Univerza v Ljubljani
Fakulteta za elektrotehniko

MATEJ JEGLIČ

OCENJEVANJE STANJA VRAT ZA MOBILNE GOSPODINJSKE ROBOTE

Magistrsko delo

Mentor: Prof. Dr. Sašo Blažič

Ljubljana, 2018
University of Ljubljana
Faculty of Electrical Engineering

MATEJ JEGLIČ

DOOR STATE ESTIMATION FOR A MOBILE HOUSEHOLD ROBOT

Master Thesis

Mentor: Prof. Dr. Sašo Blažič

Ljubljana, 2018
Acknowledgement

First, I would like to express my deep gratitude to my supervisor and project leader, Dr.-Ing. Katharina Herthorn, for her valuable guidance, constructive suggestions and useful critiques of my master thesis. Her encouraging approach and enthusiastic leadership was very helpful during the difficult parts of the project. I would like to thank her for the devoted time and contributions she made to this master thesis.

Second, I wish to thank Martin, Sören, Thilak, Tom, Yordanka and others, who participated in the project, for their technical support and collaboration during the development process.

Third, I would like to thank my university mentor, Prof. Dr. Sašo Blažič, for his time and input dedicated to this master thesis. His patience and flexibility throughout the process has been very much appreciated.

Last but not least, I would also like to express my gratitude to my family and friends for their support from the beginning till the end of my university years.
# Table of Contents

1 Introduction .................................................. 7

2 Mobile Robots ................................................ 9
   2.1 General Overview ........................................ 9
   2.2 Mapping .................................................. 11
   2.3 Localization ............................................. 12
   2.4 Path Planning ........................................... 14

3 Door State Estimation .................................... 17
   3.1 Problem Description .................................... 17
   3.2 Requirements and Assumptions .......................... 18
   3.3 Kinematic Model of a Hinged Door ...................... 20
   3.4 State of the art ......................................... 25
   3.5 Approaches .............................................. 27
   3.6 Description of the Door State Estimation Algorithm 31

4 Implementation .............................................. 43
   4.1 Robot Hardware ......................................... 43
   4.2 Robot Software .......................................... 47
List of Figures

3.1 Top view of different door types. ................................. 20
3.2 Position of coordinate frames (c.f.) in the kinematic model of a hinged door. ........................................... 21
3.3 Three potential scenarios for sensor, door and obstacle layout. .. 29
3.4 Relation between coordinate frames. ............................... 32
3.5 Illustration of a ray and a circle segment. .......................... 33
3.6 Illustration of a laser beam and the door area. ................. 34
3.7 Illustration of three different ways of laser beam crossing the door area. .................................................... 35
3.8 The discretization of the door area into seven bins ($n_{bin} = 7$) with equal size ($\alpha_{step} = 20^\circ$). ......................... 36
3.9 Vote distribution when a laser hit occurs before the door area. .. 40
3.10 Vote distribution when a laser hit occurs inside the door area. .. 40
3.11 Vote distribution when a laser hit occurs after the door area. .. 41
4.1 The internal structure with all the hardware components of the developed household service robot. ......................... 46
4.2 Software architecture for navigation for the household service robot comprised of four levels: hardware (HW) drivers, low level signal processing (SP), system components and high level commands (HL). 50
4.3 The internal structure of the move_base node with input and output connections obtained from [1].................. 54

4.4 Software architecture for environment mapping using the household service robot comprised of three levels: hardware (HW) drivers, low level signal processing (SP) and system components. . 57

5.1 Setup of the simulation environment....................... 60

5.2 The static map obtained with mapping algorithm in the simulation environment with added map and hinge coordinate frame....................... 62

5.3 The static map obtained with the mapping algorithm in the real-world environment with added map and hinge coordinate frame. . 62

5.4 Calculated global path at different time instances in the simulation environment. Adaptation of the global path is performed when new obstacles are detected with laser scanner....................... 64

5.5 The actual path the robot followed while executing the navigation task in the simulation environment....................... 65

5.6 Calculated global path at different time instances in the real-world environment. Adaptation of the global path is performed when new obstacles are detected with laser scanner....................... 66

5.7 The actual path the robot followed while executing the navigation task in the real-world environment....................... 66

5.8 Five predefined robot positions expressed in the hinge coordinate frame....................... 68

5.9 The output of the door state estimation algorithm for five predefined locations of the robot in the simulation environment............ 69

5.10 The output of the door state estimation algorithm for five predefined locations of the robot in the real-world environment............ 72
5.11 The output of the door state estimation algorithm for five different
door angles $\alpha$ in the simulation environment. . . . . . . . . . . . . 75
5.12 The output of the door state estimation algorithm for five different
door angles $\alpha$ in the real-world environment. . . . . . . . . . . . . 77
5.13 The output of the door state estimation algorithm for four different
types of obstacles in the simulation environment. . . . . . . . . . . . . 81
5.14 The output of the door state estimation algorithm for four different
types of obstacles in the real-world environment. . . . . . . . . . . . . 84
5.15 The output of the door state estimation algorithm when tracking
the door moving with constant angular velocity of $18^\circ$/s in the
simulation environment. . . . . . . . . . . . . . . . . . . . . . . . . . 87
5.16 Computation time of the door state estimation algorithm, when
the robot is positioned in front of the door (position A in Figure 5.8). 89
5.17 Correlation between the number of laser beams that cross the door
area and the computation time of the door state estimation algo-
thesis. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
List of Tables

1. List of used symbols and the corresponding quantities. . . . . . . xv

3.1 Explanation of notation used in Algorithm 1 . . . . . . . . . . . . 38

5.1 The results of the door state estimation algorithm for five prede-
    fined robot positions in the simulation environment. . . . . . . . . 73

5.2 The results of the door state estimation algorithm for five prede-
    fined robot positions in the real-world environment. . . . . . . . . 73

5.3 The results of the door state estimation algorithm for five different
    door angles $\alpha$ in the simulation environment. . . . . . . . . . 78

5.4 The results of the door state estimation algorithm for five different
    door angles $\alpha$ in the real-world environment. . . . . . . . . . 79

5.5 The results of the door state estimation algorithm for four different
    types of obstacles in the simulation environment. . . . . . . . . . 86

5.6 The results of the door state estimation algorithm for four different
    types of obstacles in the real-world environment. . . . . . . . . . 86
List of Used Symbols

In this master thesis the following physical quantities and symbols are used:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Name</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>$t$</td>
<td>second</td>
<td>s</td>
</tr>
<tr>
<td>x-coordinate of a hinge in the static map</td>
<td>$x_{\text{hinge}}$</td>
<td>meter</td>
<td>m</td>
</tr>
<tr>
<td>y-coordinate of a hinge in the static map</td>
<td>$y_{\text{hinge}}$</td>
<td>meter</td>
<td>m</td>
</tr>
<tr>
<td>direction of the door opening in the static map</td>
<td>$\theta_x$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>orientation of the closed door in the static map</td>
<td>$\theta_z$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>width of a door</td>
<td>$w$</td>
<td>meter</td>
<td>m</td>
</tr>
<tr>
<td>height of a door</td>
<td>$h$</td>
<td>meter</td>
<td>m</td>
</tr>
<tr>
<td>thickness of a door</td>
<td>$d$</td>
<td>meter</td>
<td>m</td>
</tr>
<tr>
<td>real door angle</td>
<td>$\alpha$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>maximum door angle</td>
<td>$\alpha_{\text{max}}$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>tolerance band angle</td>
<td>$\alpha_{\text{tol}}$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>size of the door area bins</td>
<td>$\alpha_{\text{step}}$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>potential door angle candidate</td>
<td>$\alpha_{\text{pdac}}$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>number of laser beams</td>
<td>$n_{\text{beams}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>number of door area bins</td>
<td>$n_{\text{bins}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>number of up-votes</td>
<td>$n_{\text{up}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>number of down-votes</td>
<td>$n_{\text{down}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>number of withheld votes</td>
<td>$n_{\text{with}}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>estimation of real door angle</td>
<td>$\alpha_{\text{est}}$</td>
<td>degree</td>
<td>$^\circ$</td>
</tr>
<tr>
<td>confidence in the real door angle estimation</td>
<td>$c_{\text{est}}$</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Vector and matrices in this thesis are written with lowercase bold and uppercase bold writing respectively. The exact meaning of vectors and matrices and their indexes is explained in the accompanying text where the symbols are used.
Abstract

A core task of a domestic mobile robot is the ability to autonomously navigate in the household environment. Crossing through doors is a common subtask of such robots. The first step of door crossing is the estimation of the door state.

This thesis addresses the door state estimation problem. It presents an approach for estimating the door opening angle of a hinged door based on the horizontal laser scanner. The approach assumes that the robot is localized in the environment and that static door parameters (such as the hinge position in the static map, door opening direction and door width) are known. Laser beams that cross the door area are used in a voting scheme to determine the best door angle estimate. The output of the developed algorithm is the estimated door angle and a confidence level of the estimation.

The thesis also validates the developed door state estimation algorithm with experiments which show promising results under the previously mentioned assumptions.

Key words: mobile household robots, autonomous systems with wheels, laser scanner, door state estimation, navigation in household environment
Povzetek


Algoritem za ocenjevanje stanja vrat razvit v sklopu tega magistrskega dela je osnovan na sledečih dveh predpostavkah: robot je lokaliziran v okolju in znani so statični parametri vrat. Algoritem analizira laserske žarke, ki prečkajo področje,
kjer se nahajajo vrata, in na podlagi glasovanja določi najboljšo oceno za kot odprtja vrat. Poleg ocene kota algoritmu poda še stopnjo zaupanja v dani rezultat. Podrobnejši opis algoritma se nahaja v tretjem poglavlju.

Implementacija algoritma je izvedena za delovanje v okolju operacijskega sistema ROS (kratica izhaja iz polnega imena Robotic Operating System – Robotski operacijski sistem), ki vključuje številna orodja, ki olajšajo razvoj mobilnih robotov. Več o podrobnostih in prednostih operacijskega sistema ROS sledi v četrtem poglavju.

V okviru magistrskega dela je bil izdelan konceptni gospodinjski robot, na katerem se bila opravljena testiranja algoritmov. Osnova robota je mobilna platforma z vsesmernim pogonom (angl. omni-directional drive), ki omogoča premikanje v vseh smereh na podlagi (levo, desno, naprej, nazaj) ter vrtenje na mestu. Robot je opremljen z dvodimenzionalnim laserskim skenerjem in štirimi ultrazvočnimi senzorji razdalje za zaznavo okolice. Natančen opis strukture robota in strojne opreme je podan v četrtem poglavju.

Testiranje razvitih algoritmov je potekalo v simulacijskem in realnem okolju. Rezultati testiranja kažejo, da algoritem za ocenjevanje stanja vrat poda natančno oceno kota, s katerim so vrata odprta v pogojih, ko so vrata dobro vidna v meritvah dvodimenzionalnega laserskega skenerja. Testiranja kažejo tudi, da algoritem ločuje med vrati in številnimi ovirami, ki so postavljene med vrata. Opravljene so bile tudi meritve računskega časa, ki ga porabi računalnik za izvedbo razvitega algoritma. Računski čas je kraji od časa vzorčenja dveh zaporednih meritev laserskega skenerja, ki je edina realno-časna omejitev, ki smo si jo zastavili.

Poleg ocenjevanja stanja vrat je robot tudi sposoben zgraditi zemljevid okolja (angl. environment mapping) in avtonomno navigirati v okolju. Programske komponente, ki omogočajo to funkcionalnost, so zagotovljene s strani operacijskega sistema ROS. Testiranje programske opreme je zajemalo tudi grajenje zemljevida okolja in izvajanje avtonomne navigacije. Rezultati teh dveh
komponent, predstavljenih v petem poglavju, so zadovoljivi.

Struktura tega magistrskega dela je naslednja. V prvem poglavju (Introduction) se nahaja kratek uvod, ki pojasni namen in tematiko magistrskega dela. V drugem poglavju (Mobile Robots) sledi kratek pregled področja mobilne robotike z opisom glavnih izzivov, s katerimi se soočajo mobilni roboti. Tretje poglavje (Door State Estimation) se osredotoča na teoretične podrobnosti ocenjevanja stanja vrat. Podrobnejše so opisani in razloženi dani problem, zahteve in predpostavke ter razviti algoritem za ocenjevanje stanja vrat. V četrtem poglavju (Implementation) sledi opis praktičnega dela, ki je bilo izvedeno v sklopu magistrskega dela. Predstavljeni so strojna oprema in zgradba dejanskega robota, uporabljen robotski simulator ter arhitektura programske opreme. V petem poglavju (Validation) se nahaja predstavitev testiranja razvite programske opreme. Podani so opisi izvedenih eksperimentov skupaj z rezultati in njihovim vrednotenjem. V zadnjem poglavju (Conclusion) se nahaja povzetek celotnega magistrskega dela in dobljenih rezultatov ter predlogi za izboljšave in nadaljnjo delo.

Ključne besede: mobilni gospodinjski robot, kolesni avtonomni sistem, laserski skener, ocenjevanje stanja vrat, avtonomna navigacija v stanovanju
Povzetek
1 Introduction

Robotics is an extensive and broad discipline that includes many branches of engineering and science. It is a fast-advancing field of technology whose progress impacts the human lives. Nowadays, all kinds of robots are present in areas such as industry, transportation, agriculture, medicine, military, entertainment, research, etc. The importance of robots is increasing and will undoubtedly continue to do so in the future.

The domain of domestic robots is no different. The number of domestic robots is increasing. In the year 2015, 5 million units of domestic service robots have been shipped according to [2]. By the year of 2020, the number is expected to grow above 12 million shipped units. This shows an increasing demand for robots in people's homes.

This thesis explores the challenges that mobile robots face in a household environment and tries to add a tiny grain of knowledge to a well established research area. The standard topics of mobile robot design, sensor coverage, environment mapping, localization and path planning are briefly addressed. The main contribution of this thesis is in the area of door state estimation. This area is bound to indoor mobile robots and is part of their essential room-to-room navigation. Estimating the correct state of the door enables the robot to select the suitable sequence of actions while navigating from one room to another. The robot’s actions differ whether the door is closed, half-closed or opened. Even the mobile robots without robotic arms are able to push open the half-closed door, if the door pivot in the right direction. To chose the correct action the robot needs a
robust door state estimation algorithm capable of estimating the door angle. In this thesis we present such algorithm that is based on 2D laser scan data.

The thesis is structured as follows. Chapter 2 gives a general overview of mobile robotics and discusses the core problems faced by mobile robots: environment mapping, localization and path planning. Chapter 3 describes the problem of door state estimation, presents the current state-of-the-art methods in this field, discusses potential solutions to tackle the problem and offers a detailed explanation of the developed door state estimation algorithm. Next, in Chapter 4, the practical implementation of the work is described. Chapter 5 follows with result evaluation from testing of the developed robot and the implemented algorithms in the simulation and real-world environment. At the end, in Chapter 6, we present a conclusion of the work and discuss the future steps.
2 Mobile Robots

This chapter gives a brief introduction to mobile robotics. It presents a general overview of the field and analyses the fundamental components of every mobile robot. The core problems of mobile robots (mapping, localization and path planning) are also discussed.

2.1 General Overview

Robots are either fixed to the environment by being mounted to the ground (e.g. industrial robotic arms) or they are able to move in its environment (e.g. Mars Rover). Robots that are able to move (capable of locomotion) are called mobile robots.

Mobile robots are found everywhere: in homes, in hospitals, in large supermarkets, on the fields, in warehouses, in factories, on the roads, in the air, in lakes and oceans, etc. Although mobile robots have a diverse area of application and diverse environmental conditions, they all have a common hardware architecture which includes the following components or parts:

- **actuators** or **motors**, which enable the locomotion of a mobile robot (e.g. DC motors, servomotors, etc.),

- **sensors**, which give a mobile robot the capability to inspect its inner state (e.g. optical encoders, gyroscopes, accelerometers, etc.) or sense the surrounding environment (e.g. laser range finders, ultrasonic sensors, stereo
Mobile Robots

- **mechanical construction**, which holds in place the components and defines the kinematic model of a system,

- **computer unit**, which processes the sensory information and generates the commands for actuators actions (e.g. PC, microcontroller, etc.),

- **power supply unit**, which provides power for the actuators, sensors and computer unit (e.g. battery, solar panels, etc.),

- **telecommunication technology**, which is needed for a mobile robot to communicate with humans or other systems (e.g. WiFi, bluetooth, etc.).

The hardware components must be appropriately supported with software components to achieve a mobile system with autonomous capabilities. The fundamental software components of a mobile robot are the following:

- **low-level hardware drivers**, needed for high speed close-loop control of actuators, filtering and processing of sensory data,

- **navigation components**, with algorithms for robot localization and path planning,

- **intelligence**, a high-level management of robot’s decisions, behaviors and action planning.

Mobile robots are divided into three categories depending upon the environment in which they travel. These three categories are: the Unmanned Ground Vehicles (UGVs) driving on the land, the Unmanned Aerial Vehicles (UAVs) flying in the air and the Autonomous Underwater Vehicles (AUVs) operating in waters. The UGVs, the most common from the three categories, can be further divided into subcategories depending on the type of work the robot performs: delivery robots, warehouse robots, cleaning robots, security robots, garden robots,
2.2 Mapping

exploration robots, service robots, etc. This thesis is focused on the mobile household service robots which are found in the people’s homes.

The purpose of a mobile household service robot is to assist a human in a day-to-day life. The tasks may include transportation of smaller goods around the house, cleaning, telepresence, entertainment features, security patrolling when no one is at home, etc. However, the most fundamental task of a mobile household robot is the ability to autonomously navigate in a household environment. The household environments are very unstructured, unpredictable and diverse. The robot needs to deal with environment that is populated with objects with different shapes, sizes and materials which are lying on the ground or are located at certain heights. Parts of the environment (such as chairs, doors, pets, humans, etc.) even change their position during the operation of the mobile robot. But there are also several advantages of household environment such as steady temperatures, low humidity levels, low degree of dust and a leveled floor surface. Typically the mobile household robots have a differential-drive or a omni-directional platforms, which offer a high degree of agility needed in the environment with limited space. In order to ensure safe and collision-free motion in the complex household environment mobile robots need a comprehensive sensor coverage. Commonly they are equipped with laser range finders, depth cameras, ultrasonic sensors, bumpers, etc.

The complete structure of the mobile household service robot developed within the scope of this thesis is presented in Chapter 4.1.

2.2 Mapping

The first task of a mobile robot in a new environment is to create a map of it. The mobile robot faces the challenge of acquiring the map of the environment, while at the same time tries to localize itself using the very same map. The problem is known as Simultaneous Localization And Mapping (SLAM). There are three
major paradigms of algorithms from which a huge number of published methods are derived.

The first paradigm uses the Extended Kalman Filter (EKF) [3][4] to estimate the robot’s state and was historically the earliest approach. The drawback of EKF SLAM is the computational burden that scales quadratically with the size of the environment. The second paradigm of methods is based on graphical representation and applying a sparse nonlinear optimization methods to SLAM problem. The first graph-based techniques were mention in [5] and [6]. The third paradigm applies nonparametric statistical filtering techniques known as particle filters to the SLAM problem [7][8][9]. Particle filters sidestep the inter-feature correlation of land marks which trouble the EKF. They are also computationally more effective for larger environments, since the computation time scales logarithmically with the size of the environment. Within the scope of this work the particle filter SLAM approach is used for environment mapping presented in [10][11] as described in Chapter 4.5.

Although the filed of SLAM have seen much progress in the last decade there is still room for improvement and granting future work. Vast majority of SLAM techniques mostly deals with the static environments, yet nearly all actual robot environments are dynamic. Another maturing area is the multi-robot SLAM.

2.3 Localization

Robot localization, also called robot pose estimation, is a problem of determining the pose of a mobile robot relative to a given map of the environment. The problem can be also seen as finding the correct transformation between the map coordinate frame and robot’s base coordinate frame. The map of the environment is described in the map coordinate frame and is independent of the robot pose. The robot’s base frame is the root coordinate frame of the robot. The rest of the robot’s components (joints, links, sensors...) are defined relative to the robot’s
2.3 Localization

Unfortunately, the pose of a mobile robot usually cannot be sensed directly. Therefore, the pose must be derived from measured data. A single measurement is usually insufficient to determine the pose, but rather several sensor measurements need to be integrated over time to obtain the pose of the mobile robot.

Three conceptually different localization problems are presented in [12]. The first one is called position tracking problem, where the initial pose of the robot is known. Localization extracts the robot pose during the robot’s operation out of the robot’s motion and sensor measurements which are both tainted with non-deterministic noise. The second is referred to as global localization problem where the robot is placed somewhere in the environment without the knowledge of initial initial position. The global localization problem is a more difficult compared to position tracking problem. The third localization problem is called the kidnapped robot problem. The robot is moved to a new location during its operation, without the knowledge that it has been moved. The localization algorithm must detect the kidnapping action by itself and must then be able to recover from it. This problem is the most difficult from the three presented problems.

Localization algorithms are usually called filters. Filters are recursive and typically work in two steps, a prediction step and a correction step. In the prediction step, a prediction of the robot pose is made based on the previous pose and previously applied actuators inputs and known kinematic model of the robot. In the correction step, sensor measurements are taken and compared against the predicted pose from first step.

Within the scope of this project, robot pose estimation is done using approach called Adaptive Monte Carlo Localization (AMCL) [13] as described in Chapter 4.4. The AMCL is effective and non-parametric localization algorithm, based on a particle filter. The robot pose is presented with the set of particles. In the prediction step, individual particles are moved according to the kinematic
model of the robot and the applied inputs to the actuators. In the correction step, the normalized weights (also called importance factors) are assigned to particles based on matching them with sensor measurements. A new set of particles is obtained by random sampling from the previous set of particles with probability distribution equal to newly calculated weights. The number of particles adapts during the robot’s operation, hence the name Adaptive Monte Carlo Localization, based on error estimation. The method is known as KLD-sampling [14]. High number of particles is needed especially at the beginning to cover the robot’s state space with uniformly random distribution of particles. The number of particle decreases when particles converge around the same location in order not to waste computation resources. The number of particles is a compromise between the estimation accuracy and computational efficiency. The advantages of AMCL are its non-parametric representation of probability distribution capable of handling multimodal distributions, the adaptive set of particles which is not discretized and the ability to solve both position tracking and global localization problems.

2.4 Path Planning

Path planning is problem of finding an optimal (typically the shortest) collision-free path from the current robot pose to the desired goal location. In order to do so, first a representation of the environment is needed. A popular representation of the environment is an occupancy grid which discretizes and divides the environment into small square-shaped cells. Each cell carries an information which expresses the probability that this cell is either free or occupied. A transition of a mobile robot is allowed only between the neighboring cells that are labeled free. A transition graph can be easily obtained from the occupancy grid. The transition graph consists of nodes, which correspond to the free cells, and edges between the nodes, which correspond to the possible transitions between the cells. A certain weight can be assigned to edge to express the cost of the transition (e.g. the distance between the two cells).
Obtaining the optimal path in the transition graph is relevant not only in mobile robotics, but also in other domains, for example in network routing, video games and gene sequencing. Many algorithms were developed to tackle this problem such as Dijkstra, A*, D* and Rapidly-Exploring Random Trees (RRT).

Within the scope of this project A* is used. The A* algorithm starts the search from the node representing the current robot pose and then expands the search to its neighbors until it reaches the goal node. For every node only the information about the optimal path (with lowest cost) to reach this node is stored in the memory alongside with the associated cost. The cost of the path for particular node is a cumulative sum of edge weights from current robot pose until this node, plus an estimated cost-to-the-goal (typically the Euclidean distance or the Manhattan distance to the goal). The A* algorithm explores one node at the time. Which node is to be explored next is determined by the lowest associated cost. Because the algorithm possesses some information about the environment in order to estimate the cost-to-the-goal, the algorithm is called to be informative or heuristic. As long as the estimation of the cost-to-the-goal is less or equal to the true cost-to-the-goal for all nodes, the resulted path will be optimal. The A* algorithm is also said to be complete, meaning it will find the path in the transition graph, if the path exist. The drawback of this algorithm is the high consumption of memory to store the information for each node.
3 Door State Estimation

This chapter describes the theoretical details of the developed door state estimation algorithm, which is the main focus of this thesis. First, the problem description is given, where challenges of a mobile robot in a household environment related to doors are addressed. Second, the requirements and assumptions are specified to exactly define the framework of the problem. The third section describes the kinematic model of a hinged door. The fourth section follows with an overview of the current state of the art in the field of door state estimation. The next section presents and discusses the potential solutions for the door state estimation problem. In the last section, a detailed description with step-by-step explanation of the developed door state estimation algorithm is given.

3.1 Problem Description

The essential function of a mobile household service robot is to autonomously drive from one place to another. The robot must not only navigate in the unstructured household environment but also avoid collisions with objects in the environment. Typically, all the objects detected by the robot are treated as obstacles and the robot plans its trajectory around them. If the mobile robot is in a room with a closed door and is given a new goal location outside that room, the execution of this task will fail, since there is no collision-free path for the robot to take. Even if the door is slightly open (but not enough to fit a robot through), the robot will still be trapped inside this room. This type of scenario
can often occur in a household environment. A slightly opened door differs from other objects in the environment. Doors are meant to be moved. Allowing the mobile robot to make contact with doors and push them open can prevent the mobile robot of being trapped in the previously described scenario.

But a distinction between a door and other objects in the environment is not trivial for a robot. Typically, a robot perceives its surroundings with sensors that measure distance between the sensor itself and the object in front of it. Laser range finders are well suited for indoor environments and the robot developed within the scope of this thesis is also equipped with one (see Chapter 4.1). This work addresses the challenge of distinguishing doors from obstacles and estimating their state (opening angle) in the household environment in real-time based only on the 2D laser scan readings.

3.2 Requirements and Assumptions

A mobile household robot operates in a wide variety of indoor environments. Our approach of door state estimation is constraint to the 2D laser-based perception, which is not limited to color or texture of a door nor to the varying lightning conditions in the environment. The door state estimation must be executed in real-time meaning the maximum computation time of the door state estimation algorithm is restricted to the time difference between the acquisition of two subsequent laser scan measurements. The algorithm must be robust in the presence of obstacles. For safety reasons, it is crucial that the algorithm does not have false positives meaning an obstacle must not be recognized as a door.

In order to simplify the problem, several assumption are made which are justified later in this section. The assumptions are the following:

- The door is of hinged type.

- The following door parameters are known:
3.2 Requirements and Assumptions

The robot is localized in the environment.

In general, there are several types of doors, illustrated in Figure 3.1, such as hinged doors, sliding doors, folding doors, rotating doors and up-and-over doors (typically found in garages). However, the hinged door type is typically found in the household environment. The door state estimation algorithm developed within the scope of this project focuses on the hinged door and is not directly applicable to other door types, because of differences in the kinematic models. Other door types can be detected using the same approach if the kinematic model is changed accordingly.

The next assumption is that six static door parameters are known. These six parameters are $x_{hinge}$, $y_{hinge}$, $\theta_x$, $\theta_z$, $w$ and $\alpha_{max}$. They are all part of the kinematic model of a hinged door and are constant. All six parameters are set only once since they do not change. The estimation can be either done manually through a graphical user interface or automatically using a method to generate topological maps, for example introduced here [15].

The algorithm is based on 2D laser scan readings. As it is described later, the laser scan readings must be transformed from the sensor coordinate frame into the map coordinate frame. In order to have a correct transformation, the robot must be localized in the environment.
3.3 Kinematic Model of a Hinged Door

As mentioned before, the door state estimation algorithm developed within the scope of this project is limited to the hinged door type. This section describes a hinged door and its kinematic model.

A hinged door consists of a door leaf which is on one side connected to a wall via a revolute joint called a hinge. A hinge is a mechanism that allows rotation around only one axis. This setup enables a door to pivot away from the doorway. Therefore, a hinged door can be modeled as a simple one-degree-of-freedom system consisting of only two links connected via a vertical rotary joint.

The robot’s knowledge about its environment is described in the static map. The map coordinate frame is attached to the static map with x-axis and y-axis.
lying in the horizontal plane and z-axis pointing vertically. The static map is used to navigate the robot in the environment as described in Chapter 4.4. The kinematic model of the hinged door is, like the rest of the environment, described according to the map coordinate frame. Knowing the kinematic model of the door and all its parameters unambiguously defines the door in the environment.

The kinematic model of a hinged door is the following (see also 3.2). The coordinate system of the first link (later referred to as the hinge coordinate frame), which is statically fixed in the environment, is placed at the position of the hinge. The z-axis is aligned vertically pointing upwards if the door pivots in mathematically positive direction or downwards if the door pivots in mathematically negative direction. The x-axis points from the hinge to the other side of the wall just along the side of the door leaf when the door is closed. Hence, the y-axis then points in the direction of the door opening.

![Figure 3.2: Position of coordinate frames (c.f.) in the kinematic model of a hinged door.](image)

The coordinate frame of the second link (later referred to as the leaf coordinate frame) is placed on the side of the door leaf, where the hinge is also located, with the x-axis aligned along the side of the door leaf, the y-axis aligned perpendicular to the door leaf and the z-axis aligned vertically along the axis of rotation.
Figure 3.2 demonstrates how the coordinate frames are positioned.

Transformations between map, hinge and leaf coordinate frames must be defined in order to unambiguously describe a door in the robot’s environment. These transformations are a set of translations and rotations.

A translation from coordinate frame A to coordinate frame B is expressed by translation matrix $T_B^A \in \mathbb{R}^{4 \times 4}$ which is determined with three parameters $d_x$, $d_y$ and $d_z$ that define the displacement along x, y and z-axis, respectively:

$$
T_B^A = \begin{pmatrix}
1 & 0 & 0 & d_x \\
0 & 1 & 0 & d_y \\
0 & 0 & 1 & d_z \\
0 & 0 & 0 & 1
\end{pmatrix} = T_B^A(d_x, d_y, d_z) \quad (3.1)
$$

A rotation that rotates coordinate frame A into coordinate frame B is defined by rotation matrix $R_B^A \in \mathbb{R}^{4 \times 4}$. In general, any rotation can be achieved by composing three elemental rotations [16]. For the purpose of this work, a rotation is defined by a sequence of three extrinsic elemental rotations, first around the x-axis, then around the y-axis and finally around the z-axis. The first rotation is made with the angle $\theta_x$, the second with angle $\theta_y$ and the third with angle $\theta_z$. A rotation matrix $R_B^A \in \mathbb{R}^{4 \times 4}$ that rotates coordinate frame A into coordinate frame B is given as:

$$
R_B^A = R_x(\theta_x) \cdot R_y(\theta_y) \cdot R_z(\theta_z) =
$$

$$
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \theta_x & -\sin \theta_x & 0 \\
0 & \sin \theta_x & \cos \theta_x & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\cos \theta_y & 0 & \sin \theta_y & 0 \\
0 & 1 & 0 & 0 \\
-\sin \theta_y & 0 & \cos \theta_y & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\cos \theta_z & -\sin \theta_z & 0 & 0 \\
\sin \theta_z & \cos \theta_z & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} =
$$

$$
R_B^A(\theta_x, \theta_y, \theta_z) \quad (3.2)
$$
3.3 Kinematic Model of a Hinged Door

An arbitrary point described in the coordinate frame A with the vector form \( \mathbf{p}_A = (x_A, y_A, z_A, 1)^T \) can be transformed into coordinate frame B with vector form \( \mathbf{p}_B = (x_B, y_B, z_B, 1)^T \) according to (3.3). The transformation matrix \( \mathbf{P}_B^A \in \mathbb{R}^{4 \times 4} \) is a product of the rotation matrix \( \mathbf{R}_B^A \) and the translation matrix \( \mathbf{T}_B^A \):

\[
\mathbf{p}_B = \mathbf{P}_B^A \cdot \mathbf{p}_A = \mathbf{R}_B^A \cdot \mathbf{T}_B^A \cdot \mathbf{p}_A
\]

Therefore, an arbitrary point described in map coordinate frame \( \mathbf{p}_{\text{map}} \) can be transformed into the same point described in leaf coordinate frame \( \mathbf{p}_{\text{leaf}} \):

\[
\mathbf{p}_{\text{leaf}} = \mathbf{R}_{\text{hinge}}^{} \cdot \mathbf{T}_{\text{hinge}}^{} \cdot \mathbf{p}_{\text{hinge}} = \mathbf{R}_{\text{hinge}}^{} \cdot \mathbf{T}_{\text{hinge}}^{} \cdot \mathbf{R}_{\text{map}}^{} \cdot \mathbf{T}_{\text{map}}^{} \cdot \mathbf{p}_{\text{map}} = \mathbf{p}_{\text{map}}
\]

The transformation from map to leaf coordinate frame in (3.4) is done with two translations and two rotations each being defined with three parameters. This accumulates in a set of twelve parameters. Some of these parameters are the same for all hinged doors in the environment. The translational and rotational matrices can be simplified as follows.

Because all doors are located on the ground, the \( d_z \) parameter in the translation matrix between map and hinge coordinate frame \( \mathbf{T}_{\text{map}}^{\text{hinge}} \) equals 0 for every door. The translation matrix \( \mathbf{T}_{\text{map}}^{\text{hinge}} \) is then a function of only two parameters, \( x_{\text{hinge}} \) and \( y_{\text{hinge}} \), which describes the position of a hinge in map coordinate frame:

\[
\mathbf{T}_{\text{map}}^{\text{hinge}} = \mathbf{T}_{\text{map}}^{\text{hinge}} \bigg|_{d_z=x_{\text{hinge}},d_y=y_{\text{hinge}},d_z=0} = \mathbf{T}_{\text{map}}^{\text{hinge}}(x_{\text{hinge}}, y_{\text{hinge}})
\]

The rotation matrix between map and hinge coordinate frame \( \mathbf{R}_{\text{map}}^{\text{hinge}} \) also depends on only two parameters \( \theta_x \) and \( \theta_z \). First parameter \( \theta_x \) depends on the direction of door pivoting. If the door pivots in positive direction, \( \theta_z \) is equal to 0. If the
Door pivots in negative direction, $\theta_z$ is equal to $\pi$, which means:

$$\theta_z = \begin{cases} 
0, & \text{if door pivots in positive direction} \\
\pi, & \text{if door pivots in negative direction}
\end{cases} \quad (3.6)$$

The second parameter $\theta_z$ is the orientation of the door when closed in the map coordinate frame:

$$\mathbf{R}_{map}^{hinge} = \mathbf{R}_{map}^{hinge} \bigg|_{\varphi_x=\theta_x, \varphi_y=0, \varphi_z=\theta_z} = \mathbf{R}_{map}^{hinge}(\theta_x, \theta_z) \quad (3.7)$$

The translation matrix between hinge and leaf coordinate frame $\mathbf{T}_{hinge}^{leaf}$ is an identity matrix $\mathbf{I}_{4\times4}$ because the origins of the two coordinate frames coincide:

$$\mathbf{T}_{hinge}^{leaf} = \mathbf{T}_{hinge}^{leaf} \bigg|_{d_x=0, d_y=0, d_z=0} = \mathbf{I}_{4\times4} \quad (3.8)$$

The rotation matrix between hinge and leaf coordinate frame $\mathbf{R}_{hinge}^{leaf}$ is a function of angle $\alpha$, which equals 0 when doors are closed and equals $\alpha_{max}$ when doors are fully open:

$$\mathbf{R}_{hinge}^{leaf} = \mathbf{R}_{hinge}^{leaf} \bigg|_{\varphi_z=0, \varphi_y=0, \varphi_z=\alpha} = \mathbf{R}_{hinge}^{leaf}(\alpha) \quad (3.9)$$

A door leaf is just a rectangular box with height $h$, width $w$ and thickness $d$. The door leaf can be easily described in the leaf coordinate frame. But what is really needed is the ability to transform any arbitrary point of the door leaf to the map coordinate frame. This can be accomplished with (3.10) if parameters $x_{hinge}, y_{hinge}, \theta_x, \theta_z$ and $\alpha$ are known.

$$\mathbf{P}_{map} = (\mathbf{R}_{hinge}^{leaf} \cdot \mathbf{T}_{hinge}^{leaf} \cdot \mathbf{R}_{map}^{hinge} \cdot \mathbf{T}_{map}^{hinge})^{-1} \cdot \mathbf{P}_{leaf} = \mathbf{P}_{leaf}(x_{hinge}, y_{hinge}, \theta_x, \theta_z, \alpha) \cdot \mathbf{P}_{leaf} \quad (3.10)$$

In conclusion, a hinged door, modeled as a rectangular box, is completely defined in the map coordinate frame with eight parameters: $x_{hinge}, y_{hinge}, \theta_x, \theta_z,$
\(\alpha, h, w\) and \(d\). The only parameter that changes with time is the door angle \(\alpha\). The estimation of door angle \(\alpha\) is the main focus of the developed door state estimation algorithm. The other parameters are constant and assumed to be known.

3.4 State of the art

Door state estimation and door crossing are part of fundamental functions for autonomous navigation of a mobile household robot, besides the localization, the optimal path planning and the collision avoidance. This is the reason why a lot of research groups have already addressed this challenge.

An approach based only on vision sensor is presented in [17]. The door is detected by extracting corner features from the image. The method is constrained only to opened doors with colors different from the background. It is unclear how the method scales to doors with different frame shapes or doors having the same color as the wall. The performance also deepens on well lightning conditions.

Another vision based approach is presented in [18]. Their approach imposes no constraint on door aperture, color, texture or state of the door. The algorithm is capable of detecting the door, recognizing the state and estimate whether the door is suitable for crossing. The calculations are based on subsequent filtering and Hough transform search. The approach depends on a large number of thresholds (such as detected lines must take three quarters of the image, lines must have an inclination between 85° and 95°, and lines must be separated by more then 20 pixels). The paper unfortunately presents results only on one door, which color pattern highly improved the Hough transform search. The robot’s position relative to the door is also an important factor because of the camera view angle. It is unclear what happens in narrow corridors when camera position is not optimal.

A more modern approach using convolutional neural network and deep learn-
Door State Estimation

is proposed in [19]. The door state estimation is again based on a RGB camera sensor. The approach is shown to yield satisfactory result, however they do not mention nor demonstrate whether it is able to detect open or half-open doors.

A method based on 3D laser scan data is shown in [20]. The 3D point cloud is clustered into vertical planes. The vertical planes characteristics are compared to expected door dimension. The potential door candidates are later additionally confirmed by extracting the door handle feature. The robot does not only acquire the 3D laser scan but also laser intensity values. The approach has been proven to be very successful on numerous doors. The disadvantage of the approach is the high-cost laser scanner and the large required computational power.

Depth information from a RGB-D camera is used to recognize only open and half-open doors in an approach used in [21] and [22]. They are able to extract the walls as vertical planes from the depth information using RANSAC (Random Sample Consensus) and calculate the door’s opening angle through the shape of the gap inside the wall. An important assumption is made that the camera is placed in front of the door.

Another approach based on RGB-D camera is presented in [23]. They developed a method which enables an Atlas robot to traverse through doors. Their approach is divided in four parts: door detection, walk to the door, door opening and walk through the door. Door detection is made by first finding vertical lines in a 2D image using Canny edge detector and Probabilistic Hough Transform. Second, the detected 2D lines are recomputed in 3D space with RANSAC algorithm. Third, the robot looks for a flat surface between each pair of the lines. If a flat surface is found, the robot recognizes it as a door. This is followed by handle detection, which is done using color segmentation.

An interesting approach related to this work is proposed in [24]. First, a 2D laser scanner is used to find door candidates in the environment. The method assumes that doors are not closed (not aligned with the wall plane). Door candidate is identified as point-cluster which forms a relatively straight line with
specific length. Afterwards, the robot visits each door candidate and scans it with a RGB-D camera. Depth information of each candidate gets analyzed with a door handle detection algorithm to identify true doors.

Another approach based on 2D laser scan data is introduced in [25]. The paper presents a rule-based approach for door detection that works under three assumptions: doors must be open, robot must be positioned near or between the door frame, doors must be aligned with the axis of reference coordinate system. Although the approach is shown to have a success rate of 90%, the required assumptions make it unsuitable for our application.

The presented approaches can be clustered in three categories based on the used sensor: the image based approaches, the 3D data approaches and the 2D data approaches. The image-based techniques have a major drawback. They are susceptible to lightning conditions and are limited to color and texture of doors and walls, which makes them unsuitable for our application. The approaches based on 3D data are also unsuitable due to the sensor’s high cost and the high computation power needed to process the data. None of the presented state-of-the-art approaches focuses on estimating a continuous door opening angle independent of the door’s initial state, which is the goal of our work.

3.5 Approaches

Our challenge is to detect the door and the state of the door (door opening angle $\alpha$) in the environment using only 2D laser-based perception. As described in Section 3.3, by knowing the eight parameters ($x_{hinge}$, $y_{hinge}$, $\theta_x$, $\theta_z$, $\alpha$, $h$, $w$ and $d$) of the door’s kinematic model, the exact pose of the door in the environment is known.

The height of the door $h$ is irrelevant, because only the projection of a door on the static map is really needed for the navigation algorithm. The door is usually thin and the door thickness $d$ can be discarded, because the door projection can
be modeled with a line instead of a rectangle without any significant error.

Five of the remaining parameters ($x_{\text{hinge}}, y_{\text{hinge}}, \theta_x, \theta_z,$ and $w$) are all constant and specific to the door. They can be obtained separately, as described in Section 3.2. The only parameter that changes and needs to be tracked in real time is the door angle $\alpha$. The problem of door state estimation becomes a problem of estimating the door angle $\alpha$.

In the following, three potential solutions are examined and discussed. The first one is based on linear regression [26] and the assumption that the door leaf corresponds to a line in the 2D laser scan readings. The second one uses a multinomial distribution [27] to consider several leaf estimates. The third approach includes additionally information from the pass-through laser beams similar to occupancy grid mapping which is a probabilistic approach of generating a map of the environment based on noisy sensor measurements [28]. All three approaches rely on the pre-clustering of the obtained 2D laser scan readings using only data that lies within the door area. We define the door area as a circle segment defined by its origin, radius and opening angle. The origin is at the position of the hinge $(x_{\text{hinge}}, y_{\text{hinge}})$, the radius equals the door width $w$ and the opening angle equals the maximum door angle $\alpha_{\text{max}}$, see Section 3.3. The presented approaches are discussed using three scenarios, illustrated in Figure 3.3. In the first scenario (Figure 3.3a), the robot is observing a half-open door. In the second scenario (Figure 3.3b), the door remains half-open, but an obstacle is added to the door area. In the third scenario (Figure 3.3c), the door is wide-open while the obstacle remains at the same position in the door area.

The first approach relies on the straightforward principle of linear regression [26]. The linear regression is applied to the 2D laser scan readings that lie inside the door area. A line model $y = a + b \cdot x$ is defined with two parameters $a$ and $b$. The $i$-th laser scan reading can be approximated with the same line model and an additional error term $y^{(i)} = a + b \cdot x^{(i)} + e^{(i)}$. The parameters $a$ and $b$ are
3.5 Approaches

Figure 3.3: Three potential scenarios for sensor, door and obstacle layout.

This approach performs well when applied to the first scenario shown in Figure 3.3a, but it does not perform well in second and third scenario shown in Figure 3.3b and Figure 3.3c respectively. This is due to laser scan readings corresponding to the obstacle which are treated equally as the laser scan readings corresponding to the door leaf. In the second scenario, the door angle would be estimated in the middle of the door leaf and obstacle, while in the third scenario, the door angle would be estimated as if the obstacle is a door leaf.

The second approach uses a multinomial distribution [27]. First, the door area is divided in smaller circle segments or bins $\alpha_{(i)}$ with equal angle step. Second, the extraction of laser scan readings that lie inside the door area is made. Third, the number of the extracted laser scan readings in the individual circle segments $n^{(i)}$ is checked. The extracted laser scan readings categorized to certain bins $\alpha^{(i)}$ are a discrete presentation of the door angle $\alpha$. Assuming the extracted laser scan readings are independent of each other, the set can be modeled with the multinomial distribution [27]. The maximization of likelihood estimation for the multinomial model results in (3.12). The equation computes the probability of door angle $\alpha$ being equal to the mean of the $i$-th bin. The probability is equal to the ratio between the number of laser scan readings in the $i$-th bin $n_i$ and the
number of all the extracted laser scan readings $N = \sum_i n^{(i)}$.

$$P(\alpha = \alpha^{(i)}) = \frac{n^{(i)}}{N} \quad (3.12)$$

In contrast to the first approach, a probability for each bin is calculated, rather than one set of parameters $a$ and $b$ of the line model. The second approach yields suitable results for first and second scenarios shown in Figure 3.3a and Figure 3.3b respectively. But the result is unacceptable for the third scenario shown in Figure 3.3c. There, the door is not observed by the laser scanner and only the laser scan readings corresponding to the obstacle are extracted. The probability of door angle $\alpha$ being around $30^\circ$ increases since the sum of all extracted laser scan readings $N$ is low.

The previous two approaches only use information from laser beams with laser scan readings being inside the door area. But the laser beams that pass through the door area and do not hit any object also carry information about the door angle $\alpha$. The third and final approach uses this information and is inspired by algorithms that build occupancy grids from 3D point-cloud data [28]. Similarly as in the second approach, the door area is divided in equally large bins. Instead of computing probabilities, the real door angle $\alpha$ is estimated based on a voting scheme. The path of each laser beam is checked. If a laser beam fully crosses the bin, the bin gets down-voted as the probability that a door is located in that bin is low. Contrary, the bin containing a laser hit gets up-voted. The count of the laser beams that would potentially cross the bin, but do not due to occlusion, is also made. At the end of this procedure, every bin is associated with three counts. The number of up-votes $n_{\text{up}}$, the number of down-votes $n_{\text{down}}$ and the number of withheld-votes $n_{\text{with}}$ caused by occlusion. Based on these counts, a score $\gamma$ is given to each bin. The score of the $i$-th bin $\gamma^{(i)}$ is computed with (3.13). The bin
with the highest score is then selected as the result.

\[
\gamma^{(i)} = \frac{n_{\text{up}}^{(i)} - n_{\text{down}}^{(i)}}{n_{\text{up}}^{(i)} + n_{\text{down}}^{(i)} + n_{\text{wth}}^{(i)}}
\]  

(3.13)

This approach performs well in all three scenarios presented in Figure 3.3. Even in the most difficult scenario shown in Figure 3.3c, the algorithm does not confuse the obstacle as part of the door. Another advantage of this approach, besides the correct estimation of the door angle \( \alpha \), is that it also provides some measure of confidence \( c \) which equals to the highest score \( \gamma_{\text{max}} \). This additional information can be used by the higher level navigation logic to decide, if the robot is allowed to cross the door area.

Since the third approach deals best with all three scenarios presented in Figure 3.3, it was chosen for implementation and testing. This approach is from now on referred to as the door state estimation algorithm. A detailed description is given in Section 3.6.

### 3.6 Description of the Door State Estimation Algorithm

This section presents a detailed description of the door state estimation algorithm. The algorithm can be summed up into 5 steps, which are executed each time a new 2D laser scan readings are obtained. The steps are the following:

1. Transform sensor measurements into hinge coordinate frame.

2. Compute intersections between laser beams and door area border.

3. Distribute votes among potential door angle candidates.

4. Compute the score of each potential door angle candidate.

5. Select the best potential door angle candidate.
Step 1: Transform sensor measurements into hinge coordinate frame

As mentioned before, the 2D laser scanner provides a set of distance measurements in predefined directions. Typically the measurements are gathered in the horizontal plane with a constant angle step. Knowing how the sensor collects distance measurements, one can easily transform distance measurements into points in space with x, y and z coordinates. The coordinates of these points are expressed in the sensor coordinate frame and are transformed to the hinge coordinate frame for further processing. Figure 3.4 shows the relation between coordinate frames. Since the sensor is fixed to the robot and the robot design is known, the static transformation between sensor coordinate frame and base coordinate frame is easily obtained. The transformation between map and base coordinate frame depends on the robot pose and is provided by the localization algorithm which runs in parallel as described in Chapter 4.4. The transformation between map and hinge coordinate frame is, as mentioned in Section 3.3, defined by static parameters $x_{hinge}, y_{hinge}, \theta_x$ and $\theta_z$ which are assumed to be known. Because this series of transformations is available, laser scan readings can be transformed from sensor to hinge coordinate frame with (3.14).

\[
\mathbf{p}_{hinge} = \mathbf{P}_{hinge}^{map} \cdot (\mathbf{P}_{map}^{sensor})^{-1} \cdot \mathbf{p}_{sensor} \tag{3.14}
\]

Figure 3.4: Relation between coordinate frames.
Step 2: Compute intersections between laser beams and door area border

In the second step of the door state estimation algorithm, the intersections between laser beams and door area border are computed. The computed intersection will be used in the next step to evaluate which bins are crossed by the laser beams. First, we define the door area and laser beams.

Geometrically speaking a laser beam is a ray. A ray, shown in Figure 3.5a, is a line starting in the first point and continuing indefinitely through the second point. The first point is the start of the laser beam and it is located in the origin of the sensor coordinate frame, where the laser beam is emitted. The second point is the laser hit, where a laser beam hits an object and reflects back. The laser hit (see Figure 3.6) of course corresponds to previously computed point from the distance measurements that was transformed into hinge coordinate frame.

![Ray](image1.png)

![Circle segment](image2.png)

(a) Ray. (b) Circle segment.

Figure 3.5: Illustration of a ray and a circle segment.

A door area is circle segment, where the door leaf is expected to be found. A circle segment, shown in Figure 3.5b, is defined with few parameters: circle origin, circle radius $r$, start angle $\beta_{\text{start}}$ and stop angle $\beta_{\text{stop}}$. The circle segment defining the door area has its origin at the position of the hinge and its radius $r$ is equal to door width $w$. As can be seen in Figure 3.6, ideally $\beta_{\text{start}} = 0$ and $\beta_{\text{stop}} = \alpha_{\text{max}}$ resemble the opening angle of the door. However, due to noise in the laser scanner data and the robot localization, we increase the door area by an angle $\alpha_{\text{tol}}$ on both sides.
The border of the door area is made out of three simple geometric parts, two line segments and an arc. The problem of calculating the laser beams intersections with the door area border can be seen as a problem of calculating the intersections between a ray (laser beam) with two line segments and an arc (door area border).

There are three different ways how a laser beam crosses the door area. All of them are illustrated in Figure 3.7. The laser beams are drawn as red lines while the intersections with door area border are marked with with yellow circles. In the first case (A), the laser beam does not cross the door area, which means the number of calculated intersections between laser beam and door area border is zero. In the second case (B), only one intersection is found, because the laser beam only touches the door area border. In the third case (C), the laser beam goes through the door area, which means two intersections are found.

For each laser beam with two intersections, two additional angles $\alpha_1$ and $\alpha_2$ are computed. These two angles, later referred to as intersection angles $\alpha_1$ and $\alpha_2$, are angles between the x-axis of hinge coordinate frame and the line that goes
through the hinge and the corresponding intersection, as shown in Figure 3.7.
The intersection angles $\alpha_1$ and $\alpha_2$ are sorted such that $\alpha_1 < \alpha_2$.

Figure 3.7: Illustration of three different ways of laser beam crossing the door area.

To sum up, in this step of the algorithm the number and exact location of intersections between laser beams and door area border are computed and stored for further processing.

**Step 3: Distribute votes among potential door angle candidates**

The real door angle $\alpha$ is a continuous function in the interval $\alpha \in [0, \alpha_{max}]$. As mentioned before, the interval of the estimated door angle $\alpha_{est}$ is extended to the interval $\alpha_{est} \in [-\alpha_{tol}, \alpha_{max} + \alpha_{tol}]$ because of the noise presence. This interval is also further divided into bins with equal size $\alpha_{step}$. The estimated door angle $\alpha_{est}$ has a discrete distribution and can take only values that equal to the means of the individual bins, see Figure 3.8. These values are called the potential door angle
Figure 3.8: The discretization of the door area into seven bins ($n_{bin} = 7$) with equal size ($\alpha_{step} = 20^\circ$).

candidates $\alpha_{pdac}$. This discretization leads to the door area being divided into smaller circle segments. If the number of bins equals $n_{bins}$, then the bin size $\alpha_{step}$ can be obtained with (3.15).

$$\alpha_{step} = \frac{\alpha_{max} + 2 \cdot \alpha_{tol}}{n_{bins}}$$  \hfill (3.15)

The bin size $\alpha_{step}$ determines the resolution of door state estimation algorithm. In other words, it is the minimum possible change of the real door angle $\alpha$ that can be detected by the algorithm.

The estimated door angle $\alpha_{est}$ is obtained based on the voting scheme. Each laser beam distributes a variable number of votes, depending on how it crosses the door area. Laser beams that have zero or one intersection with the door area border, do not cross the door area and do not carry any information about the door angle $\alpha$. These laser beams do not distribute any votes and are ignored. Only laser beams with two intersections carry information about the door angle $\alpha$. A laser beam can vote upon the $i$-th potential door angle candidate $\alpha_{pdac}^{(i)}$ if this laser beam crosses the $i$-th bin, which is determined with (3.16). The more bins
that a laser beam crosses, the more votes it can distribute.

$$\alpha_{1}^{(i)} \leq \alpha_{pdac}^{(i)} \leq \alpha_{2}^{(i)} \quad (3.16)$$

A laser beam can either give an up-vote, a down-vote or it can even withhold its vote. Three variables for each bin are being tracked, the number of up-votes \(n_{up}\), the number of down-votes \(n_{down}\) and the number of withheld-votes \(n_{wth}\). The voting action depends on the position of the laser hit with regards to the computed intersections and sensor origin. An up-vote is given to the \(i\)-th potential door angle candidate \(\alpha_{pdac}^{(i)}\) if the laser hit lies within the \(i\)-th bin. All bins that are located towards the sensor origin are down-voted and the vote is withheld for all bins that are further away from the sensor origin. To determine the voting action the laser hit angle \(\alpha_{hit}\) and the sensor origin angle \(\alpha_{org}\) are computed. An example of a laser hit angle \(\alpha_{hit}\) and a sensor origin angle \(\alpha_{org}\) can be observed in Figure 3.9a.

Algorithm 1 shows the pseudo code for the process of vote distribution. Notation is explained in Table 3.1. First, the number of up-votes \(n_{up}\), down-votes \(n_{down}\) and withheld-votes \(n_{wth}\) are initialized to zero for all bins. Then, the algorithm loops through each laser beam and distributes the votes to the potential door angle candidates.
Algorithm 1 Votes Distribution

\begin{algorithm}
\begin{algorithmic}
\For{$i \leftarrow [1, n_{\text{bins}}]$}
\State $n_{\text{up}}^{(i)} \leftarrow 0$
\State $n_{\text{down}}^{(i)} \leftarrow 0$
\State $n_{\text{wth}}^{(i)} \leftarrow 0$
\EndFor

\For{$j \leftarrow [1, n_{\text{beams}}]$}
\For{$i \leftarrow [1, n_{\text{bins}}]$}
\If{$\alpha_{1}^{(j)} \leq \alpha_{\text{pdac}}^{(i)} \leq \alpha_{2}^{(j)}$}
\If{$\alpha_{\text{org}} \leq \alpha_{\text{pdac}}^{(i)} < \alpha_{\text{hit}}^{(j)} - \frac{1}{2} \alpha_{\text{step}}$}
\State $n_{\text{down}}^{(i)} \leftarrow n_{\text{down}}^{(i)} + 1$
\ElseIf{$\alpha_{\text{hit}}^{(j)} + \frac{1}{2} \alpha_{\text{step}} \leq \alpha_{\text{pdac}}^{(i)} \leq \alpha_{\text{hit}}^{(j)} + \frac{1}{2} \alpha_{\text{step}}$}
\State $n_{\text{up}}^{(i)} \leftarrow n_{\text{up}}^{(i)} + 1$
\Else
\State $\alpha_{\text{hit}}^{(j)} + \frac{1}{2} \alpha_{\text{step}} \leq \alpha_{\text{pdac}}^{(i)}$
\State $n_{\text{wth}}^{(i)} \leftarrow n_{\text{wth}}^{(i)} + 1$
\EndIf
\EndIf
\EndIf
\EndFor
\EndFor
\end{algorithmic}
\end{algorithm}

Table 3.1: Explanation of notation used in Algorithm 1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_{\text{beams}}$</td>
<td>number of laser beams emitted from sensor</td>
</tr>
<tr>
<td>$n_{\text{bins}}$</td>
<td>number of bins</td>
</tr>
<tr>
<td>$n_{\text{up}}^{(i)}$</td>
<td>number of up-votes for $i$-th bin</td>
</tr>
<tr>
<td>$n_{\text{down}}^{(i)}$</td>
<td>number of down-votes for $i$-th bin</td>
</tr>
<tr>
<td>$n_{\text{wth}}^{(i)}$</td>
<td>number of withheld votes for $i$-th bin</td>
</tr>
<tr>
<td>$\alpha_{\text{pdac}}^{(i)}$</td>
<td>angle of $i$-th potential door angle candidate</td>
</tr>
<tr>
<td>$\alpha_{\text{step}}$</td>
<td>the size of the bins</td>
</tr>
<tr>
<td>$\alpha_{\text{org}}$</td>
<td>the sensor origin angle</td>
</tr>
<tr>
<td>$\alpha_{1}^{(j)}$</td>
<td>angle of laser hit of the $j$-th laser beam</td>
</tr>
<tr>
<td>$\alpha_{1}^{(j)}$</td>
<td>angle of the first intersection between $j$-th laser beam and door area border</td>
</tr>
<tr>
<td>$\alpha_{2}^{(j)}$</td>
<td>angle of the second intersection between $j$-th laser beam and door area border</td>
</tr>
</tbody>
</table>
To fully illustrate the vote distribution, three different exemplary scenarios are analyzed here. In the first scenario shown in Figure 3.9, the laser hit occurs before the door area, in the second scenario shown in Figure 3.10, the laser hit occurs inside the door area, and in the third scenario shown in Figure 3.11, the laser hit occurs after the door area. The door area is in all three scenarios divided into seven bins \( n_{\text{bin}} = 7 \) with equal size \( \alpha_{\text{step}} = 20^\circ \). The individual bins are colored according to the type of vote they receive. The up-voted bins are colored with dark-green, the down-voted bins are colored with red and the bins with withheld votes are colored with blue.

In the first scenario, presented in Figure 3.9a, an obstacle is located between the sensor and the door area, consequentially the laser hit occurs before the door area, \( \alpha_{\text{hit}} < \alpha_1 \). The ray of the laser beam passes the 4-th, 5-th and 6-th bin. But the obstacle occludes the bins, therefore the votes are withheld. Figure 3.9b shows how the laser beam distributes its votes among the seven potential door angle candidates \( \alpha_{\text{pdac}} \). The light-green bins represent the potential door angle candidates that were not passed through and were not give any vote by the laser beam, while the blue bins represent potential door angle candidates where number of withheld votes \( n_{\text{wth}} \) is incremented by one.

In the second scenario, presented in Figure 3.10a, the laser beam hits an object (in this case a door leaf) inside the door area. The angle of laser hit \( \alpha_{\text{hit}} \) is in between angles of first and second intersection, \( \alpha_{\text{hit}} \in [\alpha_1, \alpha_2] \). The 4-th bin receives a down-vote, because it was passed through by the laser beam without any collision. The 5-th bin receives an up-vote, because a laser hit occurs in that circle segment. The vote on the 6-th bin is withheld, because the laser beam is occluded before it actually passes through the bin. Figure 3.10b shows the final distribution of votes from the analyzed laser beam.
Door State Estimation

(a) Layout of the scenario.

(b) Distribution of votes for the observed laser beam.

Figure 3.9: Vote distribution when a laser hit occurs before the door area.

(a) Layout of the scenario.

(b) Distribution of votes for the observed laser beam.

Figure 3.10: Vote distribution when a laser hit occurs inside the door area.
3.6 Description of the Door State Estimation Algorithm

Figure 3.11: Vote distribution when a laser hit occurs after the door area.

In the third scenario, presented in Figure 3.11a, the laser hit occurs after the door area, $\alpha_{hit} > \alpha_2$. The laser beam freely passed through 4-th, 5-th and 6-th bin. A down-vote is given to this three bins. The red color indicates the bins that receive a down-vote. Figure 3.11b shows the final distribution of votes among the potential door angle candidates from the analyzed laser beam.

**Step 4: Compute the score of each potential door angle candidate**

After all votes are distributed from the laser beams, a score $\gamma_{raw}$ for every potential door angle candidate $\alpha_{pdac}$ is computed. The score of the $i$-th potential door angle candidate $\gamma_{raw}^{(i)}$ is the ratio between the difference of up-votes and down-votes, and the sum of all the votes given to the $i$-th potential door angle candidate $\alpha_{pdac}^{(i)}$. The score for $i$-th potential door angle candidate $\alpha_{pdac}^{(i)}$ is
computed with (3.17).

$$\gamma_{\text{raw}}^{(i)} = \frac{n_{\text{up}}^{(i)} - n_{\text{down}}^{(i)}}{n_{\text{up}}^{(i)} + n_{\text{down}}^{(i)} + n_{\text{wth}}^{(i)}}$$  \hspace{1cm} (3.17)

Because of the selected vote distribution, the score $\gamma_{\text{raw}}^{(i)}$ is limited to the interval $[-1, 1]$. The potential door angle candidates with higher scores are closer to the real door angle $\alpha$.

The scores $\gamma_{\text{raw}}^{(i)}$ are computed for every new laser scan that arrives from the sensor. Since noise, present in the sensor measurements, has a higher frequency than the real door angle $\alpha$, we add a low-pass filter. In this work, a simple infinite impulse response (IIR) filter is used. The filtered scores $\gamma_{\text{fil}}^{(i)}$ are obtained with (3.18), where the constant $K_{\text{fil}}$ depends on the sampling rate of the laser scanner. The $k$ presents the current discrete time instance and $k - 1$ presents the previous discrete time instance.

$$\gamma_{\text{fil}}^{(i)}(k) = (1 - K_{\text{fil}}) \cdot \gamma_{\text{fil}}^{(i)}(k - 1) + K_{\text{fil}} \cdot \gamma_{\text{raw}}^{(i)}(k)$$  \hspace{1cm} (3.18)

**Step 5: Select the best potential door angle candidate**

The last step of the algorithm is to extract the best potential door angle candidate and output the result in the form of estimated door angle $\alpha_{\text{est}}$ and level of confidence in that estimation $c_{\text{est}}$. The algorithm first calculates the index of the maximal score $i_{\text{max}}$, see (3.19). Second, the estimated door angle $\alpha_{\text{est}}$ is determined as the potential door angle candidate with index $i_{\text{max}}$, see (3.20). Third, the confidence level $c_{\text{est}}$ is determined as the filtered score with index $i_{\text{max}}$, see (3.21). The confidence level is is $c_{\text{est}} \in [0, 1]$, where 1 presents the highest level of confidence.

$$i_{\text{max}} = \arg \max_i \gamma_{\text{fil}}^{(i)}$$  \hspace{1cm} (3.19)

$$\alpha_{\text{est}} = \alpha_{\text{pdac}}^{(i_{\text{max}})}$$  \hspace{1cm} (3.20)

$$c_{\text{est}} = \gamma_{\text{fil}}^{(i_{\text{max}})}$$  \hspace{1cm} (3.21)
4 Implementation

In order to evaluate environment mapping, autonomous navigation and door state estimation algorithms, software must be implemented and tested on a robot. For the purpose of this project, an actual robot was designed and built. First, the hardware components and their position in the system must be determined according to application’s requirements. After the robot design is completed, a robot can be built either in real life or in simulation environment. Finally, the developed algorithms can be tested and evaluated for the required functionality.

This chapter contains description of the practical work that was carried out within the scope of this thesis. First, it describes the robot design and hardware components that are used. Second, a short description and explanation of software framework choices are given. This is followed by a brief description and justification of the robot simulator used in this project. Next, a general overview of the software architecture for navigation is given, where the relevant software components for navigation are described in detail. At the end of this chapter, a brief description of the environment mapping is also found. Although building a static map of the environment is not part of the active navigation task, it is an important prerequisite for it.

4.1 Robot Hardware

The first step in robot design is to identify specific requirements and the purpose of the construction. A basic list of requirements for the household service robot
used within this thesis are:

- Robot must be able to move on a flat surface.
- Robot must be equipped with sensors to sufficiently sense the surrounding environment in order to prevent collisions.
- Robot must be able to carry a payload of 10 kg.
- Robot’s dimensions must allow an agile movement inside an apartment.
- Robot must have a low level teleoperation interface.

Because the household service robot is at this point still in a very early stage of development, the designed system serves as a proof of concept and is not a product ready for market. The focus is not on designing a user friendly and low-cost robot, but rather on integrating the emerging technologies and achieving the required functionalities. It is appropriate to use off-the-shelf components to save time and decrease complexity in the development phase, despite the higher costs.

In this specific case, the Robotino Premium Edition from Festo [29] was chosen as mobile platform for the robot. As it is advertised by Festo, the Robotino is a robot mobile platform for research and training. It features an omni-directional drive with three independently driven Omni wheels. The platform is able to drive forwards, backwards and sideways, or even rotate on the spot. It can reach speeds up to 10 km/h. The stainless steel structure provides support for a payload up to 30 kg. The platform is packed in a housing with diameter of 450 mm. The Robotino is also equipped with multiple sensors such as encoders, gyroscope, camera, bumper, two opto-electronic sensors, several infrared sensors and an inductive sensor. Additionally, there is an embedded PC with an Intel i5 dual-core 2.4 GHz processor that provides the necessary computational power. There are many APIs available for programming in C/C++, Java, LabVIEW, MATLAB/Simulink or ROS. The whole system is powered with rechargeable 12 V batteries, which permit 4 hours of running time. Selecting this expensive
Robotino system as the mobile platform can be justified, since the main purpose of the project is to solve problems in autonomous indoor navigation and not to develop a high performance mobile platform at low cost for consumers, as mentioned before.

Although there are many sensors on the Robotino by default, a 2D laser scanner RPLIDAR A1 by Slamtec [30] was added to the robot. The RPLIDAR A1 by Slamtec, later referred to as Rplidar, is a low-cost laser scanner with scanning angle of 360° with angular resolution of 1°. One scan cycle consists of 360 distance measurements that have the maximum range of 6 m. The sampling frequency of the laser scans is up to 10 Hz. The device features plug and play functionality and it can be connected to a computer via micro USB cable. The drivers for ROS are also available [31].

In general, laser scanners such as the Rplidar cannot detect all surfaces. The measurement is based on detecting emitted laser beams reflected from objects. For example, transparent glass surfaces let the laser beams through, which means they are invisible to the laser scanner. During the testing of the robot, it was experienced that also very dark surfaces present a problem for a laser scanner. The laser beam is absorbed by the dark material and since there is no reflection, the measurement fails. The problematic transparent and dark surfaces, and the fact that the Rplidar only scans the horizontal plane are the reasons that four ultrasonic sensors SRF08 [32] were added in front of the robot for additional safety. These ultrasonic sensors have a sensing angle of 55° and a configurable distance range up to 11 m. These sensors provide the system with additional information about the environment in front of the robot. The measurements are not used for high level navigation logic but only to check whether the path is truly collision-free for generated movement commands before they are sent to the actuators. The ultrasonic sensors are connected to the system through a Arduino Mega single-board microcontroller [33].

Although the household service robot should be an autonomous system ca-
pable of navigating on its own, it is necessary to include some teleoperation functionality in order to manually control the robot. An interface with a wireless Logitech F710 gamepad [34] was added for this reason.

Some additional hardware is present on the robot such as a capacitive touch sensor, a tablet, RGB and depth cameras, LED stripes, a microphone and a speaker. But since this additional hardware is not used by the navigation algorithms presented in this work, their detailed description is omitted.

Figure 4.1 shows the internal structure of the designed household service robot with all the integrated hardware components. Sensors are attached to the robots structure with the mounting adapters which were designed for the purpose of this project and were 3D printed. The robot also has an emergency stop button which allows the user to shut down the robot at any given time.
4.2 Robot Software

The hardware setup must be supported by an appropriate software framework. Robot Operation System, also called ROS [35], was chosen to serve this purpose. ROS is a popular open source software framework for programming robots. ROS, being a middleware, also provides communication infrastructure with message passing, remote procedure calls and a distributed parameter system. The framework also includes some robot-specific features such as standard robot messages [36], robot geometry library [37], robot description language [38] and plenty of well tested packages for robot localization [39][40], environment mapping [41] and robot navigation [42]. ROS also offers a powerful development toolset that supports introspecting, debugging, and visualizing the state of the system that is being developed. The framework is particularly useful because it provides an hardware abstraction layer, which means that developed robotic algorithms do not depend on specific hardware and can be easily reused in other robotic application.

All this features made ROS gain its popularity and increased its community members over the last years [43]. It was started as a project at the Stanford University and was later taken over by a robotics incubator Willow Garage in 2007. The software is developed under the permissive BSD open-source license. Different ROS distributions are released in relation to Linux version. For the purpose of this project, the ROS Kinetic Kame distribution [44] was used, which is also the current recommended distribution by ROS. ROS Kinetic Kame distribution was released in May 2016 as a LTS (Long Term Support) distribution. As mentioned before, ROS distribution is akin to Linux distribution, which is the reason why Ubuntu 16.04 is the underlying operating system on the robot.

Choosing ROS was a great fit for developing a mobile household service robot within this project. The provided infrastructure, the available hardware drivers and the modular packages for navigation significantly reduced the complexity of the software development. Instead of solving a variety of smaller technical prob-
lems, the attention was focused more on high-level problems and the development of new algorithms.

4.3 Robot Simulator

During the implementation of the algorithms, one should constantly test and validate them in a controlled environment. A simulator can provide such environment, where a virtual world is designed to suit the needs of the project, and where the developers have an absolute access and control over all environmental variables and parameters. Using a simulator is not the goal of the robot’s development, but rather the first step to make algorithms work on a real robot. The reason the implementation of algorithms must be first verified in the controlled environment inside a simulator is to eliminate the logical errors in the implementation before the tests are made on the real robot.

The advantage of using a simulator is not only the controlled environment, but also elimination of risk to create a physical damage to the robot, environment or even a human. Programming directly on the real robot can be a difficult and tricky task. The simulator enables faster, cheaper and simpler development process.

The disadvantage of a simulator is that it does not perfectly reflect the real world. Certain effects, like friction, can only be estimated and approximated in the simulator. The true verification of the algorithm’s implementation must be done in the real-world environment with real robot and simulation only serves as a tool for faster and more cost-effective development process.

A simulator was also used within the scope of this thesis. Gazebo [45][46] was chosen as the robot simulator. As it is advertised on the official website, Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of
fidelity, a suite of sensors, and interfaces for other programs. Another important advantage of Gazebo is that it can also work under the ROS environment. This is enabled by Gazebo plugins, which are part of the `gazebo` package provided by ROS [47].

The robot is described using the Unified Robot Description Format (URDF) [48], which is a standard XML file format for robot representation in ROS. The URDF file describes every element of the robot including the kinematics, dynamics and sensors. To use the URDF file in Gazebo, some additional simulation-specific tags are added. The simulation of the platform movement is realized by the use of the `Move Planar` plugin [49]. The Rplidar laser scanner is simulated using the `GPU Laser` plugin [50]. This plugin enables the user to set laser scanner specific parameters. The range of the simulated Rplidar is set between 0.15 m and 6.0 m with 5 mm resolution and with Gaussian distribution of noise with mean of 0.0 mm and standard deviation of 20 mm.

### 4.4 Software Architecture for Navigation

The main purpose of robot navigation is to drive autonomously from robot’s current location to a user defined goal location. Within the scope of this thesis, additional requirements are present:

- The navigation software must be independent of the specific underlying hardware down to the layer with hardware drivers.

- The higher level algorithms must only rely on data from 2D laser scan and odometry.

- The actions produced by the navigation software must not result in robot’s collision with obstacles in the environment.

- The navigation software must offer a convenient user interface to set the goal location.
Figure 4.2 shows the software architecture for the autonomous navigation of the developed mobile robot. All the software components needed for navigation run in parallel. Appropriate interconnectivity and message flow are ensured in order to achieve a system capable of completing navigation tasks.

Figure 4.2: Software architecture for navigation for the household service robot comprised of four levels: hardware (HW) drivers, low level signal processing (SP), system components and high level commands (HL).

In the following, the four levels of the software architecture are presented, starting with the lowest level of hardware drivers.
4.4.1 Hardware Drivers

In the lowest layer hardware drivers are found, which are responsible to either transform the collected sensors readings into appropriate ROS format or to send generated commands from ROS to the actuators in a valid form. This layer is of course hardware dependent and it enables the connection between hardware and ROS framework through a predefined interface. Hardware drivers for Rplidar and the sensors present on the Robotino were provided by the associated manufacturers, while the drivers for the SRF08 ultrasonic sensors were developed within the scope of this project.

4.4.2 Low Level Signal Processing

The next layer includes the low level signal processing, which improves the quality of data or double checks commands before they are sent to the hardware components.

The Laser Filter block contains two sub-filters, an angular bounds filter and a shadow filter, which remove certain laser readings from the laser scan. This is needed, because the Rplidar is mounted inside the robot and it has a 360° view, which means some laser readings correspond to the interior of the robot. These laser readings do not hold any information about the environment and they are cropped with the angular bounds filter. The shadow filter removes laser readings that are caused by the veiling effect when the edge of an object is being scanned. This block uses the laser_filter package provided by ROS [51].

The IMU Odom Fusion block improves the odometry data. Odometry information is obtained by tracking the encoders ticks and knowing the kinematic model of the robot. But the wheels of the robot can slip, which leads to an accumulation of error in the odometry data. An inertial measurement unit (IMU) is not prone to slipping of the wheels and can improve odometry information that is only based on encoders readings. This block does exactly that. The information
from encoders and IMU is combined by using a Kalman filter.

The **Collision Avoidance** block is a safety check component. All the velocity commands from the system go through the **Collision Avoidance** block before they are forwarded to the mobile platform with actual motors. This block takes in several sensor readings, in this case a laser scan and ultrasonic measurements and velocity commands generated by the system components, and tries to predict the outcome of the executed velocity commands. If the prediction does not result in a collision, the velocity commands are simply forwarded to the motors. Otherwise, the input velocity commands are ignored and motors are provided with commands for maximum deceleration of the platform. Because the **Collision Avoidance** block is an important safety block, the code inside is kept simple and short in order to avoid bugs. The block can be extended with additional sensors such as cliff sensors or depth cameras.

### 4.4.3 System Components

In the system components layer of Figure 4.2, the most complex blocks can be found. A major part of this work was focused on the blocks **Localization**, **Path Planning** and **Door State Estimation**. The implementation of these blocks are described in more detail later in this section. There is another block in this layer. The **TF** block, which is short for Transformations, keeps track of all the coordinate systems and the transformations between them. This block is part of the standard ROS framework, called the *tf* package [37]. It is capable of managing transform data for robots that have more than one hundred degrees of freedom with an update frequency of hundreds of Hertz. The *tf* package significantly simplifies the transformation of data points from sensors to appropriate coordinate frames.
Localization

Localization of the household service robot is done by using the amcl package provided by ROS [39]. This package provides a probabilistic localization of a robot on a 2D surface. It exploits the Adaptive Monte Carlo Localization (AMCL) approach, which uses a particle filter to estimate the pose of the robot in a known map.

The map must be in the form of an occupancy grid, which is a message type defined in the ROS framework as nav_msgs/OccupancyGrid.msg. The map is obtained through a rosservice call.

The amcl node is provided with the laser scan data and the odometry information. The node estimates the transformation between robot’s base coordinate and the map coordinate frame. The transformation is appropriately registered and updated with the TF block from where the rest of the ROS nodes can access it.

The performance of the amcl node can be configured through a set of available parameters. With these parameters, one can specify for example the minimal and maximal number of particles, update and resample intervals, initial pose estimation, initial covariance of the pose, odometry type, expected noise in odometry and many more. A detailed description of all parameters can be found on the official website of the amcl package.

Path Planning

The block Path Planning presents a major part of the autonomous system. The block reads in the filtered laser scan readings, the current robot pose and the desired goal location, and produces velocity commands that drive the robot towards the desired goal location. The current pose is accessible through the

\[^1\text{http://docs.ros.org/api/nav_msgs/html/msg/OccupancyGrid.html}\]

\[^2\text{http://wiki.ros.org/amcl}\]
TF block, which is updated by the amcl node. The goal pose is provided by High Level, discussed in Section 4.4.4. The implementation uses the move_base package provided by ROS [1].

Figure 4.3 presents the internal structure of the move_base node. As it can be seen, the node itself is logically divided into five parts: the global_costmap, the global_planner, the local_costmap, the local_planner and the recovery_behaviors. The global_costmap and local_costmap parts produce costmaps based on information from a known static map and laser scan readings. A costmap is a 2-D cell grid, where the value of each cell presents a cost of the robot visiting that cell. In general, cells that are closer to obstacles have higher costs. The global_planner and local_planner then calculate the path, which has the lowest combined cost in their corresponding costmaps. The global_planner finds the cheapest path from the current robot pose to the desired goal pose. It connects the neighboring cells and uses the A* algorithm to find the optimal transitions between the cells. The sequence of cells with optimal transition is given to the local_planner, which task it is to generate velocity commands that will move the actual robot trajectory along the path (or close to) computed by the global_planner. This is done by performing a forward simulation of several trajectories that would result from a
discretely sampled robot’s control space. These trajectories are evaluated according to chosen criteria such as proximity to obstacles, proximity to the global path, proximity to the goal and speed. The trajectory with the best score is selected and the associated velocity commands are sent out to the Collision Avoidance block. The move_base node also provides recovery actions when the robot perceives itself as stuck. The actions are defined in the recovery_behaviors part. Recovery behaviors are a sequence of actions such as clearing out the obstacles from the costmaps and performing an in-place rotation of the robot. Detailed description about the recovery behaviors can be found on the ROS website\textsuperscript{3}.

A long list of parameters can be configured in the move_base node. Each part of the node can be strongly modified and adjusted to the user’s needs only by setting appropriate values of this parameters. The detailed and up to date documentation is again available on the website\textsuperscript{3}.

**Door State Estimation**

The block **Door State Estimation** in Figure 4.2 is responsible for estimating the opening angle of a door. This block consists of a single ROS node that was developed from scratch for the purpose of this work. The node receives filtered laser scan readings. Each new laser scan is processed and an estimation of the door angle is made. The results are registered with the TF block, where the kinematic model of the door is then available to the rest of the nodes in the framework. The logic behind the **Door State Estimation** node follows the theory described in Chapter 3. The node loads the static door parameters from a yaml file and initializes itself accordingly. Multiple doors can be detected in parallel. If the yaml file has multiple door entries with associated parameters, then the node will automatically initialize multiple door detectors and keep track of individual door angles. There is an additional parameter which enables or disables the visualization of the algorithm. The visualization shows several figures which in real-time

\textsuperscript{3}http://wiki.ros.org/move_base
display the state of the door state estimation algorithm. The visualization mode can be useful when introspecting or debugging the algorithm. Visualization can be disabled if necessary, because it uses additional computational resources and slows down the node execution.

4.4.4 High Level

The highest layer in Figure 4.2 includes the logic for high level commands and the user interface. This level interacts with the user, for example, it enables the user to set the desired goal location for the robot. The interaction with the user is realized with several interfaces, such as a browser based app, RViz (a tool provided by ROS [52]), command line tools and even voice recognition and control. For navigation itself, it is only important that a goal location is determined, but not how it is obtained, which is why the user interface is not described in detail.

4.5 Software Architecture for Mapping

Both the amcl node in Localization block and the move_base node in the Path Planning block from Figure 4.2 require a static map of the environment. Before a robot is given a navigation task, it must be provided with the static map of the environment. Although mapping of the environment is not part of navigation itself, it is a necessary step prior to the navigation. This section briefly presents the software architecture blocks for environment mapping, summarized in Figure 4.4.

Most of the blocks are same as in the software architecture for navigation shown in Figure 4.2. The lowest layer is extended by another hardware driver for the gamepad. The layer with low level signal processing is exactly the same. The system components layer also includes the TF block that was already described before. New blocks are Remote Control, Mapping and Map Saver.
4.5 Software Architecture for Mapping

Figure 4.4: Software architecture for environment mapping using the household service robot comprised of three levels: hardware (HW) drivers, low level signal processing (SP) and system components.

The Remote Control block enables the user to manually teleoperate the robot. It takes in control commands from the gamepad and generates appropriate velocity commands for the robot. The output is sent to the Collision Avoidance block, which increases the safety of teleoperating a robot in manual mode.

The Mapping block receives the filtered laser scan readings and odometry information. Based on that data, it creates a map of the environment. In this block, one can find a slam_gmapping node from the gmapping package that is included in the ROS framework [41]. This package provides laser-based SLAM approach which creates a 2-D occupancy grid map from pose and laser data collected by a mobile robot. The package is a ROS wrapper for the GMapping library [10][11].

The Map Saver block retrieves a map of the environment from the Mapping block by a rosservice call. The map is stored in a file, where it can be accessed
by the localization and path planning software for autonomous navigation.

To obtain the environment map, the user must run the mapping software on the robot and then drive the robot around the environment manually in tele-operation mode. While the robot is driving around and collecting data, it is also simultaneously updating the map. Observing the same parts of the environment in the laser scan more than once, usually results in better mapping. The user should teleoperate the robot until he/she is satisfied with the generated map.
5 Validation

In order to test and validate the work done within the scope of this project, experiments were carried out with the robot in both simulation and real-world environment. This chapter presents results obtained during these experiments. In the first section of this chapter a description of the simulation and the real-world environment is given to present the environment conditions where the testing took place. The second section discusses the results from environment mapping presented in Chapter 4.5. The third section evaluates the results from testing the navigation software described in Chapter 4.4. The fourth and the last section is dedicated to testing the developed door state estimation algorithm described in Chapter 3.6. The performance of the door state estimation algorithm is evaluated with six different experiments, where each focuses on a certain aspect or functionality. A detailed description and obtained results from this experiments are presented in their own subsections.

5.1 Environment Details

The work of this thesis is evaluated in the robot simulator and in the real-world. This section gives a brief overview of the test environments. It describes the layout of the simulation and the real-world environment.
Simulation

The environment in simulation is designed to evaluate the developed door state estimation algorithm. Therefore, the environment includes two equal rooms, which are separated by a wall with a hinged door. The floorplan of the environment can be observed in the Figure 5.1a. The walls of the environment are designed using the Simulation Description Format (SDF) [53], and the door is designed using the Unified Robot Description Format (URDF) [48]. Figure 5.1b shows a rendered image of the simulation environment visualized with Gazebo simulator, which is described in Chapter 4.3. The parameters of the door used by the door state estimation algorithm are the following:

\[ x_{hinge} = 5.0 \text{ m} \quad \theta_x = 0^\circ \quad w = 1.0 \text{ m} \]
\[ y_{hinge} = 3.0 \text{ m} \quad \theta_z = -90^\circ \quad \alpha_{max} = 165^\circ \]

![Floorplan of the simulation environment.](image1)
![Simulation environment visualized in Gazebo simulator.](image2)

Figure 5.1: Setup of the simulation environment.

Real-world

The experiments in the real-world environment are carried out in an office building. The real-world environment includes a long hallway and an office room. The
hallway and the room are separated by a hinged door. The environment also includes desks, chairs and other furniture similar to a household environment. The layout is suitable to test all the work implemented and described in Chapter 4. Unfortunately, the floorplan of the building is not available, but the described real-world environment can be visualized with the static map obtained by the robot. The static map is shown in Figure 5.3. The parameters of the door used by the door state estimation algorithm are the following:

\[
\begin{align*}
    x_{\text{hinge}} &= 5.87 \text{ m} & \theta_x &= 0^\circ & w &= 0.98 \text{ m} \\
    y_{\text{hinge}} &= 2.93 \text{ m} & \theta_z &= 94^\circ & \alpha_{\text{max}} &= 91.2^\circ
\end{align*}
\]

5.2 Mapping

This section presents the static maps of the simulation and real-world environments obtained as a result of the environment mapping algorithm described in Chapter 4.5.

The robot is teleoperated in the selected environment. The robot collects data from its laser scanner and odometry, and creates a map of the environment. After the mapping procedure, manual post-processing of the obtained map is carried out. During post-processing, the doors and the outliers are removed from the map, because the static map should include only static objects. The final maps of the simulation environment and real-world environment are shown in Figures 5.2 and 5.3 respectively. As mentioned in Chapter 4.5, the static map is an occupancy grid. The white cells of the map correspond to free or unoccupied space, the black cells correspond to the occupied space, and the gray cells correspond to the unexplored space. The size of a cell is 5 cm by 5 cm.

A map is defined in a map coordinate frame. The map coordinate frames are added to the figures showing the static map of the simulation and real-world
Validation environment. In order to also illustrate the door location in the environments, the hinge coordinate frames are also added.

![Figure 5.2](image1.png)

Figure 5.2: The static map obtained with mapping algorithm in the simulation environment with added map and hinge coordinate frame.

![Figure 5.3](image2.png)

Figure 5.3: The static map obtained with the mapping algorithm in the real-world environment with added map and hinge coordinate frame.

5.3 Navigation

This section presents results from testing the implemented autonomous navigation described in Chapter 4.4. The robot is given a task to move from its current
location to a new goal location in the environment.

Additional obstacles are added in both, simulation and real-world environment, in order to make the autonomous navigation task more difficult. On the beginning of the task, the robot has no information about the presence of obstacles. It first computes the path based only on the static map of the environment. While the robot is driving around the environment, obstacles are detected by the laser scanner. The robot includes the information about the obstacle in the \textit{global\_costmap} and \textit{local\_costmap} and recomputes the path.

Snapshots of visualization tool \textit{RViz} \cite{52} are made while the experiment is carried out in both environments. \textit{RViz} is a ROS tool for 3D visualization of sensor data and display of robot’s state information. The snapshots show how the planned path of the robot changes during the task execution due to detection of new obstacles in the environment. The 2D laser readings are presented with yellow color. The red arrow marks the desired goal pose of the robot. The green curve illustrates the global path computed by the \textit{global\_planner}. The \textit{global\_costmap} is also shown in the snapshots. The color-code of \textit{global\_costmap}’s cells is the following. The cyan cells present the forbidden zone, where the robot’s center point is not allowed to enter. The forbidden zone inflates around the walls and detected obstacles with a radius equal to the robot’s outer-shell radius. The forbidden area is surrounded with red and purple cells which indicate different cost value in the \textit{global\_costmap}. The cost exponentially decreases towards the grey cells which are associated with zero cost in the \textit{global\_costmap}.

During the experiment, the actual location of the robot is tracked. The actual path is later presented in separate figures for both environments. In both figures, the red line displays the actual path robot made. The blue circle marks the starting location of the robot and the green circle marks the goal location. The black parts correspond to walls and obstacles.
Simulation

Six obstacles were placed in the simulation environment. The obstacles have cylindrical shape with radius of 0.3 m and height of 1.0 m. The experiment started with robot being in the lower-left corner of the left room facing the wall. A goal pose is given in the upper-right corner of the right room. Figure 5.4 shows the previously mentioned snapshots at several time instances. At the beginning (Figure 5.4a), the robot is not aware of the obstacles and computes the global path which goes diagonally through the two rooms directly to the goal. As the robot moves around the environment and observes new obstacles (Figures 5.4b, 5.4c and 5.4d), it also recomputes the global path to avoid the obstacles. The robot successfully completes the experiment in 23 s. The actual path of the robot and the position of the obstacles is shown in Figure 5.5.

Figure 5.4: Calculated global path at different time instances in the simulation environment. Adaptation of the global path is performed when new obstacles are detected with laser scanner.
5.3 Navigation

Figure 5.5: The actual path the robot followed while executing the navigation task in the simulation environment.

Real-world

Four squared boxes were placed in the real-world environment. The length of the side of the box is 42 cm. The experiment starts with the robot being positioned in the hallway. The task is to autonomously navigate to the given goal pose inside the room. Figure 5.6 shows the previously mentioned snapshots at several time instances. At the beginning (Figure 5.6a), the robot is not aware of the obstacles and computes the global path which is almost a straight line directly to the goal. As the robot moves around the environment and observes new obstacles (Figures 5.6b, 5.6c and 5.6d), it also recomputes the global path to avoid the obstacles. The robot successfully completes the experiment in 21 sec. The actual path of the robot and the position of the obstacles is shown in Figure 5.7.
Validation

(a) Captured at $t = 0\,\text{s}$
(b) Captured at $t = 5\,\text{s}$
(c) Captured at $t = 10\,\text{s}$
(d) Captured at $t = 15\,\text{s}$

Figure 5.6: Calculated global path at different time instances in the real-world environment. Adaptation of the global path is performed when new obstacles are detected with laser scanner.

Figure 5.7: The actual path the robot followed while executing the navigation task in the real-world environment.
5.4 Door State Estimation

This section evaluates the developed door state estimation algorithm. Five experiments were carried out both in simulation and real-world environment.

The first experiment evaluates how the robot’s relative position to the door impacts the output of the algorithm. In this experiment, the robot is placed to five different predefined positions while the door angle $\alpha$ is kept constant. The second experiment examines the opposite scenario. The robot is placed in front of the door and it does not change its position while the door angle $\alpha$ is changed to several predefined angles. The impact of five different door angles $\alpha$ on the algorithm’s output is evaluated. In the third experiment, neither the robot position nor the door angle $\alpha$ change, but an obstacle is placed in the door area. The output of the algorithm is evaluated for four different types of obstacles. The fourth experiment evaluates the dynamic performance of the door state estimation algorithm. The robot is again placed in front of the door and its position does not change. The door angle $\alpha$ changes continuously with a constant rate of $18^\circ$/s. The fifth experiment measures the computational time of the algorithm when the robot is placed in front of the door. The sixth experiment evaluates the impact of the number of laser beams crossing the door area on the computation time of the door state estimation algorithm.

The developed algorithm is initialized with three parameters: the size of the tolerance band $\alpha_{tol}$, the size of the bins $\alpha_{step}$ and the IIR filter constant $K_{fil}$. This parameters have the following values for all experiments:

\[
\begin{align*}
\alpha_{tol} &= 10^\circ \\
\alpha_{step} &= 2^\circ \\
K_{fil} &= 0.30
\end{align*}
\]

The resolution of the algorithm is $\alpha_{step} = 2^\circ$ meaning that the estimation error below $2^\circ$ is considered as an accurate result.
5.4.1 First Experiment: Influence of the robot position

The first experiment evaluates the impact of the robot’s relative position to the door on the output of the door state estimation algorithm. The robot is placed on five predefined positions, shown in Figure 5.8, while the door angle $\alpha$ is kept constant at 30°. The experiment was carried out in both simulation and real-world environment.

The position A is directly in front of the door. In position B, the robot is positioned slightly to the left of the door, and in position C, the robot is placed slightly to the right of the door. Position D is further away from the door and position E is on the other side of the door. The exact coordinates are displayed in Figure 5.8.

The position A is directly in front of the door. In position B, the robot is positioned slightly to the left of the door, and in position C, the robot is placed slightly to the right of the door. Position D is further away from the door and position E is on the other side of the door. The exact coordinates are displayed in Figure 5.8.

Simulation

This section presents the results from the first experiment for the door state estimation algorithm in the simulation environment. Figure 5.9 shows all five scenarios tested in this experiment. On the left side, the robot positions with respect to the door are shown while on the right side the corresponding outputs of the door state estimation algorithm are presented.
5.4 Door State Estimation

(a) Robot in position A, directly in front of the door.

(b) Robot is positioned slightly to the left with respect to the door.

(c) Robot in position C, slightly to the right with respect to the door.

(d) Robot in position D, far away from the door.

(e) Robot in position E, on the other side of the door.

Figure 5.9: The output of the door state estimation algorithm for five predefined locations of the robot in the simulation environment.
Figure 5.9a shows the result when robot is positioned directly in front of the door (position A). The robot’s laser scanner has a good view of the door area. The estimated door angle $\alpha_{est}$ is $29^\circ$ and confidence level of estimation $c_{est}$ is 0.92.

Figure 5.9b shows the result when robot is positioned slightly to the left of the door (position B). In this layout, the door is almost parallel to the emitted laser beams. As a result, a lot less laser hits occur in the door area compared to the previous layout, where robot was directly in front of the door. Although the robot’s laser scanner does not have a good view of the door area, the results are similar. The estimated door angle $\alpha_{est}$ is $29^\circ$ and confidence level of estimation $c_{est}$ is 0.80.

Figure 5.9c shows the result when robot is positioned slightly to the right of the door (position C). The laser scanner does not have a clear view of the door area in this position. A part of the door is occluded by the wall on the right side of the door. Because of the occlusion, the level of confidence is lower. The estimated door angle $\alpha_{est}$ is $29^\circ$ and confidence level of estimation $c_{est}$ is 0.53. Although the confidence level is lower, the error of estimation does not increase.

Figure 5.9d shows the result when robot is positioned further away from the door (position D). The laser scanner still has a clear view of the door area and the result is comparable to the robot being directly in front of the door. The estimated door angle $\alpha_{est}$ is again $29^\circ$ and confidence level of estimation $c_{est}$ is 0.87.

Figure 5.9e shows the result when robot is positioned on the other side of the door (position E). The robot’s laser scanner has again a good view of the door area and outputs a similar result as in position A. The estimated door angle $\alpha_{est}$ is $31^\circ$ and confidence level of estimation $c_{est}$ is 0.95.

Real-world

This section presents the results from the first experiment for door state estimation algorithm in the real-world environment. Figure 5.10 shows all five scenarios
tested in this experiment. On the left side, the robot positions with respect to
the door are shown while on the right side the corresponding outputs of the door
state estimation algorithm are presented.

Figure 5.10a shows the result when robot is positioned directly in front of the
doors (position A). The robot’s laser scanner has a good view of the door area.
The estimated door angle $\alpha_{est}$ is $29^\circ$ and confidence level of estimation $c_{est}$ is 0.94.

Figure 5.10b shows the result when robot is positioned slightly to the left of
the door. In this layout, only nine laser beams reflect from the door, which is
significantly less than in the previous layout, when robot is directly in front of the
doors. The cluster of laser hits corresponding to the door is rather short. Some
laser beams do not reflect from the door because of the steep angle of incidence.
The estimated door angle $\alpha_{est}$ is $25^\circ$ and confidence level of estimation $c_{est}$ is 0.59.

Figure 5.10c shows the result when robot is positioned slightly to the right of
the door (position C). The laser scanner does not have a clear view of the door
area in this position. A part of the door is occluded by the wall on the right
side of the door. Because of the occlusion, the level of confidence is lower. The
estimated door angle $\alpha_{est}$ is $25^\circ$ and confidence level of estimation $c_{est}$ is 0.52.

Figure 5.10d shows the result when robot is positioned further away from the
doors (position D). The laser scanner still has clear view of the door area and
the result is comparable to the robot being directly in front of the door. The
estimated door angle $\alpha_{est}$ is $27^\circ$ and confidence level of estimation $c_{est}$ is 0.78.

Figure 5.10e shows the result when robot is positioned on the other side of the
doors (position E). The robot’s laser scanner has again a good view of the door
area and outputs similar result as in position A. The estimated door angle $\alpha_{est}$ is
$29^\circ$ and confidence level of estimation $c_{est}$ is 0.88.
(a) Robot in position A, directly in front of the door.

(b) Robot in position B, slightly to the left with respect to the door.

(c) Robot in position C, slightly to the right with respect to the door.

(d) Robot in position D, far away from the door.

(e) Robot in position E, on the other side of the door.

Figure 5.10: The output of the door state estimation algorithm for five predefined locations of the robot in the real-world environment.
Conclusion

The results from this experiment for simulation and real-world environment are gathered in Table 5.1 and Table 5.2 respectively. The first experiment shows that the robot position does not directly impact the output of the door state estimation algorithm, but it can affect the quality of the laser scan data, which then directly impacts the output of the door state estimation algorithm. It is expected that the door state estimation algorithm cannot provide a good estimate of the door angle $\alpha$ when the door is occluded or barely observed in the laser scan readings. The output with low level of confidence in estimation $c_{est}$ is a signal for the robot to change its position in order to have a better view of the door area.

When the robot has a good view of the door area, the associated result is adequate.

Table 5.1: The results of the door state estimation algorithm for five predefined robot positions in the simulation environment.

<table>
<thead>
<tr>
<th>robot position</th>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>30°</td>
<td>29°</td>
<td>$-1^\circ$</td>
<td>0.92</td>
</tr>
<tr>
<td>B</td>
<td>30°</td>
<td>29°</td>
<td>$-1^\circ$</td>
<td>0.80</td>
</tr>
<tr>
<td>C</td>
<td>30°</td>
<td>29°</td>
<td>$-1^\circ$</td>
<td>0.53</td>
</tr>
<tr>
<td>D</td>
<td>30°</td>
<td>29°</td>
<td>$-1^\circ$</td>
<td>0.87</td>
</tr>
<tr>
<td>E</td>
<td>30°</td>
<td>31°</td>
<td>$1^\circ$</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 5.2: The results of the door state estimation algorithm for five predefined robot positions in the real-world environment.

<table>
<thead>
<tr>
<th>robot position</th>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>30°</td>
<td>29°</td>
<td>$-1^\circ$</td>
<td>0.94</td>
</tr>
<tr>
<td>B</td>
<td>30°</td>
<td>25°</td>
<td>$-5^\circ$</td>
<td>0.59</td>
</tr>
<tr>
<td>C</td>
<td>30°</td>
<td>25°</td>
<td>$-5^\circ$</td>
<td>0.52</td>
</tr>
<tr>
<td>D</td>
<td>30°</td>
<td>27°</td>
<td>$-3^\circ$</td>
<td>0.78</td>
</tr>
<tr>
<td>E</td>
<td>30°</td>
<td>29°</td>
<td>$1^\circ$</td>
<td>0.88</td>
</tr>
</tbody>
</table>
5.4.2 Second Experiment: Influence of the door angle $\alpha$

The second experiment evaluates the impact of the door angle $\alpha$ on the output of the door state estimation algorithm. The robot is placed in front of the door (position A in Figure 5.8) and its position is unchanged during the experiment, while the door angle $\alpha$ is changed to five predefined angles. The experiment was carried out both in simulation and real-world environment.

Simulation

This section presents the results from the second experiment for door state estimation algorithm in the simulation environment. As mentioned, the door angle $\alpha$ is changed to five predefined positions. This positions are: $0^\circ$, $30^\circ$, $60^\circ$, $90^\circ$ and $120^\circ$. Figure 5.11 shows all five layouts tested in this experiment. On the left side, the layout of the robot with respect to the door is shown, while on the right side, the corresponding output of the door state estimation algorithm is presented.

The algorithm performs well and with comparable accuracy and confidence for the first four door angle setups ($0^\circ$, $30^\circ$, $60^\circ$ and $90^\circ$). Only the result from the last setup, when door angle was set to $120^\circ$, seems to be an outlier. The door angle error is higher compared to previous results, but also the confidence level is lower. This is caused by a very inconvenient setup of the robot and door. The door leaf is almost parallel to the laser beams. The door is barely observed in the laser scan readings, which means only a few laser beams carry information about the actual location of the door.
Figure 5.11: The output of the door state estimation algorithm for five different door angles $\alpha$ in the simulation environment.
Real-world

This section presents the results from the second experiment for door state estimation algorithm in the real-world environment. Compared to the same experiment in simulation environment, different predefined door angles $\alpha$ are chosen, because the door in the real-world environment does not open more than $91.2^\circ$. The predefined door angles $\alpha$ are: $0^\circ$, $20^\circ$, $40^\circ$, $60^\circ$ and $80^\circ$. The door was positioned manually by a human. A few degrees of error can be expected from the manually positioning of the door.

Figure 5.12 shows all five layouts tested in this experiment. On the left side, the layout of the robot with respect to the door is shown, while on the right side, the corresponding output of the door state estimation algorithm is presented.

The algorithm performs well and with comparable accuracy and confidence in all five cases. The conclusion of this experiment is similar to conclusion of the first experiment. As long as door is fully observed in the laser scan readings, the door state estimation algorithm will have an accurate measurement with a high confidence.
5.4 Door State Estimation

Figure 5.12: The output of the door state estimation algorithm for five different door angles $\alpha$ in the real-world environment.
Conclusion

The results from second experiment in the simulation and real-world environment are gathered in Table 5.3 and Table 5.4 respectively. The second experiment shows that the door angle $\alpha$ does not influence directly the performance of the door state estimation algorithm. The algorithm performs well for all door angles $\alpha$ as long as the door is well observed in the laser scan readings. The problematic door angles $\alpha$ for the tested robot position (position A) are angles around $120^\circ$.

The purpose of the door state estimation algorithm is to identify if the door area is blocked by half-open doors or some other object, the fully open door does not block the door area. The lower confidence level $c_{est}$ in the estimated door angle $\alpha_{est}$ for the fully-open door is not a concern.

When the door is closed or half-way open, it is well present in the laser scan readings, and the output of the door state estimation algorithm is adequate. When the door is wide open such that it is nearly parallel to the emitted laser beams, it is not observed well in the laser scan readings, and the door state estimation algorithm is expected to have higher estimation error and lower confidence level.

Table 5.3: The results of the door state estimation algorithm for five different door angles $\alpha$ in the simulation environment.

<table>
<thead>
<tr>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^\circ$</td>
<td>$1^\circ$</td>
<td>$1^\circ$</td>
<td>0.95</td>
</tr>
<tr>
<td>$30^\circ$</td>
<td>$29^\circ$</td>
<td>$-1^\circ$</td>
<td>0.91</td>
</tr>
<tr>
<td>$60^\circ$</td>
<td>$59^\circ$</td>
<td>$-1^\circ$</td>
<td>0.88</td>
</tr>
<tr>
<td>$90^\circ$</td>
<td>$89^\circ$</td>
<td>$-1^\circ$</td>
<td>0.84</td>
</tr>
<tr>
<td>$120^\circ$</td>
<td>$115^\circ$</td>
<td>$-5^\circ$</td>
<td>0.59</td>
</tr>
</tbody>
</table>
Table 5.4: The results of the door state estimation algorithm for five different door angles $\alpha$ in the real-world environment.

<table>
<thead>
<tr>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>$-3^\circ$</td>
<td>$-3^\circ$</td>
<td>0.75</td>
</tr>
<tr>
<td>20°</td>
<td>19°</td>
<td>$-1^\circ$</td>
<td>0.92</td>
</tr>
<tr>
<td>40°</td>
<td>41°</td>
<td>1°</td>
<td>0.92</td>
</tr>
<tr>
<td>60°</td>
<td>63°</td>
<td>3°</td>
<td>0.91</td>
</tr>
<tr>
<td>80°</td>
<td>83°</td>
<td>3°</td>
<td>0.86</td>
</tr>
</tbody>
</table>
5.4.3 Third Experiment: Influence of obstacles in the door area

The third experiment evaluates the output of the door state estimation algorithm in presence of obstacle in the door area. The robot is placed directly in front of the door (position A in Figure 5.8). The door angle $\alpha$ is set to 60°. Both, the robot and the door are not moved during this experiment. The door state estimation algorithm was tested for four different obstacles. The experiment was carried out in simulation and real-world environment.

**Simulation**

This section presents the results from the third experiment for door state estimation algorithm in the simulation environment. The algorithm is tested for the following for obstacles. The first obstacle (Figure 5.13a) is a cylinder with radius 25 cm and height 1.0 m. The second obstacle (Figure 5.13b) are two cylinders with radius 7.5 cm and height 1.0 m. The third obstacle is a rectangular box (Figure 5.13c) with dimension: length 30 cm, width 60 cm and height 50 cm. The orientation of the box is selected arbitrarily. The forth obstacle is again the same rectangular box, but its orientation is carefully selected. The front surface of the box is aligned with the z-axis of the hinge coordinate frame, just like the door at that position.

Figure 5.13 shows all four scenarios tested in this experiment. On the left side, the scenario with the robot, the door and the obstacle is shown, while on the right side, the corresponding output of the door state estimation algorithm is presented. The results from this experiment are gathered in Table 5.5.
5.4 Door State Estimation

(a) Cylindrical obstacle with diameter $d = 50$ cm.

(b) Two cylindrical obstacles with diameter $d = 15$ cm.

(c) Rectangular box ($30$ cm $\times 60$ cm $\times 50$ cm) with arbitrarily selected orientation.

(d) Rectangular box ($30$ cm $\times 60$ cm $\times 50$ cm) with its side aligned with the hinge.

Figure 5.13: The output of the door state estimation algorithm for four different types of obstacles in the simulation environment.
Figure 5.13a shows the door state estimation output when the first obstacle is placed in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.68. The lower confidence level $c_{est}$ is caused by the partial occlusion of the door by the obstacle. But a more important observation is that the algorithm does not confuse the obstacle for a door. The score of the potential door angle candidates $\alpha_{plac}$ that correspond to laser hits caused by the obstacle are equal to zero.

Figure 5.13b shows the door state estimation output when the second obstacle was placed in the door area. The estimated door angle $\alpha_{est}$ is 57° and confidence level of estimation $c_{est}$ is 0.80. The slightly lower confidence level $c_{est}$ is again caused by the partial occlusion of the door. The score of the potential door angle candidates $\alpha_{plac}$ that correspond to laser hits caused by the obstacle are again equal to zero.

Figure 5.13c shows the door state estimation output when the third obstacle is placed in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.65. Similar as in previous two tests, the lower confidence level $c_{est}$ is caused by the partial occlusion and the score of the potential door angle candidates $\alpha_{plac}$ that correspond to laser hits caused by the obstacle are equal to zero.

Figure 5.13d shows the door state estimation output when the fourth obstacle is placed in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.57. The lower confidence level $c_{est}$ is again caused by the partial occlusion of the door. But this time, the score of some potential door angle candidates $\alpha_{plac}$ that correspond to laser hits caused by the obstacle are not equal to zero.
Real-world

This section presents the results from the third experiment for door state estimation algorithm in the real-world environment. The algorithm is tested for the following obstacles which are similar to the tested obstacles in the simulation environment. The first obstacle (Figure 5.14a) is cylindrical dustbin with radius 15 cm and height 45 cm. The second obstacle (Figure 5.14b) is human. The laser scan readings are similar as the second obstacle in the simulation. The third obstacle is a rectangular box (Figure 5.14c) with dimension: length 15 cm, width 60 cm and height 50 cm. The orientation of the box is selected arbitrarily. The forth obstacle is again the same rectangular box, but its orientation is carefully selected. The front surface of the box is aligned with the z-axis of the hinge coordinate frame, just like the door at that position.

Figure 5.14 shows all four scenarios tested in this experiment. On the left side, the scenario with the robot, the door and the obstacle is shown, while on the right side, the corresponding output of the door state estimation algorithm is presented. The results from this experiment are gathered in Table 5.6.
Figure 5.14: The output of the door state estimation algorithm for four different types of obstacles in the real-world environment.

(a) Dustbin with cylindrical shape (radius = 15 m).

(b) A human standing in the door area.

(c) Rectangular box (15 cm × 60 cm × 50 cm) with arbitrarily selected orientation.

(d) Rectangular box (15 cm × 60 cm × 50 cm) with its side aligned with the hinge.
Figure 5.14a shows the door state estimation output when the dustbin is placed in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.76. The slightly lower confidence level $c_{est}$ is caused by the partial occlusion of the door by the dustbin. But more importantly, the algorithm does not confuse the obstacle for a door. The score of the potential door angle candidates $\alpha_{pda_c}$ that correspond to laser hits caused by the obstacle are equal to zero.

Figure 5.14b shows the door state estimation output when a human was standing in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.77. The slightly lower confidence level $c_{est}$ is again caused by the partial occlusion of the door. The score of the potential door angle candidates $\alpha_{pda_c}$ that correspond to laser hits caused by the obstacle are again equal to zero.

Figure 5.14c shows the door state estimation output when the unaligned box is placed in the door area. The estimated door angle $\alpha_{est}$ is 59° and confidence level of estimation $c_{est}$ is 0.65. Similar as in previous two tests, the lower confidence level $c_{est}$ is caused by the partial occlusion and the score of the potential door angle candidates $\alpha_{pda_c}$ that correspond to laser hits caused by the obstacle are equal to zero.

Figure 5.14d shows the door state estimation output when the box was aligned with the door hinge. The estimated door angle $\alpha_{est}$ is 57° and confidence level of estimation $c_{est}$ is 0.48. The lower confidence level $c_{est}$ is again caused by the partial occlusion of the door. But this time, the score of some potential door angle candidates $\alpha_{pda_c}$ that correspond to laser hits caused by the obstacle are not equal to zero.
Conclusion

The results from third experiment in the simulation and real-world environment are gathered in Table 5.5 and Table 5.6 respectively. The door state estimation algorithm performed well in the third experiment. The algorithm processed the first three obstacles both in simulation and real-world environment and assigned them zero score. With other words, obstacles had no chance to be identified as a door. The forth obstacle was tricky. The door is expected to be a vertical plane aligned with z-axis of the hinge coordinate frame. The laser scan readings from the aligned box highly resemble the laser scan readings from the door at that position. But since the box’s width is shorter then the door’s width, the algorithm is able to correctly estimates the real door angle.

Table 5.5: The results of the door state estimation algorithm for four different types of obstacles in the simulation environment.

<table>
<thead>
<tr>
<th>obstacle type</th>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinder $(r = 25$ cm)</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.68</td>
</tr>
<tr>
<td>two cylinders $(r = 7.5$ cm)</td>
<td>60°</td>
<td>57°</td>
<td>$-3°$</td>
<td>0.80</td>
</tr>
<tr>
<td>unaligned box</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.65</td>
</tr>
<tr>
<td>aligned box</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Table 5.6: The results of the door state estimation algorithm for four different types of obstacles in the real-world environment.

<table>
<thead>
<tr>
<th>obstacle type</th>
<th>real door angle $\alpha$</th>
<th>estimated door angle $\alpha_{est}$</th>
<th>estimation error $\alpha_{err}$</th>
<th>confidence level $c_{est}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dustbin</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.76</td>
</tr>
<tr>
<td>human</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.77</td>
</tr>
<tr>
<td>unaligned box</td>
<td>60°</td>
<td>59°</td>
<td>$-1°$</td>
<td>0.65</td>
</tr>
<tr>
<td>aligned box</td>
<td>60°</td>
<td>57°</td>
<td>$-3°$</td>
<td>0.46</td>
</tr>
</tbody>
</table>
5.4.4 Fourth Experiment: Tracking of the rotating door

The fourth experiment evaluates the output of the door state estimation algorithm when tracking the door moving with constant angular velocity of 18°/s. The robot is placed directly in front of the door (position A in Figure 5.8) and does not move. The door angle $\alpha$ changes continuously from 0° to 90° with a constant slope of 18°/s. The experiment was carried out only in the simulation environment, where the door angle $\alpha$ can be accurately controlled.

(a) Captured at $t = 5$ s  
(b) Captured at $t = 7.5$ s  
(c) Captured at $t = 10$ s

(d) Comparison between real door angle $\alpha$ and estimated door angle $\alpha_{est}$ with the confidence level $c_{est}$ in the background.

Figure 5.15: The output of the door state estimation algorithm when tracking the door moving with constant angular velocity of 18°/s in the simulation environment.
The results of the fourth experiment are presented in Figure 5.15. Figures 5.15a, 5.15b and 5.15c capture the robot and the door state at different moments in time. Figure 5.15d presents the output of door state estimation algorithm during the experiment. The real door angle $\alpha$ is highlighted with red color, while the estimated door angle $\alpha_{est}$ is highlighted with blue color. The confidence level of estimation $c_{est}$ is shown in the background with cyan color. As it can be observed, the estimated door angle $\alpha_{est}$ follows the real door angle $\alpha$ with negligible error. In the first five seconds of the experiment, the estimated door angle $\alpha_{est}$ was close to $0^\circ$. The door state estimation algorithm has a high confidence level around 0.90. When door starts to open, the output of door state estimation algorithm follows. During the door movement the confidence level drops. The average level of confidence during this time period is around 0.50. Once the door stops rotating the confidence level grows back to approximately 0.85.

Conclusion

This experiment confirms that door state estimation algorithm is capable of tracking the door that is being moved. The confidence level however drops and the robot is not as certain about the door position while the door is moving. This is caused by the filtering of individual confidence levels as described in fourth step of the door state estimation algorithm in Chapter 3.6. The filter is there to prevent high frequency noise from the laser scan measurements to affect the results, but unfortunately it also prevents tracking of dynamic door movement with high confidence. The performance of door state estimation algorithm for dynamic tracking can be significantly improved by adjusting the filter properties.

Fortunately, accurately tracking a door with angular velocity is not a concern for a household service robot. The application of door state estimation algorithm is to distinguish a door from normal objects with an intention to actively intervene in the environment by pushing a door. A moving door is very likely caused by a human, in which case a robot should not intervene.
5.4.5 Fifth Experiment: Measurement of the computation time

The fifth experiment evaluates the computational complexity of the door state estimation algorithm. The robot is placed directly in front of the door (position A in Figure 5.8) and it does not move. The door angle $\alpha$ does not affect the computation time and is moved randomly during this experiment. The computation time of the algorithm is measured for an interval of 10 s. The computation time is the time it takes for the on-board processor to execute all the five steps of the algorithm and return the result. This experiment was carried out only in real-world environment.

The door state estimation algorithm has only one real-time constraint. This constraint is the sampling frequency of the 2D laser scanner, which is 8.5 Hz. This means that the algorithm’s computation time must be lower than 117 ms.

![Figure 5.16: Computation time of the door state estimation algorithm, when the robot is positioned in front of the door (position A in Figure 5.8).](image)

The results of the fifth experiment are presented in Figure 5.16. The computation time of the algorithm, drawn with blue color, is around 75 ms during the whole experiment. The computation time never exceeds the computation limit of 117 ms, highlighted with red dashed line.
5.4.6 Sixth Experiment: Influence of laser beams that cross the door area on the computation time

The sixth experiment evaluates the impact of the number of laser beams that cross the door area on the computation time of the door state estimation algorithm. The computation time and the number of door-area-crossing laser beams is stored each iteration of the door state estimation algorithm. In order to obtain data with several different values for the number of door-area-crossing laser beams, the robot is driven around manually. The robot’s path is irrelevant for this experiment.

![Figure 5.17: Correlation between the number of laser beams that cross the door area and the computation time of the door state estimation algorithm.](image)

The results of the sixth experiment are presented in Figure 5.17. The mean and the standard deviation of computation time are shown for different number of door-area-crossing laser beams. The figure suggest a linear relation between the mean computation time and number of door-area-crossing laser beams. This relation can help to estimate the required computation time for the worst case scenario, if one can estimate the number of door-area-crossing laser beams in the worst case scenario.
6 Conclusion

6.1 Summary

This thesis concentrates on mobile household robots and their fundamental challenges with the autonomous navigation. The main focus is the development of an algorithm which is capable of estimating the state of the door. The purpose of this algorithm is to estimate the door angle and to distinguish between doors and obstacles in the environment based on 2D laser scan readings. This enables the robot to make contact with the doors in order to push them open and improve the local and global navigation in the household environment.

The door state estimation algorithm that is based on a voting scheme (presented in Chapter 3.6) is developed, implemented and tested. The algorithm is written as a ROS node using Python 2.7 and can be reused without changes on any robot running ROS and using 2D laser scanner. The performance of the algorithm satisfies the requirements set in Chapter 3.2 which can be summed up into accurate door angle estimation in real-time with or without the presence of obstacles.

We show that the developed door state estimation algorithm performs with accurate door angle estimation and high confidence for variety of robot positions and door angles. We also show that the algorithm handles situations with obstacles being present in the door area. Other objects than a door leaf are recognized as obstacles, meaning the robot should not attempt to push them in order to cross the door area.
We also show that the door state estimation algorithm accurately estimates the door angle of a moving door, but the confidence level is lower compared to static situation. Since the robot should not attempt to cross the door area when the door is moving, the accurate tracking of dynamic door angle with high confidence is not the priority of the algorithm. The algorithm also meets the real-time constraints imposed by sampling rate of the laser scanner as discussed in Chapter 3.2.

The results satisfy the requirements, but an important remark must be made. The algorithm relies on the static door parameters \((x_{hinge}, y_{hinge}, \theta_x, \theta_z, w \text{ and } \alpha_{max})\) and on the fact that the robot is localized in the environment. The static door parameters can be obtained with high accuracy, but this is typically not the case for robot localization. The door state estimation algorithm is guaranteed to work when the error of robot localization is smaller than the resolution of the static map.

Additionally to the door state estimation algorithm, the robot is equipped with software for environment mapping and autonomous navigation. Both of these components are part of software packages provided by the ROS framework. Our testing shows satisfactory results from both components.

### 6.2 Future Work

Although the developed door state estimation algorithm satisfies the requirements, there is still room for improvement. The algorithm can be made computationally more effective by optimizing the written code. A better computation time can be achieved by using another programming language. As mentioned before, the implementation is realized using Python 2.7, but the ROS framework also supports C++ programming language which is known to be more efficient.

Improvement can be done also for the tracking of the door angle for a moving door. We suggest modifying the score filtering. Instead of filtering the individual
levels, one could filter the scores by including the dynamic model of the hinge door. It is reasonable to assume that the door angle cannot change much between the two subsequent laser scan measurements.

With certain modifications, the approach can be extended to rotating door type. The approach can also be used to detect sliding doors by changing the kinematic model behind the algorithm.
Literature


