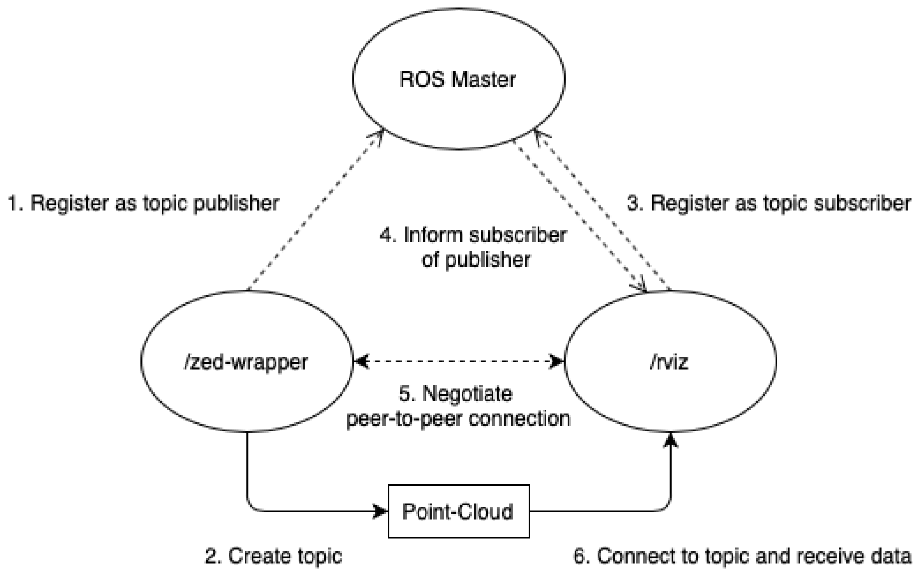

Figure 8.1: Diagram of the topic registration process in ROS at which the publishing node registers its topic before the subscriber tells the master its interest in the point-cloud topic. However a node can be registered as a subscriber to a specific topic without the topic existing yet.

instantly accessible, the tf saves this information for each frame over time. This means that transforms can be queried not just spacially but also temporally.

8.8 Simulation

Debugging and testing robot applications can be a repetitive and tedious task, especially when a test environment needs to be reset at every test cycle. In order to facilitate this part of development the Autumn robot utilises the representation and simulation technologies that are described in the following sections.

8.8.1 URDF

In order to perform simulations or compute coordinate frame transforms a robot needs to be described in some way. One of the more popular description formats is the Unified Robot Description Format (URDF) which provides a XML format for representing a robot and its components as well as a *C++* parser and tools to convert and verify and visualize these models.[14] The *check_urdf* tool parses a URDF-File and returns the robots kinematic chain if successful. To visualize the robots frames and joints in a graphviz[15] tree the *urdf_to_graphiz* tool can be used. The resulting tree corresponds to the relationship tree tf uses to calculate transforms.

[14]Open Source Robotics Foundation, *URDF- ROS Wiki*.
[15]The Graphviz Authors, *About - Graphviz*.

8.8.2 Gazebo Simulator

Gazebo is a 3D physics simulator often used in close relation to ROS projects. Therefore it provides tooling for model and world design and generation as well as comprehensive interfaces for controlling a simulated robot through ROS. Further advantages of Gazebo are its accurate sensor and sensor noise generation as well as its large community providing countless models of robots and sensors.

Chapter 9

Simultaneous Localization and Mapping

Author: Lukas Leskovar
The problem of localizing as well as navigating a system through a completely or partly unknown environment without any external coordinate system (i.e. GPS, Optical Beacon Tracking, etc.) has proven itself to be one of the most complex and yet fundamental topics in many scientific research fields with robotics being most prominent. The main approach to this problem is Simultaneous Localization and Mapping (SLAM) which dates back to the mid 1980s. Back then the first solutions based on Extended Kalman Filters or Rao-Blackwellised Filters were formulated.[1][2]
To date the topic of SLAM has matured, algorithms have gotten more reliable and robust and are utilized in many industries. Applications for SLAM range from Navigating a Mars Rover over autonomous cars or warehouse robots to simple household appliances like vacuum cleaners.
As one major topic of this thesis is robot navigation in a GPS-denied area as well as mapping of such environment, different modern SLAM approaches and solutions utilized by the project team are discussed in this chapter.

9.1 Localization

Localization or state-Estimation aims to reconstruct the state of a system using interoceptive measurements (e.g. acceleration, velocity, etc.) as well as an exteroceptive model (e.g. position and orientation) of the system.[3] Put into the context of mobile robotics this means that in order to perform comprehensive localization of a mobile robot a sensor fusion between on-board sensors and an external coordinate system or map ought to be performed as solely relying on incremental sensors for odometry would quickly result in large accumulated errors.

[1] Durrant-Whyte et al., "Simultaneous localization and mapping: part I".
[2] Cadena et al., "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age".
[3] Barfoot, *State estimation for robotics*.

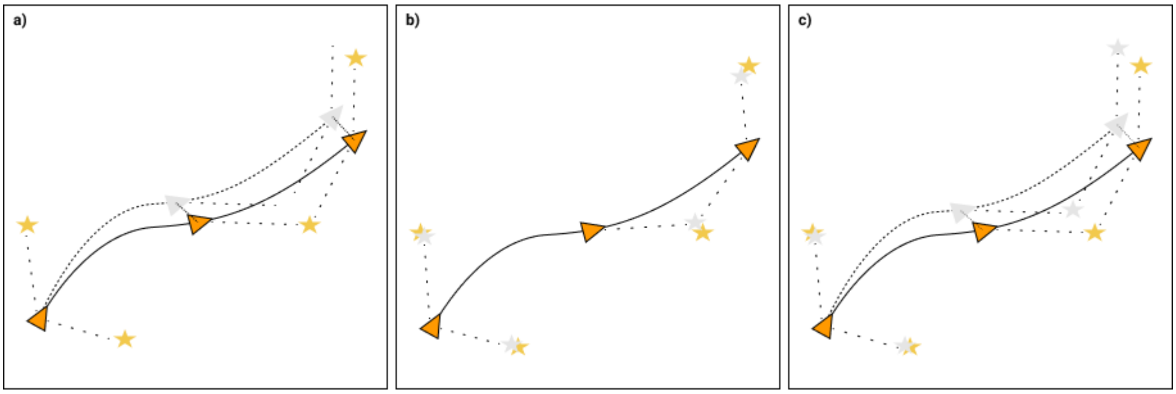

Figure 9.1: The figure above depicts the different tasks contributing to the SLAM problem. Panel a) shows a robot moving trough a known environment while the pose and path remain uncertain. In panel b) the robots pose is certain while the landmark locations need to be determined. Panel c) composes both previous problems where neither the robot state nor the landmark locations are certain.

An for this exemplary system would be an industrial robot utilizing a construction plan to correct errors by wheel-encoded odometry as well as to navigate through a factory.

9.2 Mapping

Contrary to localization or state-estimation, in mapping problems the current pose of the system is known while its environment remains uncharted. Therefore stand-alone mapping aims to generate a model of its environment by evaluating sensor readings of environmental features as well as the systems pose to reconstruct aforementioned features in a global reference frame. Mapping applications often combine technologies typically found in other scientific fields such as photogrammetry or computer vision. An example for such a system would be a drone computing camera images and GPS positional data with a photogrammetric algorithm to reconstruct a 3D-Model of a building.

9.3 Localization and Mapping

The aforementioned technologies on their own are considered rather simplistic problems as either the environments map is given or reliable pose-estimation can be provided. While most applications meet either of these criteria with pre-built maps or GPS being available, in some environments such as indoors, mineshafts or outer space both the systems pose and environment remain uncertain and need to be determined simultaneously hence the name Simultaneous Localization and Mapping. To summarize, the crux of the SLAM problem, as described in Fig. 9.1, is that localizations requires a map and mapping depends on pose estimates however neither are certain.

9.4 Map Representation

The way a robot perceives its environment and maintains a accurate map of its environment is directly dependant on many criteria such as complexity of tasks, size of its environment as well as measurement quality mainly influenced by sensor noise. Tailoring to benefit some of the aforementioned criteria most robotic mapping systems utilize either of two paradigms, metric or topological, each proposing their respective strengths and weaknesses.
Metric or grid-based maps build a map of the robots environment as occupancy grids, with each grid cell indicating the presence of an obstacle. The main benefit using metric maps is the facilitated construction of large-scale mappings as well as non-ambiguous determination of places. However such maps have significant drawbacks concerning space as well as time complexity and require accurate pose estimation of the robot.
Topological or feature-based maps reconstruct their environment as graphs with each node representing a feature or landmark perceived by the robots sensors. In contrast to metric maps they allow for comprehensive path planning and are significantly more compact as its resolution is directly proportional to the environments complexity.[4]

9.5 Probabilistic Definition

One of the first To describe probabilistic SLAM lets assume a robot is moving through an environment observing multiple landmarks at different times. The goal is that, for any time instant t, the robots state vector

$$x_t = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

describing its position and orientation as well as a time invariant set of all absolute landmark positions $m = \{m_1, m_2, ..., m_i\}$ is computed, given uncertain relative landmark observations $Z_{0:t} = \{z_0, z_1, ..., z_t\}$, controls $U_{0:t} = \{u_0, u_1, ..., u_t\}$ and known initial location x_0. Fig. 9.2 depicts this process and illustrates the relationships between the variables important in any SLAM system.

$$P(x_{0:t}, m | Z_{0:t}, U_{0:t}, x_0) \tag{9.1}$$

In 9.1, which is often referred to as "Full-SLAM" or "Offline-SLAM", the joint posterior density of the robots trajectory and landmark locations is estimated. In other words, this formulation of the problem aims to recover the whole robot trajectory.

$$P(x_t, m | Z_{0:t}, U_{0:t}, x_0) \tag{9.2}$$

The pendant to this is "Online-SLAM" which aims estimate only the robots latest location at time instant t, as seen in 9.2. In contrast to "Full-SLAM", performing batch computation on the whole data, algorithms pursuing this online approach typically compute the probability distribution incrementally.

[4] Thrun, "Learning metric-topological maps for indoor mobile robot navigation".

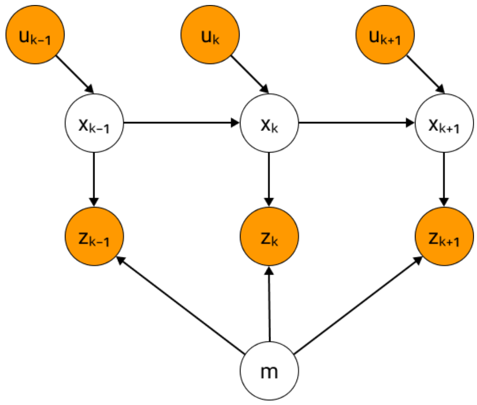

Figure 9.2: A Bayes network graph depicting the causal relationships between sensor measurements (shaded circles) and uncertain variables (blank circles). It is made apparent how the relative measurements by odometry or landmark observation influence the robot state as well as landmark locations.

To compute either problem a mathematical model representing the robots state transition and a model describing observations need to be defined:

$$P(x_t|x_{t-1}, u_t) \tag{9.3}$$

$$P(z_t|x_t) \tag{9.4}$$

The motion model 9.3 describes the robots location at time t given a known previous location x_{t-1} and odometry u_t assuming the state-transition is a Markov process[5]. 9.4 calculates the probability of observing a landmark z_t when the robots location x_t and environment m are known. For aforementioned reasons in SLAM the map is not known to the system, therefore many formulations omit the map in the observation model.
For filtering approaches which are discussed later in this chapter, the joint posterior is then implemented in a recursive prediction and correction form which is often described as time-update and measurement-update:

$$P(x_t, m|Z_{0:t-1}, U_{0:t}, x_0) = \int P(x_t|x_{t-1}, u_t)P(x_{t-1}, m|Z_{0:t-1}, U_{0:t-1}, x_0)dx_{t-1} \tag{9.5}$$

$$P(x_t, m|Z_{0:t}, U_{0:t}, x_0) = \frac{P(z_t|x_t, m)P(x_t, m|Z_{0:t-1}, U_{0:t}, x_0))}{P(z_t|Z_{0:t-1}, U_{0:t})} \tag{9.6}$$

The time update in 9.5 calculates an estimate of the robots state using a previously known state at $t-1$ and state-transition given the newest control u_t. In the next step, the measurement update 9.6, a observation z_t is taken and the previously calculated estimate is corrected using Bayes-Theorem[6].

[5] Haenelt, Karin, *Hidden Markov Models (HMM)*.
[6] Durrant-Whyte et al., "Simultaneous localization and mapping: part I".

To make the the recursive structure of such filtering approaches more apparent the equations 9.5 and 9.6 can be compared to the Bayes filter, commonly used as a base algorithm for further state-estimation problems[7].

$$\overline{\text{bel}}(x_t) = \int P(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{9.7}$$

$$bel(x_t) = \eta P(z_t|x_t) \overline{\text{bel}}(x_{t-1}) \tag{9.8}$$

9.6 Solution Paradigms

Over time many different approaches to the SLAM problem have been developed which can be categorized into either of three paradigms. The two mature ones are based on recursive filtering techniques while the modern graph-based solutions perform non-linear sparse optimization.

9.6.1 Extended Kalman Filter

The Extended Kalman Filter (EKF) is one of the first approaches to the Online-SLAM problem and similar to the standard Kalman Filter one implementation of a Bayes Filter. Contrary to the Kalman Filter it is applicable to non-linear systems by performing linear approximation.
The Kalman Filter assumes a environment in which landmarks can be fully represented as points within the coordinate frame. The earlier introduced state-vector x is extended to a state-space vector with dimensions $3 + 2N$

$$\mu = \begin{pmatrix} x \\ m \end{pmatrix} = \begin{pmatrix} x & y & \theta & m_{1,x} & m_{1,y} & \dots & m_{n,x} & m_{n,y} \end{pmatrix}^T$$

to incorporate the landmark locations $m_{i,x}$ and $m_{i,y}$.
Furthermore a covariance matrix $\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix}$ is introduced which specifies the state x and landmark m uncertainties as well as correlations between landmarks and robot pose.
The motion and observation models are implemented as non-linear functions $x_t = g(u_t, x_{t-1} + \epsilon_t$ and $z_t = h(x_t) + \delta_t$ with ϵ_t and δ_t defined as random noise variables commonly based on zero-mean additive Gaussian noise.
To still calculate the state transition as well as measurement update these functions are linearized by performing first order Taylor Expansion as follows[8]:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) = g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{9.9}$$

$$h(x_t) \approx h(\overline{\mu}_t) + h'(\overline{\mu}_t)(x_t - \overline{\mu}_t) = h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t) \tag{9.10}$$

[7] Thrun, *Probabilistic robotics*, Page 23.
[8] Ibid., Pages 33-51.

This allows for the functions to be incorporated in the vanilla Kalman Filter Algorithm which consists of the following five step procedure:

```
def extended_kalman_filter(μ_{t-1}, Σ_{t-1}, u_t, z_t):
    μ̄_t = g(u_t, μ_{t-1})      # state-prediction given newest control measurements
    Σ̄_t = G_t Σ_{t-1} G_{t-1}^T + R_t      # measurement prediction

    K_t = Σ̄_t H_t^T (H_t Σ̄_t H_t^T + Q_t)^{-1}      # measurement of actual landmark position
    μ_t = μ̄_t + K_t(z_t - h(μ̄_t))      # data association
    Σ_t = (I - K_t H_t) Σ̄_t      # update of state-space vector as well as covariance matrix

    return μ_t, Σ_t
```

Listing 9.1: Pseudo-Code describing the filter cycle of a Extenden Kalman Filter

Notice that the Jacobian matrices G_t and H_t containing all partial derivatives of g and h are utilized describing the transformation of robot state as well as observations based on the pose.

9.6.2 Particle Filter

One major problem with the EKF paradigm is its inability to perform estimations under non-Gaussian distributions. To this end particle filters estimate a specific probability distribution by proposing multiple possible hypothesis which get continuously more accurate as the algorithm proceeds. Particle filtering for SLAM with algorithms like Fast-SLAM, shares many similarities with Monte-Carlo Localization (MCL) which is roughly described in the following steps:

- For each state x_t a set $\mathcal{X} = \{\langle x_t^{[j]}, \omega^{[j]}\rangle, ...\}_{j=1,...,J}$ particles or samples with random state hypothesis is generated from a proposal distribution.
- The probability for each particle to be representing the true state is calculated by comparing its estimation with a measurement thus yielding in each particles importance weight $\omega^{[j]}$.
- The last step is to remove samples with a lower likelihood of being correct and add more samples in regions with higher importance weights, this process is called resampling and shares many similarities with survival of the fittest algorithms.

Because the number of particles greatly affects the performance of a particle filter, such approaches perform best in low dimension spaces. To this end the Fast-SLAM algorithm utilizes Rao-Blackwellization, see Equation 9.11, to separate the joint probability of robot state and map into individual distributions which can be computed with much more ease. This means that given the robots poses all landmarks are independent, which is described in Fig. 9.3.

$$P(x_{0:t}, m|Z_{0:t}, U_{0:t}) = P(x_{0:t}|Z_{0:t}, U_{0:t})P(m|x_{0:t}, Z_{0:t}) \tag{9.11}$$

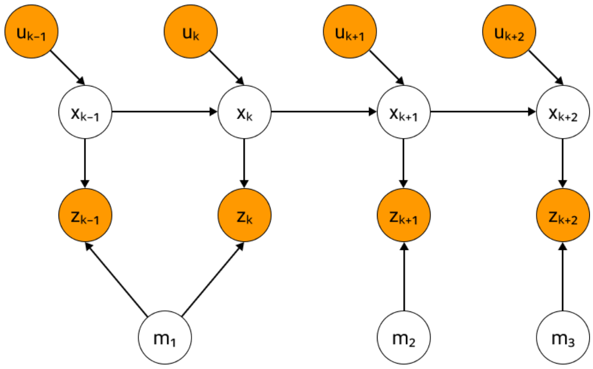

Figure 9.3: This graphical description of the SLAM problem shows the assumption that the landmark estimates are all mediated through the robot path thus allowing for easier implementation by using smaller Gaussians for each landmark m_i rather than a larger state-space vector as in EKF SLAM.

Following this approach the robots path is estimated by performing a Monte-Carlo localization drawing samples from the robots motion model $x_t^{[j]} \sim P(x_t|x_{t-1}^{[j]}, u_t)$. Besides the state information each particle maintains a set of landmarks, each represented as a low dimensional EKF. Therefore the importance weight for each particle corresponds to the accuracy between expected measurement and observation as $\omega^{[j]} \sim P(z_t|x_t^{[j]}, \overline{z}_t)$. The final algorithm functions exactly as described with the exception that it incorporates the aforementioned EKF to estimate the landmark locations and compute the particle weight before updating its belief on landmark locations and initiating the resampling process[9].

9.6.3 Graph-based

The newest paradigm to solve the offline SLAM problem are graph-based methods that essentially use a non-linear least-squares approach to compute the graph of poses best fitting the measurements. To this end the SLAM posterior is described as a graph of robot poses x_i at time t_i as nodes connected by soft constraints that correspond to spatial measurements between two states. These edges are generated in the following two cases:

$$(Xi^{-1}X_{i+1}) \tag{9.12}$$

$$(X_i^{-1}X_j) \tag{9.13}$$

Here 9.12 corresponds to a edge based on odometry transformations by moving the robot from state X_i to X_{i+1}, both of which are described as transformation matrices relative to the coordinate origin. These transformations are illustrated in more detail in Fig. 9.4. The more interesting case is when an observation of a previously captured area occurs and a virtual measurement 9.13 is computed, describing how the states X_j and X_i relate to each other according to the displacement between their observations, see Fig. 9.5.

[9]Stachniss et al., "Simultaneous localization and mapping", Pages 1159-1162.

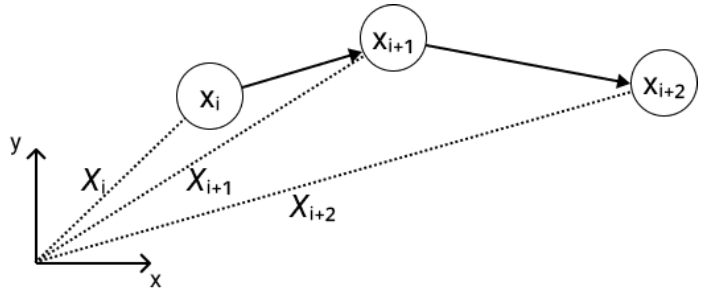

Figure 9.4: The graph of robot poses connected to each other by odometry measurements. To calculate the relative movement between two nodes each node can be represented as a matrix X_i describing how it is transformed relative to the global coordinate frame.

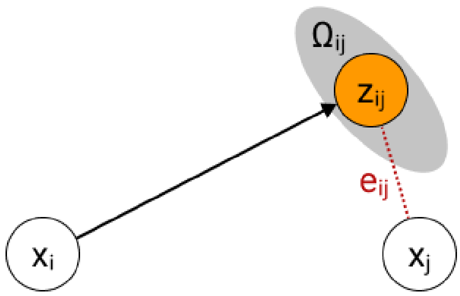

Figure 9.5: This pose-graph with two nodes describes how any pair of nodes are related to each other given a measurement. The measurement z_{ij} corresponds to how the node x_j should be placed relative to node x_i given that each of them share a similar observation. As seen in the figure this is not the case, thus demanding for the error e_{ij} to be minimized. In other words, by moving the node x_j closer to the optimum minimizes the error and corrects the pose-graph.

This approach marginalizes out any data about the map other than the correspondence of poses to primarily focus on solving the robot path. Given the pose-graph the goal is to compute the node configuration that minimizes the error between predicted edges, see 9.13, and actual measurements $\langle z_{ij}, \Omega_{ij} \rangle$, with the information matrix Ω_{ij} describing the measurement noise.
Equation 9.14 describes this optimization process, with the error function e_{ij} defined as any non-linear function essentially calculating the difference between estimate and true measurement for any pair of nodes. Once the optimal pose-graph is constructed the map data is rendered given known poses thus reducing the problem to simple mapping[10].

$$x* = argmin_x \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij} \tag{9.14}$$

[10] Grisetti, Kümmerle, et al., "A tutorial on graph-based SLAM".

9.7 Graph-SLAM topology

Most modern graph-based SLAM systems are partitioned into the following modules:

- A front-end in charge of abstracting sensory input by performing feature extraction as well as data association, see Section 9.8. Essentially it constructs the pose-graph and feeds it into the back-end.

- A back-end performing pose-graph optimization based on the abstracted data fed in by the front-end. The back-end also provides positional information about nodes to the front-end thus supporting loop closure detection.

While the front-end remains application specific as the data abstraction depends on the sensors utilized by the robot, the same back-end can be applied for many different scenarios as the pose-graph optimization essentially represents a squared-error minimization and many elaborated algorithms such as Gauss-Newton.

9.8 Data Association

One key trait that all SLAM paradigms posses is to recognise similarities in measurements and correct the incremental odometry error after revisiting previously mapped areas. This process is referred to as loop-closing or long-term data-association and reduces the uncertainty of pose and map estimates thus qualifying it as a integral part of any SLAM problem.
While loop-closures aim to detect large correspondences in the global map, short-term data-association is responsible for associating corresponding features over multiple measurements which helps establishing the pose-graph.

9.9 Hierarchical Pose-Graphs

While building the pose-graph can be computed with relatively low effort, performing loop-closures is a computational heavy process. Which makes graph-based online SLAM solutions difficult to implement as robot state and map need to be estimated in real-time scenarios. To facilitate the data association process, the global map is partitioned into sub-maps thus allowing data-associations to be calculated only in parts of the graph the robot is currently present.

9.10 Comparison of SLAM Paradigms

The Extended Kalman Filter is a mature approach to the SLAM problem and has been implemented many times in different types of applications. However it does not only lack computational efficiency in its original implementation, see. Tab. 9.1, but also has shown to be very problematic concerning consistent solution and robust data-association. These problems stem from the algorithm definition itself, firstly as local-linearization of non-linear problem impairs the system to reach complete convergence, e.g. all landmark uncertainties

Approach	Problem	Solution	Time Complexity
EKF	online SLAM	KF with local linearization	$O(m^2)$
Fast-SLAM	full & online SLAM	Based on MCL	$O(jm)$
Graph SLAM	full SLAM	Least-Squares optimization	$O(\|V\| + \|N\|)$

Table 9.1: The three main SLAM solution paradigms in basic implementation compared by their computational complexity. Note that the quadratic time complexity of the EKF approach is caused by quadratically extending the covariance matrix as landmarks m are added and updated throughout mapping. For Fast-SLAM the complexity concerning just the particles j stays linear, however during every resampling step each particle including all landmark estimate m contained within need to be copied. Lastly the back-end for most graph SLAM solution is based on least-squares error minimization of the pose graph with $|N|$ nodes as robot poses and $|V|$ vertices as measurements between nodes thus rendering the computational complexity to be $O(|V| + |N|)$ without utilized hierarchical pose-graphs as studied by Korovko and Robustov[15].

converging to zero. Secondly the data-association model of EKF assumes all landmarks to be fully identifiable from any point of view, which cannot be guaranteed in most real-life scenarios and renders data-association very fragile. Fortunately the EKF has been improved and optimized to address some of the aforementioned issues[11].
Another paradigm proposed was Particle Filtering which essentially applies a survival of the fittest model to the robot poses. Solutions like Fast-SLAM optimize MCL algorithm by decoupling landmarks and estimating through individual EKFs. The main benefit of this solution is the utilization of a non-linear model and ability to perform even with non-Gaussian distributions. Although its computationally more efficient compared to the previous approach, one problem is the need for large amounts of particles to achieve high accuracy. Montemerlo[12] suggest implementing a balanced binary tree to store and organize particles which improves the complexity of the algorithm to $O(jlogm)$ with j representing the amount of particles and m being the number of landmarks.
Lastly mentioned was the graph-based approach to SLAM which has grown significant popularity due to its versatility and reusability in different applications. The robot posterior is modelled as a sparse pose-graph which is optimized using a least-squares algorithm and is solved offline after the whole robot path has been captured. Some newer algorithms try to manipulate the pose-graph in real-time to establish a graph-based online SLAM solution[13].

9.11 SLAM in Autumn

As localization and mapping data is utilized to perform real-time path-planning as well as drone control these modules are a crucial part of the Autumn Drone, thus requiring a algorithm that solves the online SLAM problem in a efficient manner while maintaining

[11] Bailey et al., "Simultaneous localization and mapping (SLAM): Part II".
[12] Montemerlo et al., "FastSLAM: A factored solution to the simultaneous localization and mapping problem".
[13] Stachniss et al., "Simultaneous localization and mapping".

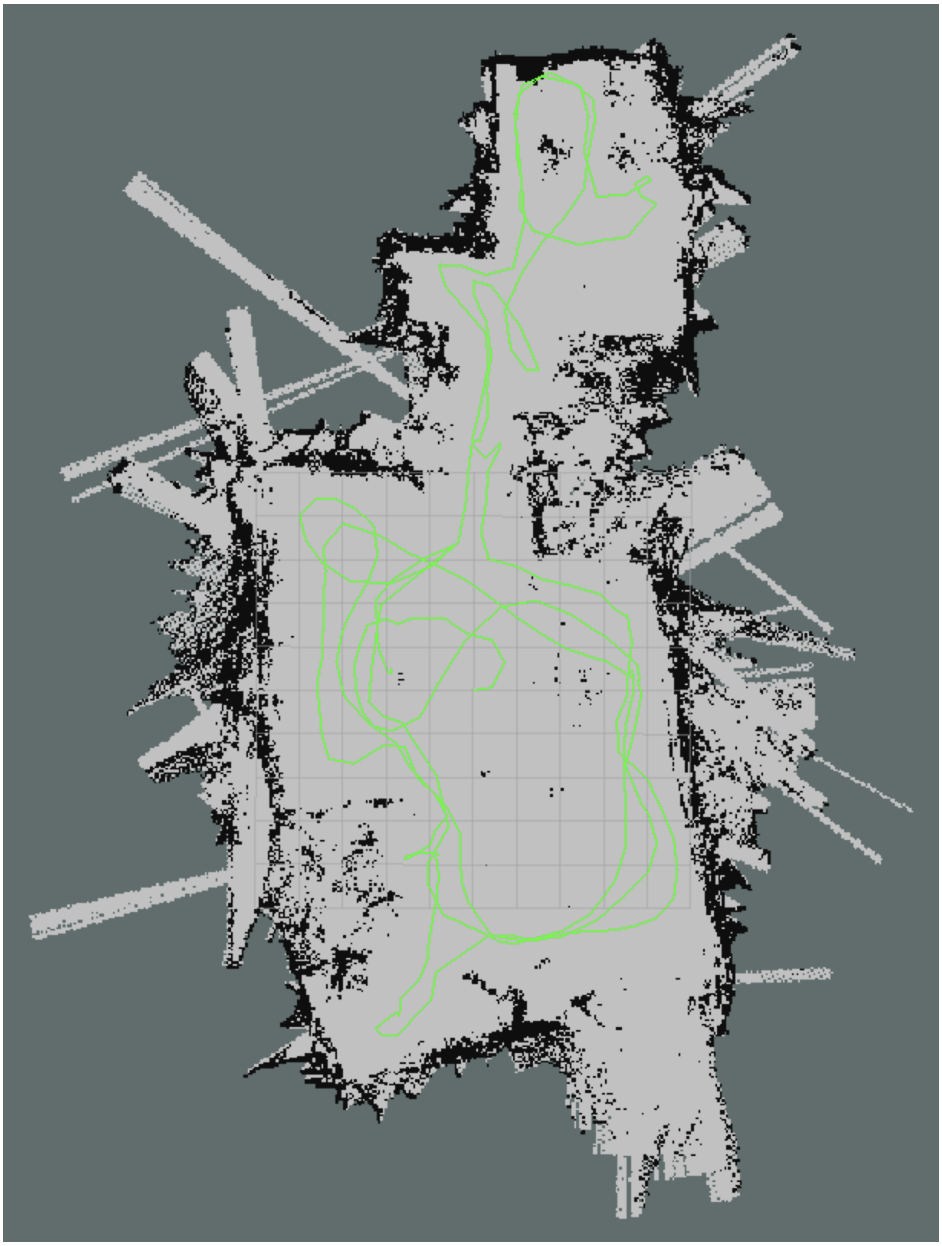

Figure 9.6: Occupancy grid of the AIRLab at the HTBLuVA Wiener Neustadt including the path pursued by the drone as it got carried through the laboratory. The map as seen above is of low quality including many artefacts, sensor noise as well as misaligned proportions. Although the main room is poorly mapped with many error, the smaller workshop at the top is depicted relatively accurate.

compatibility to the hardware in use. This restraints the solution in question to perform visual SLAM supporting stereo images and 6DoF-Odometry. While remaining within the package-space of ROS this reduces the number of potential algorithms to the following graph-based SLAM solutions:

- ORB-SLAM2
- S-PTAM
- RTAB-Map
- RGBDSLAMv2

ORB-SLAM2 and S-PTAM where not selected as both approaches do not provide online occupancy grids as ROS-Topics which makes them infeasible to use and incorporate with the path-planning module of Autumn. Furthermore they do not propose a memory-management strategy thus potentially making loop-closure processing within large maps cumbersome.
The algorithm chosen to be used for Autumn was RTAB-Map as incorporating stereo-images and external odometry is supported out-of-the-box and a variety of outputs such as 2D and 3D occupancy grids as well as a dense point cloud are offered. As loop-closure detection algorithm with memory management was the primary goal of RTAB-Map, it qualifies for usage in large-scale environments during long mapping sessions. Since its publication in 2013 this algorithm has been extended and improved to support stereo- or RGBD-footage, 2D as well as 3D Lidar data and provides nodes for processing visual or lidar based odometry if no other external topic is available. For Autumn this approach was implemented by providing the stereo-image stream as well as visual odometry data from the Stereolabs ZED2i thus yielding in multiple occupancy grids, robot paths as visualized in Fig. 9.6.
Although RGBDSLAMv2 does not fully qualify to be considered in this project as it only works with RGBD-Cameras like the XBOX Kinect or Intel RealSense it is worth mentioning as it provides a 3D occupancy grid and a dense point-cloud similar to RTAB-Map.[16]

9.12 Topics not mentioned in this Chapter

The SLAM problem is a highly diversified field of study with many different concepts not only concerning one paradigms adaption to other use cases than originally mentioned but also diving into the exact implementations of any of its component. Due to this abundance of concepts and algorithms it would be unfeasible to characterize every one of them. However for the intrigued reader the author suggests the following topics and resources for further reading:

- Particle Filtering for grid-based SLAM[17]

[16] Labbé et al., "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation".
[17] Grisetti, Stachniss, et al., "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters By Adaptive Proposals and Selective Resampling".

- Visual SLAM approaches. A wide range of literature on this topic is provided by the Technical University in Munich (TUM), Germany.
- Graph-SLAM frontends using different data association approaches for visual SLAM such as feature based matching like RANSAC[18] or point-to-point matching like ICP[19].
- Mapping in dynamic environments capable of comprehending moving obstacles within the environment[20]
- SLAM with multiple robots[21]

[18] Jia et al., "An Improved RANSAC Algorithm for Simultaneous Localization and Mapping".
[19] Horn et al., "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices".
[20] Wang et al., "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas".
[21] Gutmann et al., "Incremental Mapping of Large Cyclic Environments".

Chapter 10

Methodology

Author:

Chapter 11

Implementation

Author:

Chapter 12

Experiment 1

Author:

Chapter 13

Lessons learned

Author:

Chapter 14

Experiment 2

Author:

Chapter 15

Conclusion

Author:

Index

Authors Index

Literature Index

Bibliography

Bailey, Tim and Hugh Durrant-Whyte. "Simultaneous localization and mapping (SLAM): Part II". In: *IEEE robotics & automation magazine* 13.3 (2006), pp. 108–117.

Barfoot, Timothy D. *State estimation for robotics*. Cambridge University Press, 2017.

Bekey, George A. *Autonomous robots: from biological inspiration to implementation and control*. en. MIT press, 2005. URL: https://books.google.at/books?hl=de&lr=&id=8MbxCwAAQBAJ&oi=fnd&pg=PR7&dq=autonomous+robots&ots=5EXAZx-UDf&sig=u_CyTA0fa4aMRFAAdnbXpKhaf94&redir_esc=y#v=onepage&q=autonomous%20robots&f=false (visited on 06/30/2021).

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John 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. DOI: 10.1109/TRO.2016.2624754. (Visited on 09/26/2021).

Durrant-Whyte, H. and T. Bailey. "Simultaneous localization and mapping: part I". In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110. DOI: 10.1109/MRA.2006.1638022. (Visited on 09/26/2021).

Emesent. *Emesent_Hovermap.JPG (JPEG Image, 2018 × 910 pixels)*. URL: https://www.emesent.io/wp-content/uploads/2020/07/DSC_3700-1finalcrop.jpg?id=7339 (visited on 07/04/2021).

Emesent Hovermap. *Autonomy Level 2 for Emesent Hovermap*. URL: https://www.emesent.io/autonomy-level-2/ (visited on 07/04/2021).

Exyn Aero. *Exyn Aero - Aerial Mapping Drone*. URL: https://www.exyn.com/products/exyn-aero-aerial-mapping-drone (visited on 07/04/2021).

Gerkey, Brian. *Definition - ROS Answers*. en. URL: https://answers.ros.org/question/12230/what-is-ros-exactly-middleware-framework-operating-system/ (visited on 07/08/2021).

Grisetti, Giorgio, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. "A tutorial on graph-based SLAM". In: *IEEE Transactions on Intelligent Transportation Systems Magazine* 2 (Dec. 2010), pp. 31–43. DOI: 10.1109/MITS.2010.939925.

Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard. "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters By Adaptive Proposals and Selective Resampling". In: Jan. 2005, pp. 2432–2437. DOI: 10.1109/ROBOT.2005.1570477.

Gutmann, Steffen and Kurt Konolige. "Incremental Mapping of Large Cyclic Environments". In: (Nov. 2003). DOI: 10.1109/CIRA.1999.810068.

Haenelt, Karin. *Hidden Markov Models (HMM)*. de. 2006. URL: https://www.cis.lmu.de/~micha/kurse/statistik-SS2008/begleitmaterial/HMM.pdf (visited on 10/09/2021).

Horn, Berthold, Hugh Hilden, and Shahriar Negahdaripour. "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices". In: *Journal of the Optical Society of America A* 5 (July 1988), pp. 1127–1135. DOI: 10.1364/JOSAA.5.001127.

Jia, Songmin, Zeling Zheng, Guoliang Zhang, Jinhui Fan, Xiuzhi Li, Xiangyin Zhang, and Mingai Li. "An Improved RANSAC Algorithm for Simultaneous Localization and Mapping". In: *Journal of Physics: Conference Series* 1069 (Aug. 2018), p. 012170. DOI: 10.1088/1742-6596/1069/1/012170. URL: https://doi.org/10.1088/1742-6596/1069/1/012170.

Korovko, Alexander and Dmitry Robustov. "Partial Hierarchical Pose Graph Optimization for SLAM". In: *arXiv preprint arXiv:2110.08639* (2021).

Labbé, Mathieu and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446.

Malone, Bob. *George Devol: A Life Devoted to Invention, and Robots*. en. 2011. URL: https://spectrum.ieee.org/automaton/robotics/industrial-robots/george-devol-a-life-devoted-to-invention-and-robots (visited on 06/17/2021).

Montemerlo, Michael, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *Aaai/iaai* 593598 (2002).

NVIDIA Corporation. *NVIDIA Jetson Hardware Page*. en. URL: https://developer.nvidia.com/embedded/jetson-modules (visited on 09/19/2021).

— *Power Management for Jetson TX2 Series Devices - Supported Modes and Power Efficiency*. en. URL: https://docs.nvidia.com/jetson/archives/l4t-archived/l4t-3231/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_tx2_32.html (visited on 09/19/2021).

— *tegrastats Utility*. en. URL: https://docs.nvidia.com/jetson/archives/l4t-archived/l%E2%80%A6 (visited on 09/19/2021).

Open Source Robotics Foundation. *Building Packages - ROS Wiki*. en. URL: http://wiki.ros.org/ROS/Tutorials/BuildingPackages (visited on 07/08/2021).

— *Concepts - ROS Wiki*. en. URL: http://wiki.ros.org/ROS/Concepts (visited on 07/08/2021).

— *Master - ROS Wiki*. en. URL: http://wiki.ros.org/Master (visited on 07/22/2021).

— *Messages - ROS Wiki*. en. URL: http://wiki.ros.org/Messages (visited on 07/09/2021).

— *Metapackages - ROS Wiki*. en. URL: http://wiki.ros.org/Metapackages (visited on 07/08/2021).

— *Nodes - ROS Wiki*. en. URL: http://wiki.ros.org/Nodes (visited on 07/09/2021).

— *Packages - ROS Wiki*. en. URL: http://wiki.ros.org/Packages (visited on 07/08/2021).

— *Parameter Server - ROS Wiki*. en. URL: http://wiki.ros.org/Parameter%20Server (visited on 07/22/2021).

— *Services - ROS Wiki*. en. URL: http://wiki.ros.org/Services (visited on 07/09/2021).

— *URDF- ROS Wiki*. en. URL: http://wiki.ros.org/urdf/Tutorials (visited on 07/28/2021).

Srinivasan, Raj. *RPC: Remote Procedure Call Protocol Specification Version 2*. RFC 1831. Aug. 1995. DOI: 10.17487/RFC1831. URL: https://rfc-editor.org/rfc/rfc1831.txt.

Stachniss, Cyrill, John J Leonard, and Sebastian Thrun. "Simultaneous localization and mapping". In: *Springer Handbook of Robotics*. Springer, 2016, pp. 1153–1176.

Stephens, Rod. *Beginning Software Engineering*. en. Wiley, 2015. ISBN: 9781118969168. URL: https://books.google.at/books?id=SyHWBgAAQBAJ (visited on 07/09/2021).

The Graphviz Authors. *About - Graphviz*. en. URL: https://graphviz.org/about/ (visited on 07/28/2021).

Thrun, Sebastian. "Learning metric-topological maps for indoor mobile robot navigation". In: *Artificial Intelligence* 99.1 (1998), pp. 21–71.

— *Probabilistic robotics*. Vol. 45. 3. ACM New York, NY, USA, 2002.

Wang, Chieh-Chih, C. Thorpe, and S. Thrun. "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. 2003, 842–849 vol.1. DOI: 10.1109/ROBOT.2003.1241698.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —

Breite = 100 mm
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —