

Aalto University
School of Electrical Engineering
Master's Programme in Automation and Electrical Engineering

Niilo Metsänen

Assessment of local path planners in a indoor and outdoor robot

Master's Thesis
Espoo, 27. July 2020

Supervisors: Professor Ville Kyrki, Aalto University
Advisor: Tapio Taipalus D.Sc. (Tech.)

Aalto University
 School of Electrical Engineering
 Master's Programme in Automation and Electrical Engineer-
 ing

ABSTRACT OF
 MASTER'S THESIS

Author:	Niilo Metsänen	
Title:	Assessment of local path planners in a indoor and outdoor robot	
Date:	27. July 2020	Pages: 61
Major:	Control, Robotics, and Autonomous Systems	Code: ELEC3025
Supervisors:	Professor Ville Kyrki, Aalto University	
Advisor:	Tapio Taipalus D.Sc. (Tech.)	
<p>Modern mobile robotics are entering commercial use in a variety of non-controlled environments. One such robot is the Roboguide service and guide robot for the visually impaired. For the smooth operation of a service robot in the daily life of its users, it is imperative that the paths along which the robot travels are intuitive, comfortable, and above all, safe.</p> <p>The goal of this thesis is to assess the viability of the Elastic Band, Timed Elastic Band and Dynamic Window Approach path-planners in a dynamic environment. This is accomplished through testing various scenarios typical in dynamic environments, including outright collisions and near-miss scenarios. The testing is done on a simulated platform.</p> <p>In addition to assessing the current viability of the path-planners in question, this thesis also aims to identify challenges and problems caused by the dynamic nature of the environment. The results suggest the Timed Elastic Band is the superior path-planner. Dynamic obstacles create problems for all the tested path-planners, and a future approach to cost-efficient dynamic prediction is suggested.</p> <p>The tests within this thesis are implemented using Robotic Operating System(ROS) and the robot simulation environment Gazebo. Implementations are based on real products and software modules.</p>		
Keywords:	Path-planning, mobile robotics, autonomous robot, laser scanning, robot navigation, dynamic environment, costmap, asymmetric Gaussian cost function, simulation	
Language:	English	

Tekijä:	Niilo Metsänen		
Työn nimi:	Assessment of local path planners in a indoor and outdoor robot		
Päiväys:	27. Heinäkuuta 2020	Sivumäärä:	61
Pääaine:	Control, Robotics, and Autono- mous Systems	Koodi:	ELEC3025
Valvojat:	Professori Ville Kyrki, Aalto University		
Ohjaaja:	Tapio Taipalus D.Sc. (Tech.)		
<p>Nyky aikaista autonomista robotiikkaa on alettu soveltaa kaupallisessa käytössä erilaisissa kontrolloimattomissa toimintaympäristöissä. Yksi näistä on sovelluskohteista on Roboguide, näkövammaisille suunnattu opasrobotti. Jotta robottia olisi intuitiivista ja turvallista käyttää, on oleellista että robotti pystyy toimimaan arkipäivän eri tilanteissa helppokäyttöisesti, mukavasti ja ennen kaikkea turvallisesti.</p> <p>Tämän diplomityön tavoite on arvioida Elastic Band, Timed Elastic Band ja Dynamic Window Approach reittisuunnittelualgoritmien soveltuvuutta dynaamisessa ympäristössä. Tätä varten on toteutettu testisarja, jossa simuloidaan tyypillisiä dynaamisen ympäristön haasteita, kuten törmäys- ja läheltä-ohitustilanteita. Testaus toteutettiin simuloitulla alustalla.</p> <p>Eri reittisuunnittelualgoritmien soveltuvuuden arvioinnin lisäksi diplomityö pyrkii tunnistamaan dynaamisessa ympäristössä liikkumiseen liittyviä haasteita ja uhkakuvia. Testatuista algoritmeista Timed Elastic Band soveltuu selvästi parhaiten dynaamiseen ympäristöön. Lisäksi työssä ehdotetaan lähestymistapaa dynaamisten esteiden sijainnin ennustamiseen laskennallisesti tehokkaasti.</p> <p>Testaus on toteutettu ROS-pohjaisella robotilla ja testit on suoritettu Gazebo-simulointiympäristössä. Testaus ja simuloitu robotti perustuu aitoon tuotteeseen ja sen komponentteihin.</p>			
Asiasanat:	Reittisuunnittelu, mobiili robotiikka, autonominen robotti, laserkeilaus, robottinen navigointi, dynaaminen ympäristö, kustannuskartta, asymmetrinen normaalijakautunut kustannusfunktio, simulaatio		
Kieli:	Englanti		

Preface

I would like to thank my employer, GIM Robotics, for the opportunity to learn from the country's most talented robotic engineers and for the means to complete my thesis. I would especially like to thank Tapio for both his invaluable insight and the motivation to complete this work. Thank you Ville for your help as supervisor and your work as well as the great work you do teaching.

Additionally I extend my thanks to my guild AS, a source of endless support and help in my studies. Thank you to ASH14 and ASH15, with whom I shared a lot of amazing experiences with. Thanks to FTMK15 as well, for all harebrained fun we got up to. Thank you Hankkijat, thank you Luola, and thank you to all the close friends I've met. Without you I'd have graduated years ago, and be so many unforgettable memories poorer for it.

Finally I'd like to thank my family, for the motivation to move forward. Thank you Noora, as well, for putting up with me when I wasn't the most fun to be around.

Helsinki, 27. July 2020

Niilo Metsänen

Contents

1	Introduction	7
1.1	Thesis case	8
1.2	Problem statement	10
1.3	Structure of the Thesis	12
2	Robot Navigaton	13
2.1	Simultaneous Localization and Mapping	13
2.2	Costmap generation	15
2.3	Path planning	17
2.3.1	Reasoning	18
2.3.2	Dynamic Window Approach	18
2.3.3	Elastic Band	19
2.3.4	Timed Elastic Band	20
2.4	Evaluation criteria	21
3	Implemented methods	24
3.1	Implemented teach-and-repeat method	24
3.2	Modelling the cost surrounding an obstacle with Gaussians	25
4	Experiments	28
4.1	Test environment	28
4.2	Test setup	30
4.2.1	Robotic Operating System	31
4.2.2	Simulation platform	33
4.3	Experiments	34
4.3.1	Baseline	34
4.3.2	Tests	36
4.4	Results	44
4.4.1	Divergent results	44
4.5	Path-planner performance	45
4.5.1	Elastic Band	45

4.5.2	Timed Elastic Band	46
4.5.3	Dynamic Window Approach	47
5	Discussion	49
5.0.1	Evaluation criteria	49
5.0.1.1	Success Ratio	49
5.0.1.2	Fluctuation Ratio	50
5.0.1.3	Time and Cumulative Distance	50
5.1	Comparison between path-planners	50
5.2	Future implementation	52
6	Conclusions and Future Work	54

Chapter 1

Introduction

The rapidly developing field of mobile robotics has steadily expanded to encompass a wide array of different industries. Some of the challenges are universal, but many, if not most of the challenges in autonomous mobile robotics are partially or completely specific to the particular use-case in question. While many generic solutions have been proposed and accepted, the ease of implementing them depends on the available sensors, their configuration, intended environment, etc.

Robotic navigation comprises of localization of the robot within a known coordinate frame, the planning of a path [1] to a specified goal, and the control of the robot to achieve a trajectory that conforms to the plan. Path-planning can be further divided into global path-planning, with a focus on devising a path through what is typically a known or partially known environments, and local path-planning, which will create an adjusted locally applicable path to avoid unexpected obstacles[2].

In the case of this thesis, the global path-plan is generated manually, by recording a remote-controlled path as a series of poses. The local path-planner's goal is to avoid any new obstacles on or near the path.

The ROS framework is a useful environment to develop robotic systems with multiple modules running independently. It is widely used in robotics development, and various plug-and-play implementations already exist for it.

When approaching a mobile robotics problem, it is important to understand the environment as well. The perceptual, computational and safety requirements may vary wildly between two scenarios, even if the sensor setup is nearly identical. It is vital to understand the tools at your disposal and their limitations within the environment to design a functional autonomous robot.

As such, it is currently nearly impossible to implement a generic robotic approach that fits all possible applications that utilize the same core system.

For example, an offroad vehicle tasked with hauling lumber and a truck travelling between cities along a highway may use similar base vehicles, adhering to the same dynamic and control constraints, but the challenges they face and the rules they must abide differ meaningfully. The offroad robotic system does not need to worry about other vehicles on the road or traffic laws; instead it must be able to detect difficult or impassable terrain. The intercity truck can often assume good road conditions, but must be able to navigate with other, independently travelling vehicles. One focuses on analyzing the environment to find paths that are possible to drive on, the other knows the terrain and must instead analyze the context to find socially and legally acceptable paths to navigate.

1.1 Thesis case

Robotic implementations have become widespread in various fields. Typical implementations that exist commercially are centered around largely human-free environments, ensuring that even in the case of catastrophic failures, the costs are limited to material damages and not human wellbeing. A typical example would be a fully automated warehouse[3], in which the environment can be set up specifically for ease of navigation for robots, as well as the possibility for communication and cooperation between different robotic agents within the environment. Recently, however, initial commercial forays into autonomous navigation in public and dynamic environments have commenced[4]. Navigating a public road or other space means the robot can no longer rely on strictly defined paths or communication between parties, instead the robot must be able to predict and adapt to new dynamic obstacles as they move about the environment in which the robot is operating.

The system in question for this thesis is the Roboguide robot, illustrated in 1.1, a guide robot for the visually impaired. The design of the robot is to aid a visually impaired user navigate a predefined path between preset start and goal points. The path to be followed is not generated by the robot itself but is instead recorded. The environment the Roboguide is used in is a pedestrian passageway, such as a sidewalk, and is, for the purpose of this thesis, assumed to be populated by pedestrians in addition to typical static obstacles one might find near or on a sidewalk.

The general design of the robot is a two-wheeled non-balancing robot. Instead of balancing itself, a handle rod is attached to the base of the robot. This handle rod contains a handle and controls for the user interface, as well as the majority of the sensors. Along the rod there is a spot for the LiDAR,

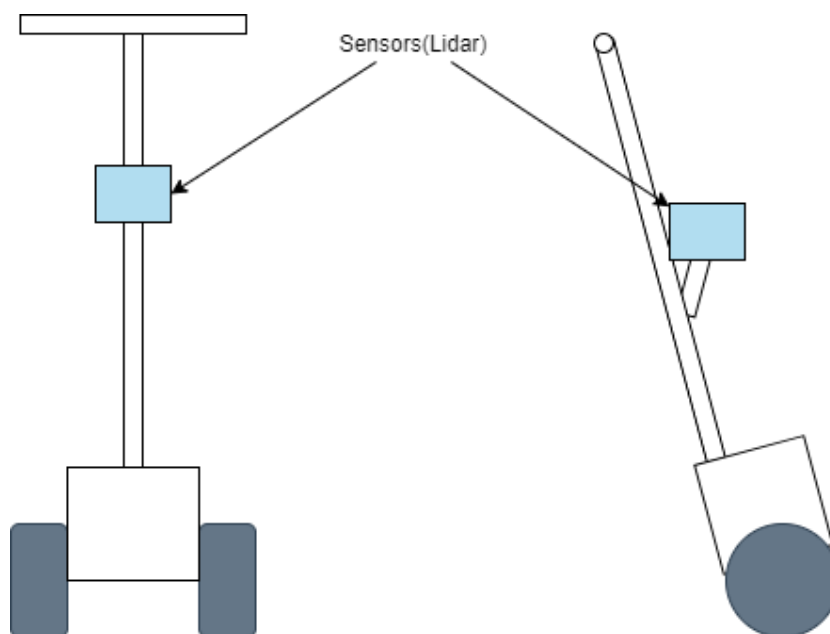


Figure 1.1: An illustration of the Roboguide robot. The Lidar is situated on the handle. The base contains the processor and wheel motors.

IMU and, in the future, a camera. The rod is connected to the base by a ball joint, giving the rod and by extension the sensors a degree of freedom in movement in relation to the base. The base contains odometry sensors, the computer unit as well as a sensor for the state of the ball joint.

As a service robot meant to guide a walking person from point to point, we will impose certain requirements for the robot, particularly its path-planner. As the user is visually impaired, potentially completely blind, the comfort and intuitiveness of operation is particularly important. While a logistics robot may take roundabout or looping routes to reach its objective, it is unreasonable to expect the user of a guide robot to follow a robot on winding trail when a much smoother one is available. Likewise, intuitively understandable and thus comfortable to follow routes can be assumed to be the straightest viable options. In addition, it can be assumed that constant minor corrections to the heading, a so-called "wobble", is undesirable. While not necessarily impacting the other performance metrics of the path-planning algorithm, following a wobbling path feels unintuitive and will make predicting when the robot is actually making a turn difficult.

Safety is of paramount importance when designing a human-operated robot, especially so when the robot is making autonomous decisions. In

this case however it is possible to make some assumptions regarding collision avoidance. As a guide tool for the visually impaired, it can generally be taken for granted that other pedestrians will both notice and know to avoid collisions with the robot and user. It is also possible that other pedestrians may approach or wish to stop the user for a variety of reasons; it would be impractical and awkward for the robot to lead the user on a chase in an attempt to avoid a perceived collision. Nonetheless, the robot should react to obstacles moving in its path, ideally with minimum inconvenience to the user.

The Roboguide robot includes a 16-beam LiDAR sensor. In this thesis experiments will be run with a simulated 16-beam Lidar sensor used to detect the environment and the sensor measurements will be used to generate a costmap. The platform on which the experiments will be run on is a simulated logistics robot. The roboguide robot requires a user to balance the robot, and the positioning of the LiDAR and the effects of the dynamic transformation between base and sensors are still open issues at the time of writing. However, neither directly impact the requirements of the path-planner. As such, a simpler and more stable robot is chosen for testing. This is done to ensure the test results are as unaffected by outside factors possible, to best isolate the path-planners' characteristics.

1.2 Problem statement

The goal of this thesis is to evaluate the performance of three selected path-planners in a dynamic environment, particularly their suitability in a guide robot for the visually impaired. The challenges posed by dynamic obstacles are also analyzed, and an approach for future work is presented. To achieve this, a set of simulated tests are implemented, with each test case representing a simplified but common scenario a robot may encounter in a dynamic environment. Metrics used in evaluation focus on safety of the operator, how intuitive the movement is for the operator, as well as cumulative distance and time to destination.

While ideally a mobile robot would be able to function in a perfectly known location without any disturbances, that is rarely possible in a real-world application. As such, a robust and safe system must be able to dynamically react to unknown obstacles or other factors. It is vital that the reactive actions do not endanger the robot, environment or bystanders. In addition, however, many mobile robots have mission-specific requirements. For a service robot, for example, it is of almost equal importance that reactive measures are both comfortable and easy to follow for the user. Thus,

when evaluating a reactive system, it is not enough to check whether the goal is reached or not, but to also factor in implementation-specific abstracts such as comfort as well.

Local path-planning aims to react to obstacles blocking the global path. To achieve this, the path-planner must be able to model its environment in a way that allows for qualitative comparison between possible paths. For this thesis, a costmap-based approach has been chosen. By calculating a local costmap, the mobile robot deforms the trajectory locally to avoid the obstacle while still staying as true to the original path as is reasonable within given constraints. By doing so recalculating the entire global path can be avoided. Local path-planning is computationally lighter, affects energy and time-cost constraints less than a completely new global path, and is a necessity when given a path in cases of pre-recorded trajectories.

Static path-planning is a heavily researched subject and multiple viable solutions exist[5]. However, with dynamic environments, various new problems arise. The addition of dynamic obstacles necessitates some form of predictive analysis, to ensure that a path that is currently viable is viable in the future as well. Prediction creates a new temporal dimension in which a path-planner must explore to find a valid path, and this alone has a major impact on calculation loads. Predictive measures also are rarely, if ever, completely certain. Simple predictions simply assume that whatever dynamic action the obstacle is currently committed to, it will continue to do so. They may also incorporate simple *a priori* known goals or restrictions, such as collision avoidance[6]. In a realistic environment such as a crowded sidewalk, the behavior of different actors on it can follow rules complex enough to appear largely random for modern predictive algorithms. As such, an ability to react to changing circumstances is a necessity. Since the rules that obstacles follow in such an environment are, in general, impossible to clearly state, the robot navigating such an environment necessarily needs to be able to make predictions based on the obstacle's measured movements[7].

However, implementing an entire predictive interface to communicate with the path-planner is far out of the scope of the thesis. Using the knowledge of an obstacle's dynamic characteristics, namely its velocity, we can project a sort of "danger zone" in front of the obstacle. This is not true prediction, but a pseudo-predictive form of cost calculation that pertains directly to the cost calculation within a costmap and through that to the path-planner's behavior around dynamic obstacles. With this, the thesis aims to assess the viability of different path-planning approaches in a dynamic environment. As an additional goal this thesis aims to establish key problems that arise from dynamic environments in conjunction with traditional path-planning and suggest a method of accounting for them. Concisely, this thesis

aims to provide an answer to which path-planning approaches are best suited for a mobile, primarily pedestrian indoor and outdoor mobile robot in a dynamic environment, and provide suggestions on how to solve problems related to dynamic obstacles within the robot's path.

1.3 Structure of the Thesis

This thesis consists of eight chapters. In addition to the Introduction chapter the structure of this thesis contains 7 chapters, described here. **Chapter 2: State of the art** consists of the background and state of the art related to mobile robotic path-planning in a real environment. **Chapter 3: Environment** consists of the testing environment setup within the simulator, as well as the evaluation criteria used to evaluate test results. **Chapter 4: Methods** focus on the methods used to create the test and on the implemented additional modules for testing. **Chapter 5: Implementation** defines the simulator testing environment. **Chapter 6: Results** consists of the results and plotted paths from the tests. **Chapter 7: Analysis** contains the analysis of results and speculation on what those results imply for the future of the robot. **Chapter 8: Conclusions** consists of a final overview and wrap-up of the thesis.

Chapter 2

Robot Navigaton

To fully appreciate the requirements of a mobile platform's path-planner in a realistic, dynamic environment, it is necessary to understand entire pipeline between the first sensor reading and the final control command sent. This comprises of map generation and localization using Simultaneous Localization and Mapping(SLAM), obstacle detection from sensor data and its translation into an appropriate spatial representation, the creation of a multi-layered costmap based on the variety of information gathered from the state of the immediate environment, and path-planning. The costmap functions as a simplified representation of the robotic platform's surroundings, and allows for fast and functional planning. In addition, a robot typically contains a semantic detection and tracking system. However for the purpose of this thesis, we will use an automatically generated detection value, as the aim is not to evaluate the functionality of the detection software.

2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping(SLAM), is a problem defined by the need to both map an environment and localize the robot within the environment in tandem[8][9]. The fundamental challenge of SLAM is the on-line estimation of both robot location and landmark distances to form a cohesive trajectory and map throughout the environment.

The core problem associated with SLAM is the uncertainty of the absolute location of both the robot and landmarks. All measurement methods have inaccuracy, and the drift of the measurement error accumulates over time. Any practical applications face the problem of how well they can trust what their measurements reflect the true position of the robot and landmarks. Especially robot location measurements typically rely on particularly uncertain

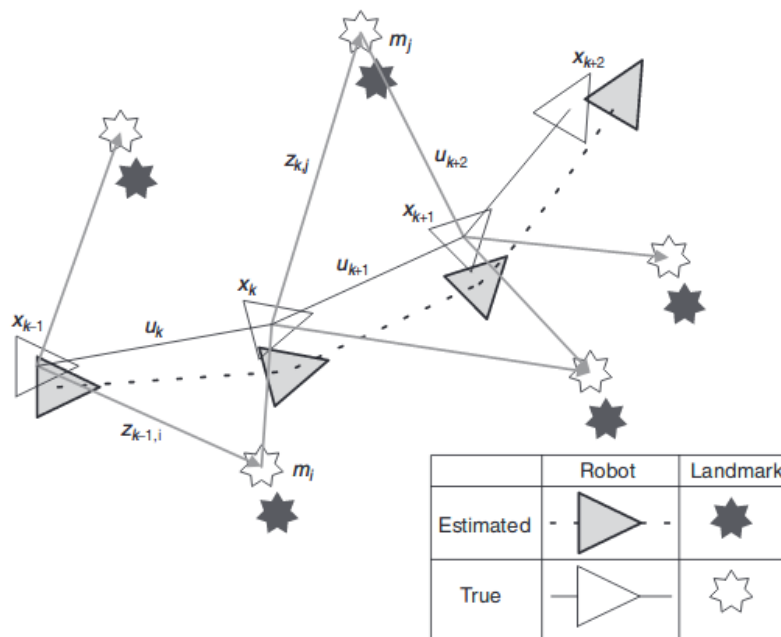


Figure 2.1: The SLAM problem, in which both an estimation of landmark and robot locations is required. The true positions are never directly measured. Image by Durrant-Whyte et al.[8]

measurements, such as wheel odometry, and diverge significantly from the ground truth.

However, Dissanayake et al. identified an important characteristic of measurements made by a mobile robot: while absolute locations of the robot and measured landmarks remain highly inaccurate, the relative locations of landmarks are correlated, and the correlation increases monotonically as more measurements are made[10]. This is illustrated Figure 2.1, in which the relative positions of obstacles are consistent despite an error in robot location. While this answers the issue of relative location, a map that is only accurate in relation to the distance to other landmarks within it would hardly be useful.

Fortunately, loop-closure methods, such as outlined in [11], exist. The loop-closure problem states that following a re-visitation, or loop, the absolute pose and location estimate of the robot tend to be catastrophically wrong, preventing naive SLAM methods of matching the landmark measurements to each other[12]. As a consequence of this, further mapping will both

be globally inaccurate and computationally expensive, as it assumes that the location is unvisited and as such assigns it map space accordingly. By completing loop-closure by matching prominent features to known landmarks, it is possible to correct the drift, and localize the robot within the local frame of the mapped area accurately.

Combined, these characteristics of SLAM allow us to create extremely accurate map representations of the environment. For the purpose of this thesis, the following of a specific, non-changeable path is required, and as such a reliable mapping method is needed. At this stage of the robot development the platform does not rely on real-time SLAM, as the robot operates within pre-mapped areas. Mapping using the SLAM method is conducted offline, prior to testing, using a recording that was manually controlled. Graph-SLAM offers a robust solution to map creation after the initial recording operation[13]. It creates a probabilistic graph in which raw measurements are abstracted to create a type of virtual measurement that is represented within the graph as an edge[14]. The model of observations of landmarks $\mathbf{z}_{1:T}$ within the static map \mathbf{m} from measured from robot locations $\mathbf{x}_{1:T}$ is described by:

$$p(z_t|x_t, m_t), \quad (2.1)$$

which corresponds to a probability distribution. This means that a single measurement needs to be associated with the landmark most likely to be correct. This fundamentally allows for simplification in data association and makes the entire computation feasible.

2.2 Costmap generation

A costmap is a map of the environment that, as the name implies, characterizes the measured surroundings by a cost function. The cost in a costmap cell is a representation of how difficult or dangerous traversing an area is, in other words, how expensive operation within that grid is. There are many map presentations that fit within the costmap definition, including potential field representations[15] and work-based configuration spaces[16]. These are useful for environments where the terrain itself creates a variable cost to traversing, however for the purpose of this thesis it is assumed that the traversable areas or cost-free by default. As such, the thesis will focus on grid-based occupancy orientated representations.

Typically the cost can vary within a spectrum, from free space to fatal space, which is considered catastrophic to operation[17][18]. Any values assigned to a space will add a virtual cost to any path, making it simple to evaluate the goodness of a planned path within the map. A physical obstacle, such

as a wall or object, will be assigned the maximum cost, which is effectively untraversable. This is due to a collision potentially being catastrophic for continued operation. There exist methods for traversability analysis[19][20]. These are useful for off-road mobile robotics, however it can be assumed that typical indoor-outdoor robot operation environments do not have meaningful terrain-dependant incurred costs, and they can be safely dismissed.

Instead, it can be assumed that any cell within the costmap will either simply be occupied, unoccupied or unknown. If a cell is occupied, it is considered a lethal obstacle and no operation within it will be permitted. If the cell is empty, it will be considered free and operation within will not be restricted. An unknown cell is one that has not been measured. Path-planners assume that, effectively, an unknown cell is free unless a measurement shows otherwise. However, the environment used in this thesis is thoroughly mapped, and unknown cells are not factored into path-planning. Sensor measurements will be used to determine whether an obstacle is within a cell, and the whole cell will be flagged accordingly[21]. The drawback to flagging the entire cell regardless of how much of it is occupied is plain: there is a remarkable loss of resolution and data on surface shapes. However, since a costmap's purpose is to facilitate path viability, and data that pertains to path planning is not lost, the data reduction is not problematic.

A modern approach to creating costmaps is to create a composite costmap consisting of a global costmap from the saved representation of the environment, and a smaller, local costmap maintained with current measurements[22]. This allows for long-range path evaluation that is accurate so long as the environment has not changed. The local costmap will in turn record any changes within the environment. This will allow flexible local reactionary actions while retaining a global long-term actionable plan.

In addition to distinct global and local costmaps, Lu et al.[22] propose semantic composite layers. These can contain information not directly extrapolatable from the raw costmap. In particular, it can be used to inflate the cost of obstacles. A non-lethal but non-zero cost near obstacles function as a repulsive force, not strictly preventing a mobile robot from operating near an obstacle but causing any path-finding algorithms to prefer avoiding proximity. This both creates a safety buffer from collisions, especially if the obstacle is dynamic, and in cases where humans actively participate keeping a distance will be less invasive. Layers can further be used to direct robot behavior with socially or culturally motivated costs. Examples of use cases for this would be right-sided traffic in hallways or caution zones near corners or door.

2.3 Path planning

Path-planning is a vital part of any mobile robot's navigation system. Once a robot has perceived its environment and localized itself, it needs to be able to create and, more importantly, evaluate a path to follow to its goal[23]. As a cognitive decision making function, path-planning is perhaps the closest system module to human abstraction in a traditional mobile robot system, and also one of the hardest to accomplish safely and at a level of quality comparable or higher than human control. More importantly, path planning has been shown to be NP-complete, making generic solving of the path computationally expensive[24].

Pathfinding and path-planning are often used interchangeably. For the purpose of this thesis, we make the distinction of pathfinding comprising of the core algorithm that generates a path, such as the A* algorithm[25], and path-planning adding planner-specific constraints so as to optimize the path for the specific use-case. Many path-planners actively practice pathfinding, and in sufficiently constrained cases the pathfinding effectively generates what can be used as a path plan. A path planner takes a path found by a pathfinding algorithm and imposes additional constraints. This can be done with dynamics, temporal or some additional semantic cost data, and the result is typically not the shortest path, but the optimal path based on the robot's inherent functionality.

Path planning can be coarsely categorized to global and local path planning[2]. A global planner generally requires the environment to be completely known and static. That is rarely the case for realistic environments, so planning is typically an approximate path. For the purpose of this thesis, however, global path planners are ignored. Pre-recorded paths through a known environment will be used, both for replicability as well as simplicity.

A local path planner is an implementation of path planning which, when a mobile robot is following a global path encounters an obstacle upon its path, is able to react locally and quickly to the threat of collision. Instead of recomputing the entire path to the goal, which can be computationally expensive and result in a differing path, it is better to compute a local path around the obstacle and back to the global path plan[26]. Additionally, should the robot drift from desired path, the local path planner works to return it to its trajectory in a computationally reasonable time.

There is a reasonably well-documented case for the use of genetic algorithms(GA) as more optimal than classical path-finding and motion-planning approaches[2][27][28]. However, the use of GA is still largely theoretical, where classical approaches, or approaches that use concrete heuristics such

as direct distance, have seen success in real use. Furthermore, existing implementations for classical approaches exist more readily, and the state-space of local path planning in a 2D costmap environment is limited enough that computational feasibility is not a problem. For this reason, this thesis will focus on the use of classical methods.

2.3.1 Reasoning

This selection of local path-planners was chosen due to existing research which gives credence to their validity in a practical application. Additionally each planner has a different approach: dynamics-based(DWA), distance-based(Eband) and time interval -based(TEB). The aim of these choices is to give a meaningful understanding on each approach's validity in dynamic obstacle avoidance.

2.3.2 Dynamic Window Approach

The Dynamic Window approach (DWA) is a well-known path-planning approach. Unlike many methods, which approach the planning as a purely geometric challenge in finding a series of robot poses which do not overlap with obstacles, minimize cost, and reach the goal, DWA takes into account the dynamics of the mobile platform[29].

DWA approximates the possible trajectories of a platform as a series of circular curves. Each curve corresponds to a time interval t ,

$$t \subset [t_0, t_n]$$

,with n being the number of time intervals(effectively the time horizon of the planner). Each curve also has a corresponding pair of velocities (v_i, ω_i) . For computational optimization, for each curve corresponding to time intervals $[t_1, t_n]$ it is assumed that the velocity pairs stay the same, i.e. acceleration is 0. This is possible because effectively the robot will have a constant velocity until a new control command is given, and because the calculation of velocities is completed at the beginning of each time interval anyway, thus making constant recalculation redundant.

To calculate possible dynamic combinations for a time interval, it is necessary to define the admissible velocities for the interval. Admissible velocities are defined as a set of velocities V_a such that the platform can avoid collision with an obstacle on the trajectory by stopping in time. By defining $dist(v, \omega)$ as the distance to the nearest obstacle upon the curve in question, a set of admissible velocities can be defined as

$$V_a = \left\{ (v, \omega) \mid v \leq \sqrt{2 \cdot dist(v, \omega) \cdot v_b} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \dot{\omega}_b} \right\}$$

$\dot{v}_b, \dot{\omega}_b$ are the decelerations due to braking. This gives us the valid velocities within the velocity space.

The admissible velocities are all the velocities that will guarantee a collision-free trajectory, however, not all velocities are necessarily reachable. The acceleration of the robot is limited by the capabilities of its motors, thus, not all velocities can be reached with a given time-frame. The area of the velocity space which is reachable by the robot during one interval is called the dynamic window. The dynamic window is defined as

$$V_d = \left\{ (v, \omega) \mid v \in [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge \omega \in [\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t] \right\}$$

where $\dot{v}, \dot{\omega}$ refer to the possible accelerations applied to the velocities and v_a, ω_a refer to the actual velocities.

The restricted space V_r , which can be formulated as

$$V_r = V_a \cap V_d$$

defines the set of velocities which can be used for the path-finding algorithm.

Some research on using DWA in conjunction with dynamic obstacles has already been done, and the results are promising[30][31][32]. DWA has inherent positive qualities for dynamic obstacle avoidance: by analyzing future time intervals and at least some of the robot's state dynamics at those points, the approach integrates a form of prediction, which is a core principle in successful reactions to dynamic obstacles. This approach through dynamics is the principle reason for including DWA in this thesis.

2.3.3 Elastic Band

Elastic Bands (Ebands) are a proposed method of path-planning. The Eband approach creates what are called elastic bands between start and goal positions. An elastic band planner takes a rough grid-based plan generated with any kind of simple planner, and deforms it to more smoothly avoid obstacles[33]. An elastic band is characterised by overlapping bubbles along the trajectory, where each bubble is bigger than the footprint of the robot. This assumes a circular footprint for the robot, which may result in a limited configuration space for a robot as outlined in [34]. However in a dynamic environment, near-collision navigation should be avoided, and the reduction in configuration space should not present a major issue.

An elastic band bubble can be represented as a subset of free space $B(b)$, where b is the configuration of the robot. To find the exact parameters of the bubble $B(b)$, we define the distance $\rho(b)$, the distance to the closest obstacle

at the configuration point b . We define it as:

$$B(b) = \{q : \|b - q\| < \rho(b)\}$$

Thus, the space of $B(b)$ defines an area of free space in which the robot can move in without colliding with an obstacle. An elastic band is not limited to a point representation of the robot, however, as the dimensionality of the bubble of free configurations is defined by the complexity of the configuration space.

The series of bubbles, effectively a path, will then be deformed by a repulsive force from obstacles to give better clearance to the robot. As such, the result is a path of overlapping bubbles which each consist of configurationally free space. Due to the overlap, it is guaranteed that each bubble is reachable from the previous one, and it is not necessary to compute the entirety of the free space in the environment. The resulting path is the shortest collision-free path created.

Though separate concepts, DWA and elastic bands are not mutually exclusive. They can be implemented together, so as to impose additional constraints with the elastic band, as seen in the work of Phillippsen et al.[35]. Additionally, the ROS package `eband_local_planner` contains a ready-made implementation that utilizes a DWA-based algorithm[36].

2.3.4 Timed Elastic Band

Timed elastic bands (TEB) have been proposed as an elastic band-based approach that takes into account dynamic constraints[37][38]. Similarly to elastic bands, timed elastic bands also compute a series of bubbles that create a free space continuum between the start and goal. However, where traditional elastic bands only take into account absolute distance, timed elastic bands do not look for the shortest path but the fastest path by time travelled.

Much like regular elastic bands, TEBs are a series of positions:

$$Q = \{x_i\}_{i=0,1,\dots,n} \in N$$

where x_i is a robot pose. Additionally, we consider time intervals between consecutive poses within the series:

$$\tau = \{\Delta T_i\}_{i=0,1,\dots,n-1}$$

Together, these make up a series of tuples:

$$B := (Q, \tau)$$

This bubble sequence is then optimizable based on necessary constraints, such as robot dynamics and temporal or spacial restraints.

TEB has been shown to be a generally good local planner for non-dynamic situations[39]. However, the exact way a time-constrained planner will react to a dynamic scenario where optimal trajectories are temporally dependent is difficult to predict. There have been some promising results with modified TEBs[40], in which the use of semantic knowledge of target obstacles is used to supplement direct measurements. These modified TEB implementations are, however, heavily use- and environment specific, lacking a generic applications.

2.4 Evaluation criteria

The performance of a path-planner is not straightforward to assess. Metrics may be simple, but their usefulness for a specific application is not guaranteed. A safety-critical industrial robot values safety, and thus avoidance of collisions, to be paramount, where as a consumer vacuum robot can acceptably collide with obstacles. However, getting stuck or lost easily is unacceptable, as its practicality suffers. A service robot may value user comfort over collision free paths, as it is reasonable to assume moving agents are both capable of spotting and able to avoid the robot.

Traditional evaluation criteria for path-planning in static environments focus on distance to obstacles, either directly from the shortest distance to the closest obstacle or through path-cost-analysis [41][42][43]. This is a reasonable approach when the environment can be considered static and is considered known at the the time of planning, as the main risk factor to a robot is its proximity to an obstacle. However, in the case of a dynamic obstacle, the location of an obstacle can not be determined with certainty when planning. In the test cases presented here, the dynamic obstacles will intersect with the path, and simply following the initial trajectory will guarantee a collision. As proximity with the obstacle is a design feature for testing, actual collision avoidance and smooth navigation are both more important than maintaining a constant minimum distance from an obstacle.

Some criteria for static environments have been suggested by [42]. In particular, the Narrow Road(NR) approach is a useful criterion for use with a dynamic obstacle. However, the criterion are geared towards static environments, and thus incorporate distances to near obstacles as cost factor. Additionally, it evaluates the planner output as a plan and not an actualized path the robot moved along, and so does not factor in behavior typical for a robot, such as localization issues. The criterion will be simplified to better

evaluate the meaningful differences of the path planners. To minimize the effects of localization errors and uncertainty of the position of the robot, we will sample only every tenth pose along the recorded path. This is done to mitigate the effect of slight discrepancies of the pose of the robot, which would accumulate over time. The criterion used will be the Fluctuation Ratio(FR), defined as:

$$FR = \frac{\sum_{i=2}^{pathsize} |\theta_i - \theta_{i-1}|}{pathsize}$$

Where θ is the heading of the robot. The result, in layman's terms, is the average change in heading between the heading of consecutive poses. This gives an estimate of how often the robot changes heading to get to the goal. Alone, this is a meaningless metric but it is useful for comparing different path-planner implementations.

Any distance travelled by a robot incurs a cost. This may be measured in time lost on unnecessary detours, wear and tear on the physical components due to unnecessary operation, and the confusion or irritation of human users due to unintuitive path choice[44]. A shorter path, provided it's safe and within semantic constraints such as traffic flow, is preferred almost universally. For simplicity and efficiency, not every pose along a traversed path needs to be accounted for. As such, every tenth pose will be sampled. Cumulative Distance is defined as:

$$CD = \sum_{i=2}^{pathsize} \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2}$$

Where X and Y denote the robot's coordinates.

A third good metric to evaluate performance of a robot is time to destination. For a logistic robot, robot productivity is inherently tied to time spent in navigation: the shorter the time spent travelling, the faster the robot is able to accomplish a task and consequently begin a new task. For a service or assistive robot, the shorter the time spent navigating, especially if it results in less idle time, improves the experience of the user. As such, a reasonable evaluation criterion is how fast the robot arrives at the destination. The robot, associated systems, and scenario do not change between test iterations. As such, the only factor to affect time to destination is the path planned and the local path-planners ability to react to changes in the environment.

A final and simple evaluation criteria is simply how often the path-planner was successful. Success is defined as the successful planning and implemen-

tation of a path that allows the robot to reach the desired goal without colliding, getting lost or otherwise stuck in a position it is unable to navigate away from. Success Rate(SR) is defined as the ratio of successful runs to all runs.

Chapter 3

Implemented methods

This chapter outlines the implementation of a teach-and-repeat path creation algorithm and a Gaussian cost model for a dynamic obstacle. These implementations play a key role in the testing of the path-planners: the teach-and-repeat algorithm's output functions both as a global path-plan and the path for the dynamic obstacle, and the Gaussian cost functions as a simplified dynamic prediction model.

3.1 Implemented teach-and-repeat method

The use of global path-planning in many robot navigation scenarios can be problematic. A path-planning algorithm will typically attempt to minimize distance travelled, and accounting for more abstract costs is often non-trivial. For example, crossing over to the left side of the sidewalk or cutting across a curve may yield the fastest path to the goal, however, typical traffic customs will result in the robot travelling against traffic or otherwise behaving socially unacceptably. This will not only be inconvenient for pedestrians, but can also create dangerous situations. Automatic semantic labeling of maps to enforce certain types of appropriate behaviours for different areas of the environment has been researched[45], but it is resource-intensive and unrefined. Adding semantic information to a map by hand is possible, but is labour-intensive and not generic. Additionally, each generated global plan adds a minor, though not negligible, computational cost, even if the path has been travelled before.

In many scenarios it's possible to plan a single path between two points, teach it manually to a robot, and use it as global path for navigation. This path will abide by semantic rules as long as they were followed when the path is taught, and social conventions can be recorded within. Utilizing

localization and a series of robot poses within an existing map, it is possible to create such a path with a high degree of accuracy. This method is called teach-and-repeat[46].

Once a map has been created, the robot is manually moved along a desired path. The pose of the robot $x = (pos_x, pos_y, pos_z, pitch, yaw, roll)$ is recorded once every time interval t . As the cost of recording is minimal, using the shortest time interval that still makes a meaningful difference is preferable. The resulting series of poses

$$P = \{x_i, t_i\}$$

form a structured path that can then be given to the robot as a global path.

3.2 Modelling the cost surrounding an obstacle with Gaussians

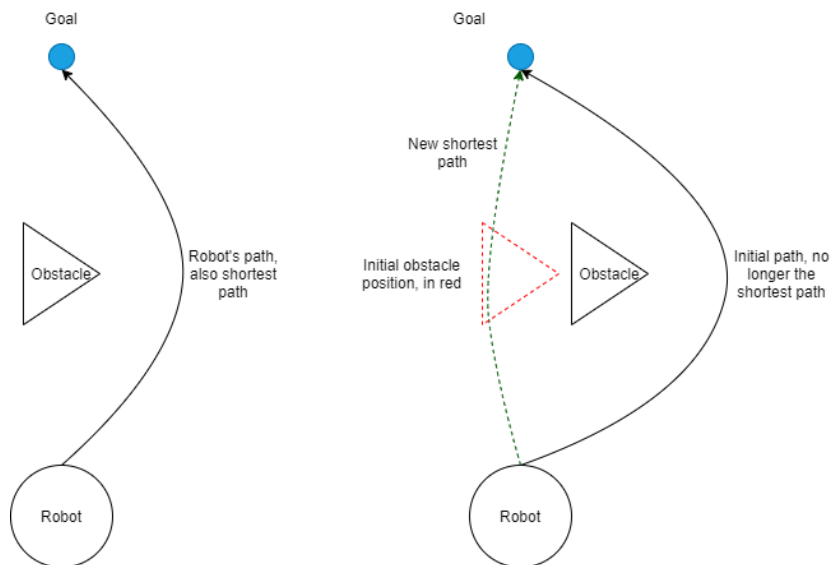


Figure 3.1: An example of a situation in which a robot prefers a longer path to in front of the obstacle over a shorter path behind it due to the lack of predictive capabilities. In the second instance, stopping or moving straight towards the obstacle would avoid collision and keep the path short.

With modern processors, local path-planning is computationally feasible even in real-time. This has been shown to be effective with unexpected static

obstacles, allowing the robot to react to them without a noticeable delay[47]. Likewise, planning for dynamic obstacles is computationally feasible. However, the dynamics of the obstacle must be taken into account, otherwise continuous recalculation of the path may result in the robot attempting to deform the path further and further in front of the obstacle instead of passing it from behind as shown in 3.1.

Multiple approaches to tackle this issue have been proposed. These methods are based on time horizon exploration, where, using measurement data, predictions are made for obstacles and paths are evaluated by temporal viability[48][49]. The proposed approaches, while creating better results overall, are complex, difficult to implement, often require particular sensors, and more importantly computational costs grow exponentially with a new dimension to explore. Time horizon exploration offers effective tools for far-reaching predictive planning, but in many cases such planning is unnecessary. While on-road predictions usually retain good accuracy over time, indoor or urban applications tend to become inaccurate as dynamic actors such as pedestrians behave erratically.

For this thesis, time horizons will not be considered. Instead, to account for dynamic behaviour, we will create a "cost bubble" around obstacles identified as moving. Using the velocity of the obstacle, the cost bubble will be extended towards the direction the object is moving towards. This will create a shadow within the costmap in front of the obstacle, resulting in the planner avoiding the immediate vicinity of a moving obstacle. This method does not predict the future positions of an obstacle, and does not account for meaningful time steps. The cost bubble does, however, raise the uncertainty of dynamic objects, and while it does not explicitly predict future poses of the obstacle, it makes moving to close proximity of them expensive.

To create a cost bubble, we will use an asymmetric Gaussian function, as outlined in[50]. A regular 2-dimensional Gaussian can be defined as follows:

$$f(x, y) = e^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)} \quad (3.1)$$

This is symmetric about coordinate axes. However, we want to create a Gaussian that is only symmetric about one axis, namely, the direction. Additionally, the axis and heading will not necessarily align, so it is necessary to take into account the direction of movement. To find the heading, we use:

$$\theta = \text{atan2}(v_x, v_y) \quad (3.2)$$

The algorithm for computing the Gaussian cost at a discrete costmap

point (x, y) is defined by Kirby[50]. The Gaussian cost is outlined in 1.

```

for  $(x, y)$  do
   $\alpha \leftarrow \text{atan2}(x - x_c, y - y_c) - \theta + \pi/2$ 
  normalize  $\alpha$ 
   $a \leftarrow (\cos(\theta)^2 / (2\sigma^2) + (\cos(\theta)^2 / (2\sigma_s^2))$ 
   $b \leftarrow \sin(2\theta) / (4\sigma^2) - \sin(2\theta) / (4\sigma_s^2)$ 
   $c \leftarrow \sin(\theta)^2 / (2\theta^2) + \cos(\theta)^2 / 2\sigma_s^2$ 
  return  $e^{-(x-x_c)^2 + 2b(x-x_c)(y-y_c) + c(y-y_c)^2}$ 
end

```

Algorithm 1: Gaussian cost for (x, y) in relation to the center point of the detected obstacle. (x_c, y_c) refer to the center points of the obstacle, σ is variance with regards to heading, and σ_p is variance perpendicular to the heading.

σ and σ_s are calculated when the obstacle is detected and tracked, and depend on the implementation of the detector and tracker. The detector and tracker implementation is outside of the scope of this thesis, so we will use the default initial value provided by the data struct. This only incorporates sensor measurement variance, and the effects of velocity to the cost of a particular point are not taken into account. To account for this, an additional phase needs to be added in which the velocity of the obstacle affects the Gaussian cost. To achieve this, we incorporate a calculation in which the variance in the direction is amplified by velocity factor V_{factor} :

$$\sigma = \begin{cases} \sigma * V_{factor}, & \text{if } \sigma \geq 0 \\ \sigma, & \text{otherwise} \end{cases} \quad (3.3)$$

The velocity factor is simply a factor that scales with the velocity. For the purpose of this thesis, we do not need to factor in the dynamics of momentum and acceleration, as the velocities will not exceed a relaxed walking pace and the momentum of potential obstacles for the robot do not exceed that of an average pedestrian. As such, we can assume the velocity factor should scale linearly within the scope of this experiment.

Additionally, for the sake of simplicity and for better results, we will assume variances to be small and constant. As the tests focus on the performance of the path-planner, not on the performance on the system as a whole, and as the performance of a detection and tracking system cannot be assumed to be entirely comparable to a real environment when implemented within a simulator, these assumptions and approximations serve to minimize noise and unrelated issues within the tests.

Chapter 4

Experiments

A mobile robot implementation is typically a complex system comprised of multiple inter-dependant modules. As these systems are not perfect, it can be a challenge to pinpoint the cause of phenomena observed while the robot is operating. As such, accurate evaluation requires careful preparation to minimize or even eliminate interference caused by unrelated module interactions. This chapter outlines the steps taken to isolate the path-planner as effectively as possible from the effects of outside influences, as the nature of the interactions of modules directly interacting with the path-planner. Furthermore, the experiments and results are presented.

4.1 Test environment

Mobile robotics is a varied field, and the environment defines what is required of the robot navigating it. In the case of a robot expected to move indoors or in an urban environment, ability to react to unexpected and dynamic obstacles is of paramount importance. Spatial requirements such as minimum clearance create a difficult environment to navigate where risky situations may be unavoidable. Pedestrians that both move unpredictably and may react unexpectedly to the robot movement create random, and potentially dangerous, situations.

Urban navigation will often impose constraints such as strict tolerances to global paths, limited fields of view due to obstacles, and a preference for smooth movements for social convenience. As such, testing should be able to account for these factors. For reproducibility and ease of implementation, testing was conducted with the Gazebo simulator. The Gazebo simulator allows for realistic simulation not only of the environment, but of the dynamics and physical capabilities of the robot and its sensors as well. Using

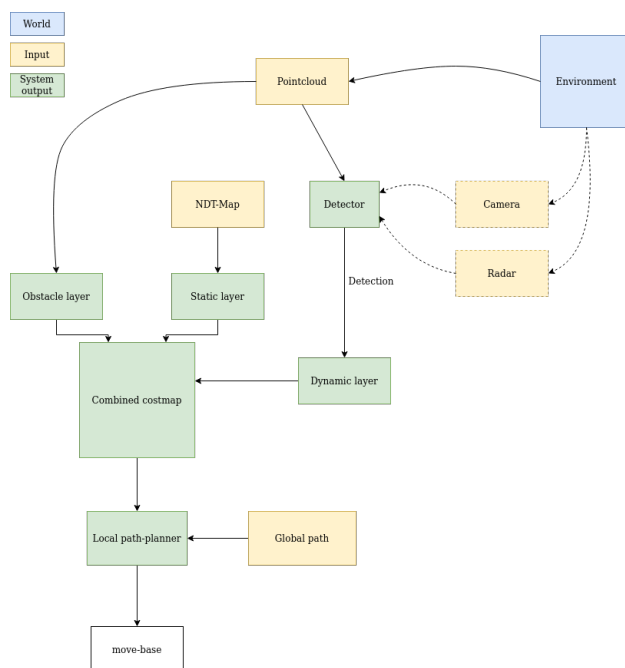


Figure 4.1: Flowchart of a real robot environment

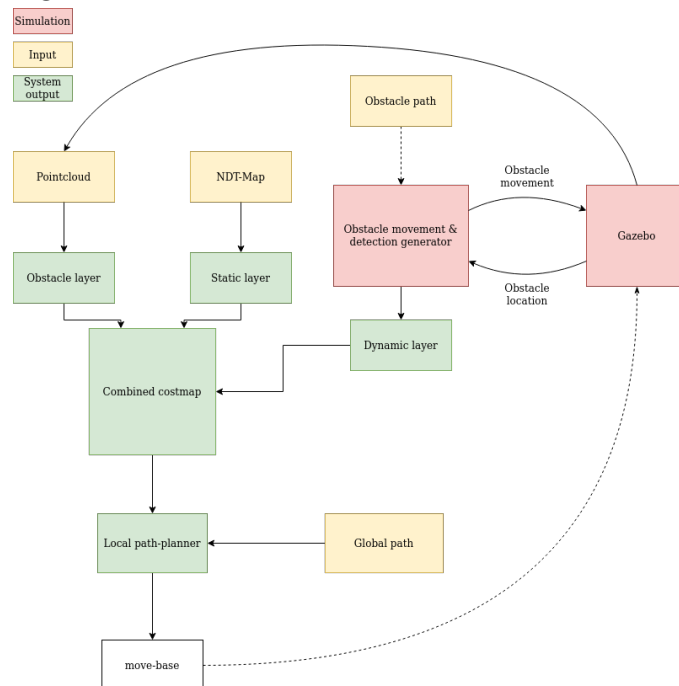


Figure 4.2: Flowchart of the test environment

the 7.16 version, it is possible to implement basic moving objects that serve as rudimentary dynamic obstacles.

A generic cube-like box served as the dynamic obstacle in testing. A separate ROS node updates the obstacle position along a prerecorded path. The same ROS node also generates a detection based on the updated location of the obstacle. The motivation for this was that the detector depends on a combination of camera images and LIDAR pointclouds, the first of which is both difficult to implement in a simulator and requires extensive retraining of the neural network for it to function with simulator images.

Figures 4.1 and 4.2 show flowchart models of a mobile robot within a real environment and a simulated test environment. The core functionality in both environments is effectively the same, as they share software modules. The main difference is the reliability of the sensors: simulated LIDARs do not produce noise and errors that are inherent to any physical sensor. Thus it should be noted that any simulation data is in all probability cleaner and more ideal than running a system in a real environment.

4.2 Test setup

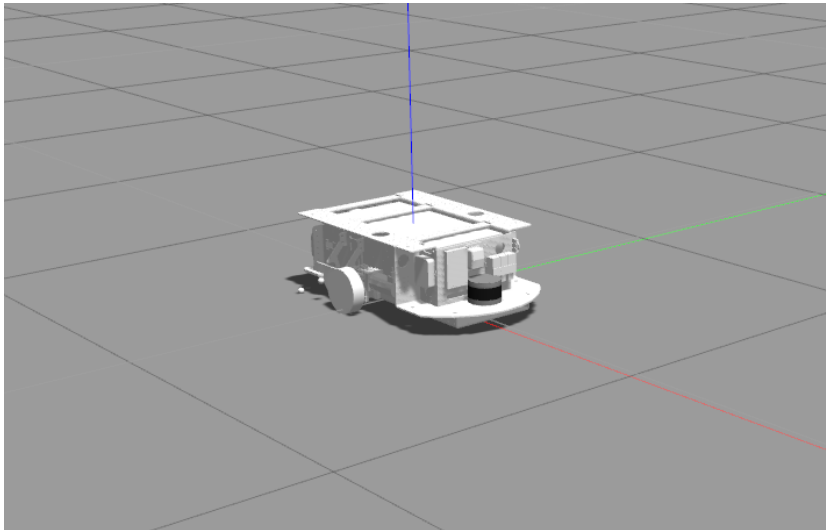


Figure 4.3: The model of a GIM Speedster logistic robot within the Gazebo simulator.

The test setup is designed to take into account various potentially hazardous situations for a mobile robot within a dynamic, indoor or urban out-

door environment. A simulated GIM Speedster logistics robot will be used as the testing platform. As the logistics robot is stable and already functional, it serves as an adequate substitute for other differential drive robots. A simulation image of the robot is seen in 4.3.

The testing will consist of six different situations, each comparable to a realistic potentially hazardous event:

1. A baseline test with static obstacles
2. A dynamic obstacle moving perpendicularly across the trajectory to behind a wall
3. A dynamic obstacle moving directly towards the robot along the robot's trajectory
4. A dynamic obstacle moving perpendicularly across the trajectory from behind a wall
5. A dynamic obstacle moving diagonally across the robot's trajectory
6. A dynamic obstacle first moving towards the robot then turning away from the trajectory

Using teach-and-repeat methods as proposed in [51], a path is created for both the dynamic obstacle and the robot itself. For the test, the dynamic obstacle is considered "stupid", in that it will not react to its environment in any way. It will simply be moved along the path recorded for it. The robot will be given a different path as a reference trajectory which acts as a global path-plan. The test robot will dynamically create a local path-plan according to which the control system will navigate. Consequently, the robot will deviate from the global path should an obstacle be present.

Furthermore, a dynamic cost implementation, referred to in 4.2 as the "dynamic layer", will create a bubble of heightened cost around the dynamic obstacle. This aims to promote keeping a distance from the obstacle, and thus avoiding potentially dangerous situations.

4.2.1 Robotic Operating System

The Robotic Operating System(ROS) is a open-source middleware software framework for the development of complex robotic systems. Not an operating system despite what the name implies, it functions as a platform for easy communication between software components and central storing of

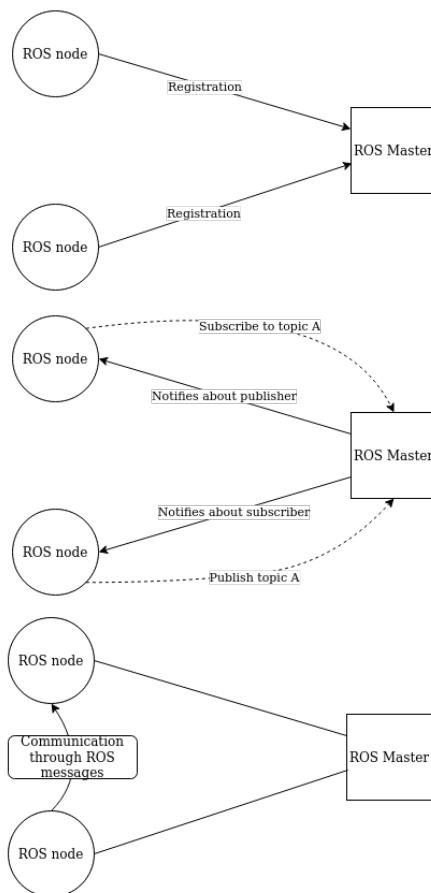


Figure 4.4: Illustration of two ROS nodes establishing communication through a ROS Master.

parameters[52]. Furthermore, due to its open-sourced and collaborative nature, ready-made implementations for many tools exist and design for ROS ensures modularity and generality.

For the purpose of this thesis, it is necessary to understand ROS node functionality, ROS messages and rosbag recording, which are outlined below:

- **ROS Master** provides a method of registering and looking up ROS nodes. Nodes communicate through subscribed and published topics, through which they either publish or receive information. They make use of named publishers and subscribers. The ROS Master's function is not to act as a middle man for the messages, but to help them locate each other, after which a connection is made between the nodes and

communication happens directly between the nodes. Rosbags are also passed to nodes through the ROS Master.

- **ROS nodes** are executables that function as independent, though interconnected, graph cells of a ROS system. A node is the core computational source of a ROS robotic system. Ideally a ROS node will separate functionality from ROS connectivity, making it both modular with other ROS systems, as well systems that do not utilize ROS.
- **ROS messages** are handled through a publisher-subscriber system. A ROS message is a data representation in a simplified description language. A message is given fields, which themselves can be data types or other message types. These messages can be passed to a publisher within a ROS node, from which it is published. Any published topics are available for subscription by any other ROS nodes within the system, making data sharing between components simple and effective.
- **ROSBags** are the recorded message topics, allowing for any data recorded within a ROS system to be saved and played back at a later time. This makes it possible to not only work offline, it creates a simple way to recreate exact conditions.

The initialization of inter-node communication is illustrated in Figure 4.4.

4.2.2 Simulation platform

The testing platform was Gazebo 7.16. The main motivation for this is an existing implementation of the Speedster robot for the specified version of the simulator. To simplify testing and reduce localization errors, the test area consists of a wall that runs parallel to the global path and a wall that is perpendicular to the path near the goal. To reduce processing requirements, LIDAR point detections past a threshold of 20 meters will not be considered. The walls offer sufficient points to localize the robot while also restricting the robot's movements, simulating realistic environmental restrictions. As neither wall is directly in the path of the global path, they merely restrict the correctional movements of the robot attempting to avoid the dynamic obstacle.

The dynamic obstacle is implemented through a ROS node that updates the location of both the physical box within the simulator and the detection of the box. Prerecorded paths are implemented with a ROS Path navigation message. The robot is manually controlled along a desired a path. The path

is recorded with the same implementation of teach-and-repeat as the global path for the robot.

Each test is launched by initializing the simulator and all robot systems and given the execution command. A test is allowed to run for a period of 2 minutes, which, should the test succeed, is enough time for the robot to reach the destination. Ten test runs are run per test, and all results are recorded regardless of success of the test.

4.3 Experiments

4.3.1 Baseline

To provide a baseline understanding of how the local path-planners work in a static environment, a simple test is conducted. The simulation world is the same as within the actual tests. Four static obstacles that do not directly intersect the global path are added. As with the test scenarios, each of the baselines are run ten times.

As shown in Figure 4.5, given a static environment each local path-planner performs reasonably well. Importantly it can be noted that each path-planner's realized paths are nearly identical, and differences can be attributed to initial localization errors. The paths are typical of the path-planners' properties.

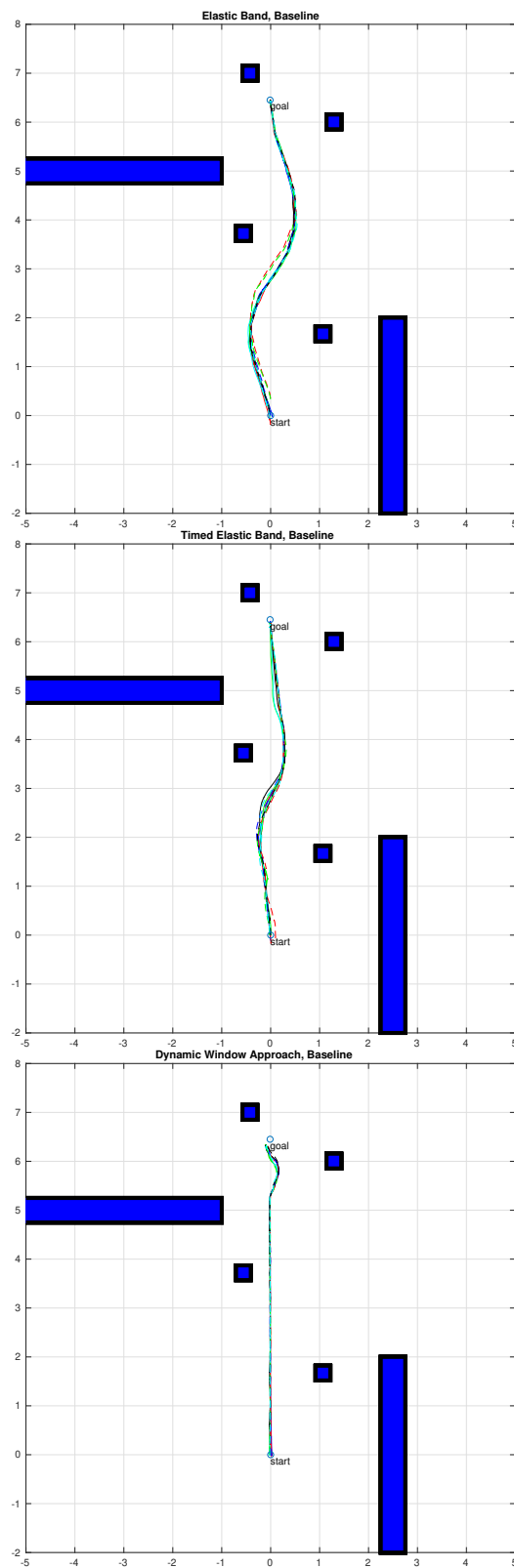


Figure 4.5: Baseline test plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

4.3.2 Tests

Test 1, found in Figure 4.6, had the obstacle moving perpendicular to the robot’s desired path, from left to right. Initial analysis of the plotted paths shows that both the TEB and DWA planners both follow a relatively straight route to the goal point. The EB planner initially attempts to pass the obstacle in the direction of the obstacles movement, until encountering a wall and is forced to re-evaluate it’s path. The TEB planner shows a lot more discrepancy in paths compared to both the EB and DWA planners.

As seen in Table 4.1, both EB and TEB planners show a comparatively high FR. As the FR takes into account changes in heading between very close points along the robot’s trajectory, a high FR does not necessarily mean the plotted path itself visibly fluctuates. EB failed to reach the goal point on two occasions. This was caused by phantom measurements from the obstacle that it failed to clear and thus couldn’t plan a path to the goal.

Table 4.1: Evaluation scores for Test 1

Path-Planner	FR	CD	Time	SR
EB	0.0549	6.98m	46.0s	0.8
TEB	0.0518	6.74m	46.7s	1.0
DWA	0.0344	6.47m	42.4s	1.0

Test 2, plotted in Figure 4.7, saw the obstacle moving directly towards and along the global path of the planner. The EB planner failed categorically to react as the position of the obstacle changed and consequently the local path-planner’s path was constantly re-evaluated. The DWA planner failed occasionally and the TEB planner managed to avoid the obstacle each time.

The evaluation scores are shown in Table 4.2. The distance travelled by the TEB planner was larger, but it reached its destination faster.

Table 4.2: Evaluation scores for Test 2

Path-Planner	FR	CD	Time	SR
EB	N/A	N/A	N/A	0
TEB	0.0174	10.22m	54.4s	1.0
DWA	0.0161	9.80m	77.0s	0.4

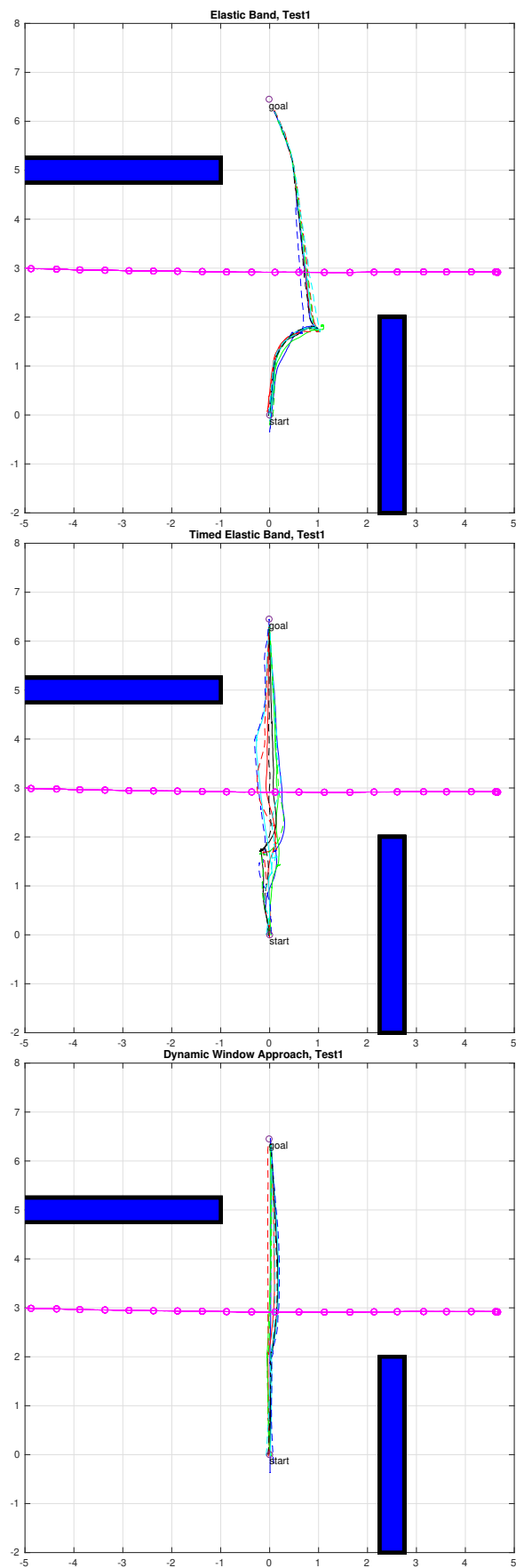


Figure 4.6: Test 1 plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

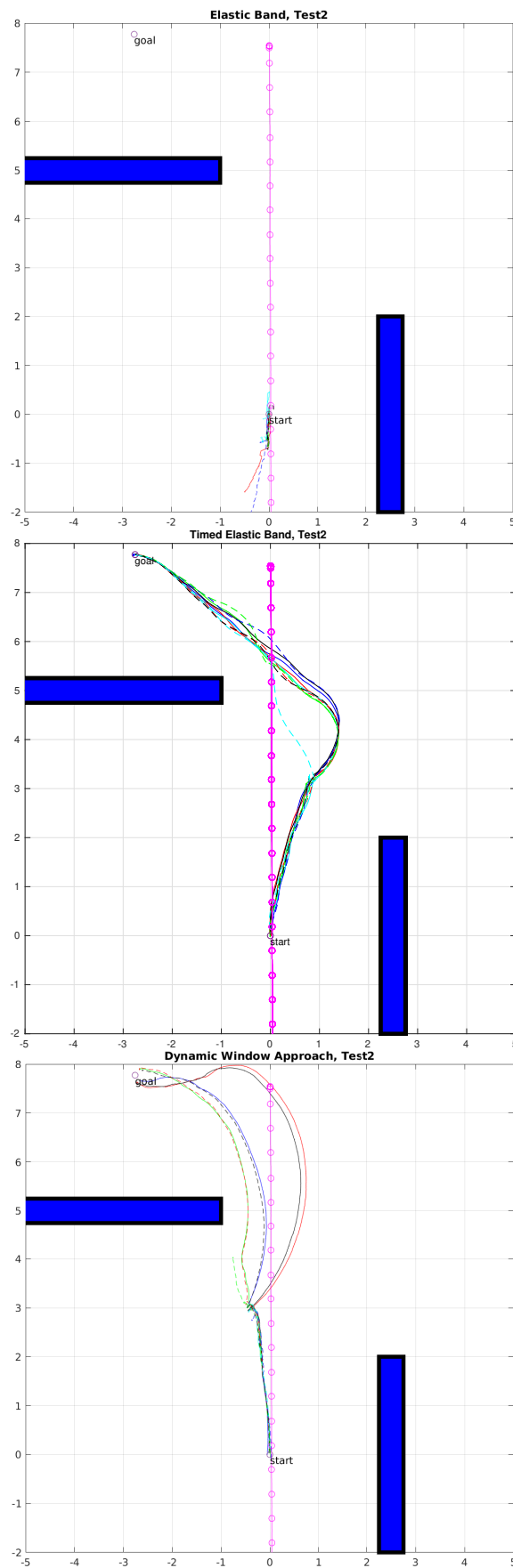


Figure 4.7: Test 2 plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

Test 3, plotted in Figure 4.8, had the obstacle moving right to left, perpendicular to the path of the robot. Compared to Test 1, the main difference in this scenario is the initial right-side wall that makes passing from the right impossible. From Table `reftab:Test3`, we can see the EB planner performed poorly, with a SR of 0.1. In most cases it failed to clear the measurements of the obstacle, as in Test 1, and followed a path parallel to the obstacle in an attempt to clear it. The TEB planner had wildly varying paths, occasionally managing to plan a path behind the obstacle, and occasionally attempting to pass it from the front. Unlike the EB planner, however, it managed to identify a plausible path behind the obstacle to eventually reach the goal. On one occasion it got lost and started to explore to the left and failed to reach the goal. The DWA managed to reach the goal each time.

Table 4.3: Evaluation scores for Test 3

Path-Planner	FR	CD	Time	SR
EB	0.0352	7.63m	79.4s	0.1
TEB	0.0302	8.67m	61.5s	0.9
DWA	0.0311	10.61m	52.4s	1.0

Test 4, shown in Figure 4.9, had the obstacle moving diagonally from top-right to bottom-left. From the evaluation scores in Table 4.4, we see the EB planner had a SR of 0.9, and both the TEB and DWA planners succeeded every run. The TEB planner resulted in more unique paths being found, including paths that required the robot to move backwards or turn in place. The DWA planner kept a relatively even heading in general.

Table 4.4: Evaluation scores for Test 4

Path-Planner	FR	CD	Time	SR
EB	0.0655	6.69m	75.6s	0.9
TEB	0.0333	7.14m	51.1s	1.0
DWA	0.0296	7.15m	47.2s	1.0

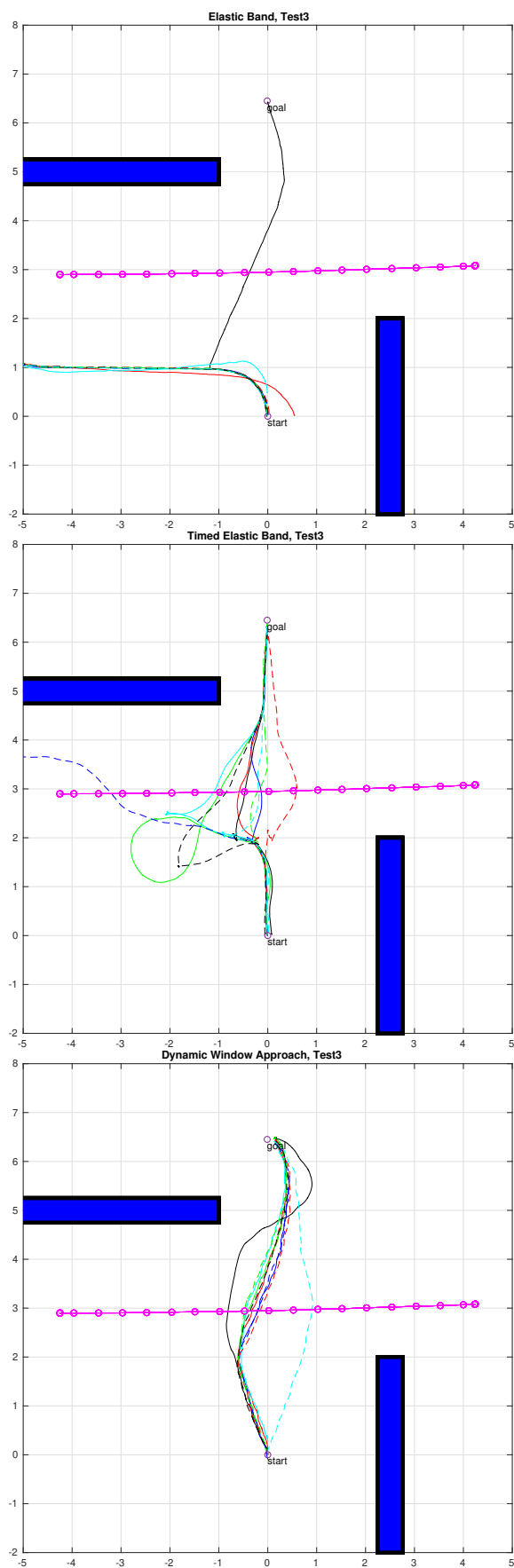


Figure 4.8: Test 3 plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

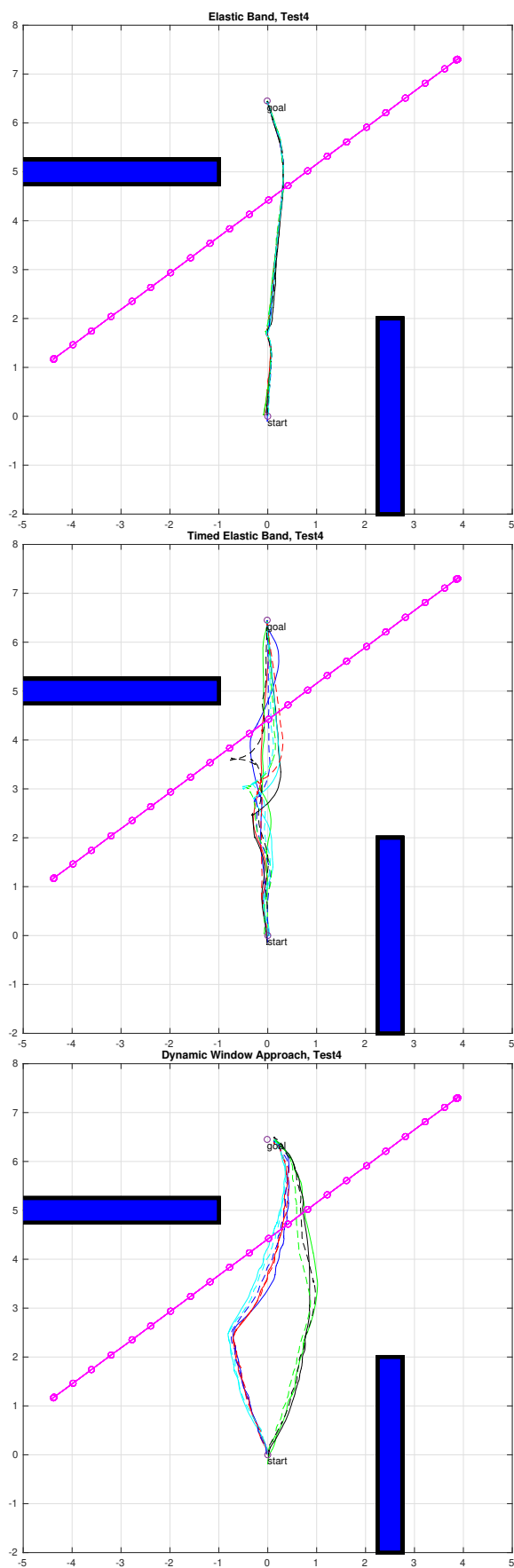


Figure 4.9: Test 4 plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

Test 5, plotted in Figure 4.10 had the obstacle move first along the global path, and then turn to the left, off the global path. All three path-planners succeeded in reaching the goal each run. Looking at the evaluation scores in Table 4.5, the EB planner showed almost no fluctuation in paths, with the TEB and DWA planners both showing some fluctuations. Notable for this run was that the EB and TEB planners performed roughly as fast, while the DWA took, on average, over 20 seconds longer to reach the goal. The EB planner showed a noticeably higher FR, while the TEB planner had the lowest FR.

Table 4.5: Evaluation scores for Test 5

Path-Planner	FR	CD	Time	SR
EB	0.0652	9.38m	47.2s	1.0
TEB	0.0186	9.65m	46.4s	1.0
DWA	0.0284	9.57m	69.6s	1.0

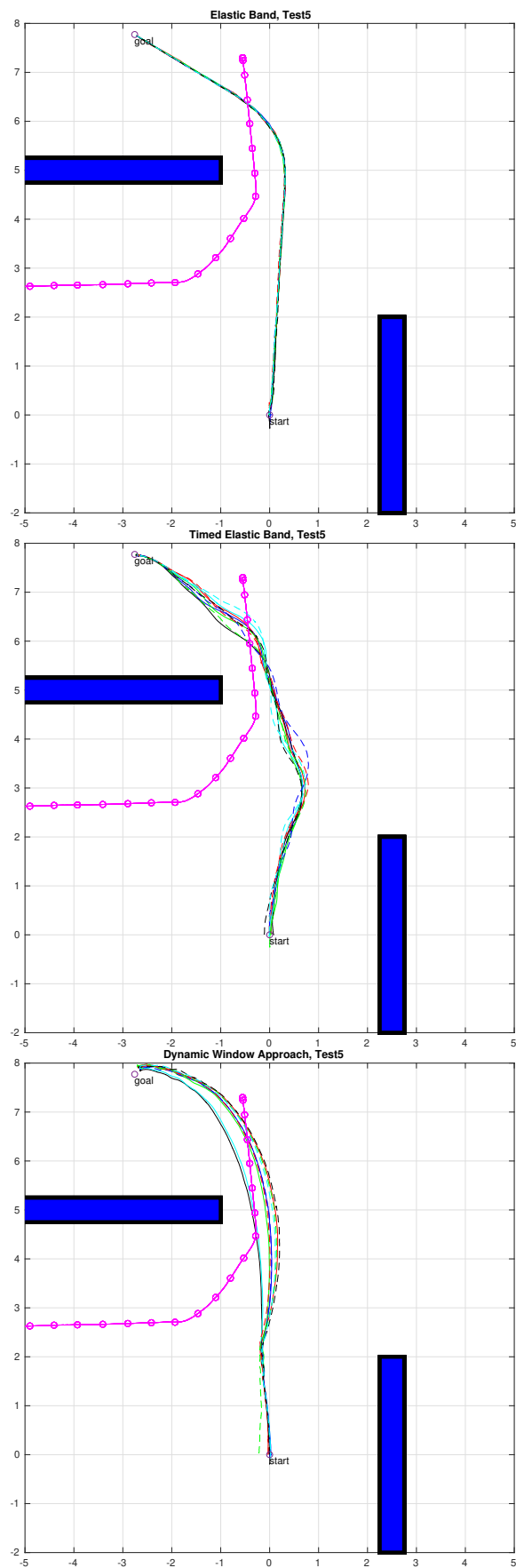


Figure 4.10: Test 5 plot for Elastic Band, Timed Elastic Band and Dynamic Window Approach.

4.4 Results

The test results shed some light on the strengths and vulnerabilities of each path-planner, and may give insight on the best use-case scenarios for each, if such exist. No one criterion or scenario can fully evaluate the usefulness of the path-planner. This section will discuss the ramifications of the results.

4.4.1 Divergent results

What can immediately be noticed is that all the path-planners' paths diverge from each other within a test on at least one occasion. As the scenarios are identical, path following is initiated at the same time instance each run, and the algorithms are initiated the same way each time, this implies that the systems depicted are not deterministic.

Of the path-planners, the EB path-planner finds the optimal solution to a given position each time. This can be seen in that most of the paths it finds are very close or identical to each other. The path-planner has no temporal or dynamic constraints, and as such will converge on the optimal path eventually, even if means staying still until the path is found. The DWA planner contains similar though slightly diverging paths. With the inclusion of a dynamic constraint, paths should be similar, but it does not necessarily converge on the exact same path each time. Of the three, the TEB planner had the most varied paths; this is in part due to the temporal constraint.

However, differences in algorithms do not completely explain the divergence. Even with the non-deterministic path-planners, divergence should be minimal between runs. The main reason for the differences is the measurement inaccuracies accrued during the run. Each portion of the test system is run in separate nodes, and there's no guarantee a measurement will arrive at the same time each run, nor that the measurement is handled in the same manner. Additionally, initial localization of the robot differs for each individual run. This can be seen particularly clearly in the DWA test run in Figure 4.10, where one initialization localized the robot to the left of the other runs.

These inaccuracies are acceptable, and in fact mirror interference in real applications. Calculation load, localization discrepancies and communication lag between nodes are issues that real robots must be able to overcome. In fact, it can be considered an important capability of a mobile robot's path-planner to be able to handle these kinds of errors without failure or degradation of operation.

4.5 Path-planner performance

In this section the individual path-planners and their performance will be discussed. Performance by both individual criteria as well as the over-all implications.

4.5.1 Elastic Band

The Elastic Band path-planner is the only planner in the selected path-planners that does not account for temporal constraints. It stands to reason that this should be visible in the results. Examining the plotted paths, the Elastic Band path-planner stands out in that all the plotted paths tend to be extremely similar. The baseline plot in Figure 4.5 shows an avoidance of stationary obstacles. Tests 1(Figure 4.6), 4(Figure 4.9) and 5(Figure 4.10) show that in cases where the Elastic Band is able to successfully navigate to the goal, there is little difference in paths. It implies a determinism to the path-planner, in that in identical scenarios it will react identically, which is in itself a positive attribute. However, in Test 2(4.7) the planner fails to react appropriately. In attempting to find the ideal path, it locks the robot into a state of constant reiteration as the moving obstacle continuously ruins the validity of the current planned path. Additionally, in Test 1(Figure 4.6) and Test 3(Figure 4.8) the robot is unable to identify a valid path to the goal through previously occupied space, presumably due to the lack of a "innovation" factor within the planner that would return to previously discarded paths.

The path-planner's performance in the FR metric sheds light on how the EB reacts mid-navigation. Despite the relative straightness of the paths in Figure 4.6, Figure 4.9, and Figure 4.10, the FR shows a remarkably higher values than either of the two other path-planners. In Table 4.1, the FR score is comparable to the FR score of the TEB planner, despite appearing to be a much straighter path. In Table 4.4 and Table 4.5, the FR score is clearly much higher. These results imply a large amount of very minor corrections in the path. As a candidate path-planner for a service robot, this may present a problem for comfortable operation.

The Elastic Band path-planner tends to find a shorter path than either of the two other planners, though the difference is marginal. In time spent in transit, the Elastic Band's performance varied widely. In Test 1, the Elastic Band performed close to equally well as the TEB planner. In Test 3 and 4, the time spent in transit was far longer than in either of the two other path-planners. In Test 5, the time spent was again comparable to the TEB

planner. The similarity to the TEB planner under ideal circumstances is unsurprising: the TEB method is based on the Elastic Band method, and it stands to reason that optimal paths are similar.

Overall the Elastic Band path-planner performed relatively poorly in a dynamic environment. Successes can be attributed to the robot being overwhelmed and unable to cope with the dynamic obstacle, resuming navigation only once the obstacle had passed, and failures were the result of the path-planner being driven into what the planner interpreted as a dead end by the dynamic obstacle.

4.5.2 Timed Elastic Band

The Timed Elastic Band path-planner is based on the Elastic Band path-planner, but it takes temporal constraints into consideration. That does not mean it is well-suited for dynamic situations. Time constraints should benefit replanning, in that it will find a viable path quickly. This is ideal for dynamic environments, since quick re-evaluation will mean less or no pauses in transit as a new path is found in a changed environment. It does not, however, mean the path-planner can predict or react pre-emptively to events that have not happened at the time of planning. Ideally, the dynamic cost-map implementation will help with giving advance knowledge of obstacle movement.

Optimality is often a trade-off for computational speed, and this is visible in the plots of the realized paths generated by the path-planner. While the baseline paths in Figure 4.5 show little variation, and notably a similarity to the plot of EB plot, all the scenario tests show some variation between paths. Non-optimal paths may lead to quickly computed but wild and non-intuitive paths. Ideally, the temporal and elastic constraints imposed by the TEB algorithm will cull the worst of these.

Examining the TEB path-planner's plots what stands out immediately is that there is clear deviation between different runs. In Figure 4.6, Figure 4.7 and Figure 4.10 the different plans largely follow the same general path, but show some deviation. Figure 4.8 and Figure 4.9 show wildly deviating paths and very different strategies for obstacle avoidance. This stands to reason, as in both cases the obstacle moves into the direction the local path-planner initially moves toward. As such, the temporal constraints do not allow the TEB planner to converge on an optimal path, resulting in variation over multiple tests. Additionally, where the EB path-planner appears to favor straight paths, and the DWA path-planner uniformly curving trajectories, the TEB path-planner tends towards small twists along and multiple changes of direction along the path.

The TEB planner's FR metric was largely comparable to the DWA planner's scores. In Tests 2, 3 and 4, the score was very similar. In Test 1, DWA clearly outperformed the TEB planner, and in Test 5 the TEB planner had a slightly better score. Overall, the TEB planner doesn't appear to wobble much and even in scenarios where the planner produces a looping or wandering path, the control signal doesn't oscillate. Despite the changes in direction at the world scale, the TEB path-planner's FR score does not imply constant minor corrections.

The TEB planner's Cumulative Distance does not imply any unreasonably long paths, but neither does it find the straightest path to the goal. In Tables 4.1, 4.4 and 4.5, the TEB planner's results were largely comparable to the other two. In Tables 4.2 and 4.3 the distance travelled was noticeably longer. The time spent in transit, however, did not fluctuate especially much except in Table 4.3, in which the robot tended to wander and loop around again, explaining the added time. Overall, it appears that the TEB planner did not spend much time in place, and was able find a path quickly.

In general, it appears that the TEB path-planner reacted quickly to dynamic obstacles, and did not lock up out of indecisiveness. For service use, this is very promising, as a hard stop tends to be less comfortable than a slightly longer path if it means there's no pause in operation. What may be problematic, however, is if the robot starts following loops while being operated, as happened in Figure 4.8. The TEB path-planner, in the end, had the best Success Ratio and most benefited from the dynamic costmapping.

4.5.3 Dynamic Window Approach

The DWA planner, in contrast to both other planners, accounts for the dynamics of the robot to find valid paths. As is, however, it does not account for the dynamics of an obstacle. The consideration of robot dynamics results in the consistent curves of the paths in the plotted paths. This can be seen even in Figure 4.5 and Figure 4.6, in which the overall path tended to be straight, slight curvature is introduced. This is caused by the fact that when considering different possible dynamic paths, there exists only one straight path, and far more paths with curvature. Additionally, the algorithm may be biased to curved paths when approaching goals. In Figures 4.7 and 4.10 the different paths appear to follow the same general trend of curving, with variation in the initial angle of the curve. In the rest of the tests, the paths are much closer to each other. In Figure 4.9, there are two main variations of path: one in which it initially attempts to circumvent the obstacle from in front, and the other in which it starts to pass behind the obstacle from the start.

The FR score of the DWA path-planner is comparable or noticeably lower than either of the two other path-planners, except in Table 4.5, in which the TEB path-planner out-performed the DWA planner. Compared to the EB path-planner, the DWA path-planner comes out clearly on top in all tests except Table 4.3. It appears that despite the gradual curve typical in DWA, there are generally fewer minor adjustments along the path. This bodes well for general comfortability of operation when deployed in service robots. However, the aforementioned gradual curves of the trajectory may seem confusing, or at least unintuitive for both operator and agents moving in proximity of the the robot.

The Cumulative Distance travelled by the robot is comparable or lower to the other two path-planners, with the exception of Table 4.2, in which the additional covered distance is caused by superfluous curves of the path. Examining the time spent in transit, the DWA path-planner was clearly the fastest in Test 3 (Table 4.3). It averaged a slightly faster time in Tables 4.1 and 4.4, while performing comparatively very poorly in 4.7 and 4.10. It appears that unlike the EB path-planner, the DWA path-planner is able to clear the costmap and find a path quickly once the obstacle has passed, however, it is not quite fast enough or otherwise capable of simultaneous planning and movement while a dynamic obstacle is in the plan horizon of the path-planner. This is supported by the plots clearly having a point near the the path of the object having a clear transition point from a straight path to a gradual curve, and the time in transit being low when the dynamic obstacle clears the collision zone of the robot quickly, and higher when it doesn't.

The DWA generally managed to reach the goal. With the exception of Figure 4.7, the DWA path-planner reached the goal every run. The failures to find a path are a result of a collision, as the path-planner freezes when a dynamic object is in its immediate proximity. Like the EB path-planner, it is unable to react to an obstacle moving on a direct collision course. Unlike the EB path-planner, however, it is initially able to plan and navigate, before the dynamic obstacle is within it's dynamic window, only seizing once the obstacle is close. This can be seen in the initial movement along the path, followed by a stop that leads to a divergence in paths.

Chapter 5

Discussion

In this chapter I will reflect on the results of the tests, what conclusions can be made and upcoming stages in development, as well as potential solutions to the challenges encountered. Dynamic environments pose challenges that have not been exhaustively explored, and a generic optimal approach is rarely available. Thus choosing a path-planner for a dynamic environment is heavily dependant on the context and use-case. In the use case of a service robot for the visually impaired, it can generally be assumed that dynamic obstacles, which in most cases would be pedestrians and cyclists, would know to avoid the robot and operator. The ability to keep operating while an obstacles approaches, however, is important, as is the ability to react accordingly and adjust when an obstacle gives way.

5.0.1 Evaluation criteria

5.0.1.1 Success Ratio

At face value the Success Ratio of a path-planner may appear the most important criterion to consider. However, the optimal approach to safe robotic navigation has long been considered to be a distributed layer-based approach[53][54], and in that regard our robot architecture is not different. In situations where a collision is imminent, the path-planning component's output will be overridden by a low-level collision avoidance system. Additionally, it can typically be assumed that a logistics or service robot operates in an environment in which potential hazardous dynamic obstacles such as walking people will take steps to avoid collisions themselves. Thus, a collision caused by the path-planner is not necessarily as fatal a mistake as it may appear. However, failure to reach the goal can also be caused by other reasons, such as inability to clear previously observed dynamic obstacles from

the cost-map or due to the path-planner simply being unable to identify a path that would be plausible for the robot to traverse.

5.0.1.2 Fluctuation Ratio

The Fluctuation Ratio gives the average change in heading between very close consecutive points along the recorded trajectory. Due to the closeness of the points, the ratio is small, and in some of the test cases is, counter-intuitively, to be smaller for the path-planner with an aggressively fluctuating plotted trajectory. This is because FR is sensitive to minor but constant corrections that, in practice, would cause a very small wobble in navigation. This is particularly important in the case of a guide-type service robot: a constant minor wobble and corrections, even slight ones, to the heading are noticeable to a user, and can result in worse over-all experience.

5.0.1.3 Time and Cumulative Distance

Both time to destination and Cumulative Distance are fairly straightforward metrics on their own. The shorter a safe path is, and the shorter the time to complete it, are both preferable to longer distances and longer times in transit for both logistics and service robots. Considered together, however, they can help identify stops and slow-downs in a robot's navigation. A service robot stopping or slowing down, while occasionally necessary to avoid hazards, can significantly degrade the experience of using the robot.

A long time spent in transit that doesn't result in a longer distance travelled means the robot has slowed down or stopped along the path. Stops in particular are indicative of a pause to re-evaluate the path plan. Re-evaluation is an important part of path-planning in dynamic environments, as unpredictable movements can render previously valid paths invalid. A path-planner must be able to find a path in real time. A realistic environment can be far more dynamically cluttered than the bare-bone scenarios presented within the tests, and failures to smoothly operate with single obstacles will multiply with more obstacles.

5.1 Comparison between path-planners

The results and analysis highlight certain aspects of the path-planners. The Elastic Band path-planner performed extremely poorly, not adapting to a dynamic obstacle at all. Path calculation times, downtime, heading-related "wobble" and ability to reset or clear previously unfeasible paths all were

lacking. While able to perform decently within a static but cluttered environment, the path-planner has very little merit in regards to dynamic environments. The test scenarios were simplified and focused on a single dynamic interaction; in a realistic environment such as a busy sidewalk in a town, or a shopping center, the interference of dynamic obstacles would be constant. There exists a very real risk that the Elastic Band path-planner would be completely unable to function in such an environment.

Additionally the Fluctuation Ratio of the path-planner's path raises questions of whether it is at all suitable for human operation. The path-planner seems to introduce constant wobble in virtually all scenarios, even though it typically tends to lock up until the obstacle has passed. The occasionally lacking ability to revisit paths that have been discarded as unfeasible makes the planner unreliable even after dynamic obstacles have passed. It appears in general that there are no redeeming qualities to the path-planner.

Comparatively the Timed Elastic-Band path planner fared well. The path-planner managed to reach the goal in almost all the runs. Perhaps most interestingly, the TEB path-planner was able to operate even during critical situations, where the dynamic obstacle was in the immediate vicinity of the robot and the projected cost bubble overlapped a previously free sector which the path-planner had already planned it's path in. A dynamic environment will almost always require re-evaluation as previously acceptable routes become unacceptable. The ability to do so in real-time while not ceasing operation otherwise is promising. A passage through a somewhat crowded sidewalk would be uncomfortable if it resulted in stopping for every obstacle, no matter how far or unlikely the collision is.

The TEB path-planner's FR did not differ to it's detriment from the other tested path-planners. However, the realized paths did contain some loops, backtracking and U-turns. In a real environment, with real pedestrians and other obstacles, it would not be acceptable for the robot operator to be led in looping or otherwise bizarre paths. At worst, it could confuse passers-by and create difficult and bothersome scenarios in which other pedestrians do not know how to react. However, in general, it can be assumed that a pedestrian passing by would take care to avoid a guide robot. The dynamic obstacles within the tests did not slow down or otherwise react, but an actual person would likely not continue straight on a collision path.

To further cement the TEB path-planner's success, it seems that it is best suited at finding a solution after running itself into a difficult position. It succeeded well at revisiting discarded path options and eventually found an acceptable path in almost all the test scenarios.

The Dynamic Window Approach path-planner fell in between the two others success-wise. It's FR scores were comparable to the TEB planner,

and it also was able to partially function in a dynamic environment. It did however suffer from some indecision, in some scenarios staying still as the dynamic object approached. It was however slower to reach the goal than the TEB path-planner, except in cases where the DWA path-planner froze, and a near-direct path presented itself to the planner.

The paths that were realized fell somewhat in between the two other path-planners as well. There exist variance between paths, but not to the extent of the TEB path-planner's. Paths tend to follow a similar overall trend. This is a positive characteristic for a guide robot, as once the operator becomes accustomed to the robot and its behavior those paths would feel natural and predictable. The DWA planner didn't exhibit a tendency for looping or abrupt changes in direction, instead opting for gradual and smooth curves towards the goal.

A particularly interesting characteristic of the DWA path-planner is the dynamic window behavior. The robot won't react strongly to obstacles outside its dynamic window. The result is that obstacles that are far away from the robot do not create problematic reactionary path changes before they become relevant to the robot, nor do they cause the robot to stop and re-evaluate the path too early. A realistic environment, with multiple moving obstacles both near and far, may pose a problem for the path-planner.

In conclusion, it can be said that of the tested path-planners, the Elastic Band path-planner is categorically unsuited for dynamic environments. It has little to no positive characteristics. The Dynamic Window Approach path-planner has some merit, and especially the behavior of the dynamic windows warrant consideration for future research. However, as standalone a path-planner the Timed Elastic Band path-planner is clearly the preferable choice for a dynamic environment.

5.2 Future implementation

It should be noted that a fully functional guide robot operating in a dynamic environment would incorporate many other software modules. Collision detection and avoidance should never be left for a path-planner but to a separate, specifically designed system that may override the default path-planner to avoid a collision. Safety, while of the utmost importance, does not have requirements as strict as with a road-worthy vehicle. Collisions would neither be as catastrophic as on an open road, nor would the speeds on a pedestrian passage way be high enough to create unavoidable collisions as easily. The rules regarding using a pedestrian thoroughfare are not as strict or set in stone as on an open road, potential damage caused by a collision is

less severe, and other users can be assumed to be able to react to a robot more intuitively and freely than on a road.

Nonetheless, the behaviour of the default path-planner is an important factor in the design of such a robot. A collision avoidance system should only affect the behaviour of the robot in cases where a collision is either imminent or extremely likely. In all other cases, it would be preferable for the path-planner to react to dynamic and static obstacles in such a way that high-risk situations would be avoided altogether. As this thesis has shown, there are naturally better performing path-planners. However, none of them truly accounted for dynamic movement along the temporal axis.

Future work in path-planning for dynamic environments would necessarily have to focus on finding a path in regard to the expected positions of obstacles. To solve this, it would be necessary to add a time value to each point along the path, and explore the potential paths with time in mind. A new dimension in space exploration would result in the calculation load growing by an order of magnitude. Additionally, the likelihood to recalculate the path would be increased, as unlike in traditional path-planning, in which the robot can always return to a planned path if it finds itself drifting off course, one can't return in time to a previous coordinate. Thus any drifting would have to be accounted for throughout the entire path and consequently in predictions for moving obstacles. Not only would it be calculation-heavy, it would introduce multiple points at which the path-planner may fail or radically change its approach.

The suggested solution to this is the implementation of time-horizons according to which a section of the path may be planned. By projecting the predicted locations of dynamic obstacles to a future point in time and planning the path traditionally in between horizons, it is possible to account for the dynamic nature of obstacles while avoiding calculation-heavy exhaustive exploration of the dynamic configuration space of the paths. Conceptually this borrows from the DWA path-planners dynamic windows, only instead of considering the dynamics of the robot, we consider a temporal window within the near future of the robot.

The reasoning for this is the ability to alter portions of the path in such a way that other, still plausible portions would remain unchanged. Recalculating only windows in which dynamic objects have travelled onto the path of the robot instead every part of the path will lessen the calculation load of the path planner, and avoid situations where the plan drastically changes from the one the robot has committed to. While this may result in less optimal paths, they would not create a feeling of drastic change in the behavior for the user, and ensure that obstacles can be reacted to in advance without compromising the current operations of the robot.

Chapter 6

Conclusions and Future Work

As an academic problem, path-planning has been explored over decades. It is, however, a relatively recent development that active path-planning has been implemented for commercial applications in safety-critical environments. The real-world applications for mobile robots typically function in dynamic and often non-controlled environments, populated by traffic or people acting in according to complex, and sometimes wholly random, behavior models. A perfect answer to this problem is still an open question; but this thesis aimed to both identify the best suited "out-of-the-box" path-planner for dynamic environments and to isolate the major problems a dynamic environment poses.

The tests were run in a simulated environment with a pseudo-predictive dynamics-derived costmap implementation to effectively create a very bare-bones predictive interface. The test scenarios were chosen to represent simple but realistic real-world scenarios in which a mobile robot might find itself. Test results were measured both by numerical metrics derived from literature as well as subjective visual analysis of the realized path plotted in relation to the obstacle path.

The results imply that out of the path-planners chosen, the Elastic Band path-planner is categorically unsuited for dynamic environments, especially in scenarios requiring reactive actions to avoid collision in lieu of passively waiting for the obstacle to pass, due to the constant restarting of the calculation process. The Dynamic Window Approach path-planner was better able to navigate a dynamic environment, but showed some difficulty in reacting to a threat of collision. The Timed Elastic Band method was clearly the superior path-planner, able to react and plan in a timely manner even in a shifting environment. This doesn't mean it was especially well-suited for dynamic environments as-is, however. It was able to react to dynamic obstacles, but the behavior during the reactive action was not always optimal.

Analysis of the results showed a need for predictive analysis in a dynamic environment. Measurements of obstacle position and velocity alone are not enough to navigate an environment. Not only is the path of the obstacle necessary to estimate into the future, the position of the robot itself needs to be estimated in conjunction to ensure that the robot won't react to an obstacle that won't be near the trajectory by the time the robot gets closer to its former position. A suggestion for this was found in temporal windows.

For the purpose of future implementations of path-planner related applications within the Roboguide service robot, this thesis has outlined the best path-planner of the selected planners as well as identified requirements for future work on the robot.

Bibliography

- [1] Zvi Shiller, Yu-Rwei Gwo, et al. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, 1991.
- [2] Kamran H Sedighi, Kaveh Ashenayi, Theodore W Manikas, Roger L Wainwright, and Heng-Ming Tai. Autonomous local path planning for a mobile robot using a genetic algorithm. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, volume 2, pages 1338–1345. IEEE, 2004.
- [3] Autonomous Warehouse Robots: a Brief History. <https://www.inviarobotics.com/blog/autonomous-warehouse-robots-brief-history/>. Accessed: 2020-08-05.
- [4] New robot can follow pedestrian traffic rules. <https://www.deccanchronicle.com/technology/in-other-news/310817/new-robot-can-follow-pedestrian-traffic-rules.html>. Accessed: 2020-08-05.
- [5] BK Patle, Anish Pandey, DRK Parhi, A Jagadeesh, et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606, 2019.
- [6] E Bevilacqua and T Kimura. Obstacle movement prediction considering obstacle’s dynamics and a priori knowledge of its goals. In *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT’02.*, volume 2, pages 724–729. IEEE, 2002.
- [7] Dave Ferguson, Michael Darms, Chris Urmson, and Sascha Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1149–1154. IEEE, 2008.

- [8] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.
- [9] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.
- [10] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.
- [11] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.
- [12] Paul Newman and Kin Ho. Slam-loop closing with visually salient features. In *proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 635–642. IEEE, 2005.
- [13] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.
- [14] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.
- [15] Jérôme Barraquand and J-C Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1712–1717. IEEE, 1990.
- [16] Léonard Jaillet, Juan Cortés, and Thierry Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.
- [17] Liz Murphy, Steven Martin, and Peter Corke. Creating and using probabilistic costmaps from vehicle experience. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4689–4694. IEEE, 2012.

- [18] Juil Sock, Jun Kim, Jihong Min, and Kiho Kwak. Probabilistic traversability map generation using 3d-lidar and camera. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5631–5637. IEEE, 2016.
- [19] Sanjiv Singh, Reid Simmons, Trey Smith, Anthony Stentz, Vandi Verma, Alex Yahja, and Kurt Schwehr. Recent progress in local and global traversability for planetary rovers. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1194–1200. IEEE, 2000.
- [20] Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.
- [21] Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. In *Sensor devices and systems for robotics*, pages 253–276. Springer, 1989.
- [22] David V Lu, Dave Hershberger, and William D Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–715. IEEE, 2014.
- [23] Purushothaman Raja and Sivagurunathan Pugazhenthii. Optimal path planning of mobile robots: A review. *International journal of physical sciences*, 7(9):1314–1320, 2012.
- [24] Yong K Hwang and Narendra Ahuja. Gross motion planning—a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.
- [25] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [26] Winston L Nelson and Ingemar J Cox. Local path control for an autonomous vehicle. In *Autonomous robot vehicles*, pages 38–44. Springer, 1990.
- [27] AT Ismail, Alaa Sheta, and Mohammed Al-Weshah. A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4):341–344, 2008.
- [28] Adem Tuncer and Mehmet Yildirim. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6):1564–1572, 2012.

- [29] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
- [30] Dieter Fox, Wolfram Burgard, Sebastian Thrun, and Armin B Cremers. A hybrid collision avoidance method for mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 2, pages 1238–1243. IEEE, 1998.
- [31] Marija Seder and Ivan Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1986–1991. IEEE, 2007.
- [32] Yi-Chun Lin, Chih-Chung Chou, and Feng-Li Lian. Indoor robot navigation based on dwa*: Velocity space approach with region analysis. In *2009 ICCAS-SICE*, pages 700–705. IEEE, 2009.
- [33] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.
- [34] Jennifer King and Maxim Likhachev. Efficient cost computation in cost map planning for non-circular robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3924–3930. IEEE, 2009.
- [35] Roland Philippsen and Roland Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *Proceedings. 2003 IEEE International Conference on Robotics and Automation, September 14-19, 2003, The Grand Hotel, Taipei, Taiwan*, volume 1, pages 446–451. IEEE Operations Center, 2003.
- [36] Ros eband local planner. http://wiki.ros.org/eband_local_planner. Accessed: 2019-11-06.
- [37] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6. VDE, 2012.
- [38] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse

- model. In *2013 European Conference on Mobile Robots*, pages 138–143. IEEE, 2013.
- [39] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. Global and local path planning study in a ros-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, 2018, 2018.
- [40] Christoph Rösmann, Malte Oeljeklaus, Frank Hoffmann, and Torsten Bertram. Online trajectory prediction and planning for social robot navigation. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1255–1260. IEEE, 2017.
- [41] Amir R Soltani, Hissam Tawfik, John Yannis Goulermas, and Terrence Fernando. Path planning in construction sites: performance evaluation of the dijkstra, a, and ga search algorithms. *Advanced engineering informatics*, 16(4):291–303, 2002.
- [42] Zhixin Chen, Mengxiang Lin, Shangzhe Li, and Rui Liu. Evaluation on path planning with a view towards application. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 27–30. IEEE, 2017.
- [43] S-H Suh and Kang G Shin. A variational dynamic programming approach to robot-path planning with a distance-safety criterion. *IEEE Journal on Robotics and Automation*, 4(3):334–349, 1988.
- [44] Guy Hoffman and Wendy Ju. Designing robots with movement in mind. *Journal of Human-Robot Interaction*, 3(1):91–122, 2014.
- [45] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.
- [46] Matías Nitsche, Taihú Pire, Tomáš Krajník, Miroslav Kulich, and Marta Mejail. Monte carlo localization for teach-and-repeat feature-based navigation. In *Conference Towards Autonomous Robotic Systems*, pages 13–24. Springer, 2014.
- [47] Hui-Zhong Zhuang, Shu-xin Du, and Tie-jun Wu. On-line real-time path planning of mobile robots in dynamic uncertain environment. *Journal of Zhejiang University-SCIENCE A*, 7(4):516–524, 2006.

- [48] Jur Van Den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2366–2371. IEEE, 2006.
- [49] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009.
- [50] Rachel Kirby. *Social robot navigation*. PhD thesis, figshare, 2010.
- [51] Christoph Sprunk, Gian Diego Tipaldi, Andrea Cherubini, and Wolfram Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3144–3149. IEEE, 2013.
- [52] Robotic Operating System. <https://www.ros.org>. Accessed 2020-05-23.
- [53] Erann Gat, R Peter Bonnasso, Robin Murphy, et al. On three-layer architectures. *Artificial intelligence and mobile robots*, 195:210, 1998.
- [54] Anders Orebäck and Henrik I Christensen. Evaluation of architectures for mobile robotics. *Autonomous robots*, 14(1):33–49, 2003.