UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Control of Mobile Robots Moving in Cluttered Environments

Von der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.))

genehmigte Dissertation

von

MSc-Eng. Muhannad Abdallah Shaker Mujahed

Erster Gutachter: Prof. Dr.-Ing. Bärbel Mertsching
Zweiter Gutachter: Prof. Dr.-Ing. Dietrich Paulus

Tag der mündlichen Prüfung: 30.10.2020

Paderborn 2020

Diss. EIM-E/352

Dedication

This thesis is dedicated to my beloved parents, who have raised me to be the person I am today;

Wife and kids, who have been with me every step of the way through good times and bad;

Wonderful grandmother, for her love and support;

Sister and brothers, for their encouragement;

Thank you all, I love you!

Declaration

I hereby declare that I have completed the work on this PhD dissertation with my own efforts and no part of this work or documentation has been copied from any other source. It is also assured that this work is not submitted to any other institution for award of any degree or certificate.

Paderborn, November 9, 2020

Muhannad Mujahed

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Zusammenfassung der Dissertation

Control of Mobile Robots Moving in Cluttered Environments
Des Herrn Muhannad Mujahed

Während der letzten Jahrzehnte hat die Entwicklung mobiler Roboter viel Aufmerksamkeit erfahren, besonders bei den Entwicklungen in den Anwendungsbereichen Exploration, Suchen, Bergen und Haushalt. Der Bau solcher Roboter erfordert die Bewältigung verschiedenster Anforderungen wie Wahrnehmen, Verfolgen und Kartieren. Ungeachtet der zu erfüllenden Aufgabe oder des Anwendungsgebiets muss ein Roboter dennoch in der Lage sein, seine eigene Bewegung zu planen. Die Bewegungssteuerung ist daher Kern der Robotertechnik und wurde seit der Entwicklung des ersten mobilen Roboters eingehend betrachtet. In der Regel müssen reale Umgebungen als unbekannt und zeitlich veränderlich angekommen werden. *Herkömmliche Bewegungsplanungsverfahren*, die auf vorgefertigte Karten angewiesen sind, funktionieren daher in solchen Umgebungen nicht mehr zuverlässig. *Reaktive Kollisionsvermeidungsverfahren* gehen dieses Problem an, in dem sie sensorische Wahrnehmungen in das Regelungssystem einbeziehen und damit die Lücke zwischen Planen eines Pfades und Ausführen der Bewegung schließen. Der Großteil dieser Methoden unterliegt jedoch klassischen Problemen, die ihre Leistungsfähigkeit in *unübersichtlichen* Umgebungen begrenzt; dazugehören Anfälligkeit für Oszillationen, Fehlfunktion beim Steuern des Roboters durch enge Passagen, Nichtberücksichtigung der Einschränkungen des Roboters, sowie die Tendenz längere Pfade und höhere Ausführungszeiten zu generieren.

Die vorgestellte Arbeit ziel darauf ab, die oben genannten Probleme anzugehen. Dazu wurde ein neuartiger Ansatz zur Kollisionsvermeidung entwickelt. Die Schlüsselidee ist eine Analyse der Umgebungsstruktur, um die vielversprechendste Lücke zu finden und ein Subziel in einen kollisionsfreien Bereich zu legen, so dass der Öffnungswinkel der Lücke berücksichtigt wird und ein sicherer, glatterer Übergang zwischen Kollisionsvermeidung und dem Erreichen des Ziels vorhanden ist. Dies hat auch kürzere Pfade und kleinere Ausführungszeiten zur Folge. Dieser vorgeschlagene Ansatz ist durch Berücksichtigung des Abstandes zu Hindernissen verbessert worden, in dem alle umgebenen Hindernisse in die Berechnung des Lenkwinkels einfließen. Dies wurde möglich durch die Einführung und Integration von zwei Konzepten, nämlich „tangential navigation“ und „gap flow navigation“. Ein weiterer Beitrag liegt in der Berechnung von Steuerbefehlen derart, dass das System garantiert Ljapunow-stabil ist. Des Weiteren wird ein neues Konzept, genannt „admissible gap“, vorgestellt, das sich mit der Frage befasst, ob eine gegebene Lücke durchfahrbar ist, in dem eine zulässige, kollisionsfreie Bewegungsregelung ausgeführt wird. Dieses Konzept ist erfolgreich eingesetzt worden, um ein Kollisionsvermeidungsverfahren zu entwickeln, das direkt die Fahrzeugeinschränkungen berücksichtigt, ohne eine holonome Lösung anzupassen. Ein weiterer Beitrag ist die Entwicklung einer neuen Strategie zur Lückensuche, die Oszillationsmöglichkeiten reduziert und die Stabilität der Navigation verbessert. Zum Abschluss werden experimentelle Ergebnisse zusammen mit einer Leistungsbewertung für hochkomplexe Szenarien vorgestellt, um zu verifizieren, dass das vorgestellte Verfahren andere aktuell übliche Techniken in Bezug auf Gleichmäßigkeit, Effizienz, Zuverlässigkeit und Sicherheit übertrifft.

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Abstract

Control of Mobile Robots Moving in Cluttered Environments
Mr. Muhannad Mujahed

Over the past few decades, mobile robots have gained a lot of attention, particularly with the evolution of application fields such as search and rescue, cleaning, and exploration. Developing such robots requires to cope with different challenges such as perception, tracking, and mapping. Nevertheless, regardless of the mission to be performed or the application domain, robots must be able to plan their own motion. Hence, motion planning is at the heart of robotics and has been thoroughly addressed since the first mobile robot was developed. Usually, real-world environments are unknown and change over time. Therefore, *traditional path planning* methods that build upon a previously known map fail to work properly in these environments. *Reactive collision avoidance* approaches tackle this problem by incorporating the perceived information into the control system, bridging the gap between planning a path and executing a motion. Unfortunately, the majority of these methods undergo some classical drawbacks limiting their performance in *cluttered* environments. These include being prone to oscillations, failure of guiding a robot through narrow spaces, neglect of the robot constraints, and the tendency to generate longer paths and higher execution times.

The work presented in this thesis aims to cope with the above mentioned problems. To this end, a novel collision avoidance approach was developed and implemented. The key idea is to analyze the environmental structure and find out the most promising gap, once determined, a subgoal is located in a collision-free area. It is located in such a way that the opening angle of the selected gap is considered, providing a safer and smoother bridge between collision avoidance and target approach. This also leads to shorter paths and less execution times. The proposed approach has been improved by considering the clearance to obstacles and by computing the steering angle in such a way that all surrounding obstacles are taken into account. This has been possible by introducing and integrating two concepts, called “tangential” and “gap flow” navigation. Another contribution is the computation of the motion command in such a way that the stability of the system is guaranteed in the Lyapunov sense. Furthermore, this work presents a new concept, the “admissible gap”, which addresses the question of whether a given gap is traversable by performing an admissible collision-free motion control. This concept has been successfully employed to develop a collision avoidance approach, that directly respects the vehicle constraints rather than adapting a holonomic-based solution. Another contribution is the development of a new strategy for extracting gaps, which reduces the possibility of oscillation and improves the stability of navigation. Finally, experimental results along with performance assessment in highly cluttered scenarios are presented to verify that the proposed approaches outperform state-of-the-art techniques in terms of smoothness, efficiency, reliability, and safety.

Acknowledgements

First of all, praise and thanks be to Allah who enabled me to reach this level of academic achievement. Secondly, I would like to express my greatest gratitude to the following people for helping me to turn a dream into reality, be it through scientific and technical discussions, through moral support during periods of stress and doubt, or by providing the possibility of participating in the adventures of mobile robotics. Without their guidance, help, and love, the work presented in this thesis would not have been possible.

The first person that comes to mind is my principal adviser Prof. Dr.-Ing. Bärbel Mertsching, who believed in me from the very beginning and allowed me to develop professionally as well as individually. With here patience, guidance, deep vision, support, constant encouragement, now I am at the end of this long way. I would also like to thank here for giving me the opportunity to visit GET Lab in 2009 as an internship student. At that time, I was assigned to a team preparing the participation of GET Lab in the SICK Robot Day 2009. Since then, my interest in mobile robotics has grown and it continues to evolve.

Obviously, the research work presented in thesis could not have been possible without the great team in our lab. Thank you all for the nice time we have had together. In particular, I owe my gratitude to Dirk Fischer, whose knowledge about all kinds of mechanical and electrical equipments has been reflected by amazing hours discussing the different possibilities of configuring our robot. Dirk was my supervisor at the time of my visit to GET Lab in 2009. His guidance at that time was extremely helpful. Although I have had little experience in the field of robotics, he helped me to step ahead and overcome all difficulties. I also present my sincere thanks to Mahmoud A. Mohamed, who was always there. I am also grateful to Markus Hennig, Musa Kazmi, Jan Tünnermann,

and Daniel Nickchen for their support, suggestions, and positive discussion. I also have good memories from colleagues that were in GETLab in the past, in particular from Irtiza Ali, Mohamed Shafik, Hossein Mirabdollah, Zaheer Aziz, Tobias Kotthäuser, Shakeel Ahmad, and Zubair Kamran.

I would also like to thank my friends for making my stay in Germany a wonderful time. In particular, many thanks goes to Hamzah, Yahia, Husam, Muhannad, Majde, Abdelruham, Adnan, and Asad who have made life easier and interesting at the same time. Thanks to my friends from Palestine; Hammam, Muhannad, Anan, Safwat, and Ala; they simply never forget me and I never forget them. My sincere thanks goes to Prof. Hussein Jaddu and Prof. Karim Tahboub, the advisor and external examiner of my master's thesis. They have always supported me and provided the best research advice with a high human quality.

This research has been sponsored by the German Academic Exchange Service (DAAD). The funding is gratefully acknowledged. I am very fortunate to have had the opportunity to participate in several conferences and events worldwide, especially the RoboCup German Open 2012 - 2015 competitions and the World Robocup 2016. It is worth to remember all researchers whom I came across at those events and conferences; I would like to thank you all for making each trip an enriching experience.

Above all, I wish to thank my parents, wife, kids, sister, and brothers, who have always remained a reliable anchor providing love, support, and understanding. I am forever indebted to my mom; the older I get the more I realize and appreciate what you have given me. You always pray for me and wish all the best and success throughout my life. Without your love and emotions I would not be in such a position. Thanks mom. My dad spent his life helping and encouraging me for education and I will always be grateful to him. I am endlessly grateful to my wife Fida, my daughter Aseel, and my sons Obadah and Adam, who have suffered my unlimited working hours at days and nights. Fida has always encouraged me when I was feeling unsure of my next steps. Obadah and Adam have been a constant source of joy and wonder. Their smile has always brighten my day, giving me renewed energy to push on. Aseel missed me a lot during the last couple of years and she always used to ask: why don't you stay with us, dad?

Contents

1 Introduction **1**
- 1.1 Motivation and Goals . . . 2
- 1.2 Contributions of this Work . . . 4
- 1.3 Thesis Outline . . . 7

2 Autonomous Mobile Robot Navigation **9**
- 2.1 Introduction . . . 9
- 2.2 Basic Concepts . . . 11
  - 2.2.1 Configuration Space . . . 11
  - 2.2.2 Configuration Space Obstacle . . . 12
  - 2.2.3 Notion of a Path and a Trajectory . . . 13
  - 2.2.4 Non-holonomic Mobile Robots . . . 14
  - 2.2.5 Lyapunov Stability Theory . . . 17
  - 2.2.6 Sensor Data . . . 19
- 2.3 Path Planning Techniques . . . 21
  - 2.3.1 Roadmap Path Planning . . . 21
  - 2.3.2 Graph and Grid based Methods . . . 22
  - 2.3.3 Safe Interval Path Planning . . . 23
  - 2.3.4 Probabalistic Roadmap . . . 24
  - 2.3.5 Rapidly-exploring Random Tree . . . 25
- 2.4 Reactive Navigation Techniques . . . 26
  - 2.4.1 Bug Algorithms . . . 27
  - 2.4.2 Artificial Potential Fields . . . 29
  - 2.4.3 Virtual Force Field . . . 31
  - 2.4.4 Vector Field Histogram . . . 32
  - 2.4.5 Dynamic Window Approach . . . 33
  - 2.4.6 Velocity Obstacles . . . 36
  - 2.4.7 Nearness-Diagram Navigation . . . 38

3 A New Gap-based Collision Avoidance Approach - A Holonomic Solution **41**
- 3.1 The Reactive Navigation Strategy . . . 42
  - 3.1.1 Preliminary Definitions and Notations . . . 43
  - 3.1.2 Selecting the Direction of Motion . . . 45
  - 3.1.3 Extracting Gaps . . . 47

3.1.4 Locating the Subgoal . . . 49
3.1.5 Determining Motion Commands . . . 54
3.2 Simulation Results . . . 56
3.2.1 Scenario 1 Simulations . . . 57
3.2.2 Scenario 2 Simulations . . . 58
3.3 Experimental Results . . . 61
3.4 Conclusions . . . 62

4 Smooth Navigation in Unstructured Narrow Spaces - A Holonomic Solution 63
4.1 Motion Situations and Corresponding Actions . . . 64
4.1.1 Gap Rotation Angle . . . 66
4.1.2 Tangential Rotation Angle . . . 67
4.1.3 Gap Flow Rotation Angle . . . 69
4.1.4 Tangential Gap Flow Rotation Angle . . . 74
4.2 Determining Motion Commands . . . 80
4.2.1 Limiting Speed . . . 82
4.2.2 Stability Analysis . . . 83
4.3 Experimental Results . . . 84
4.3.1 Experimental Results for ND+ and CG . . . 84
4.3.2 Experimental Results for SG and TGF . . . 85
4.4 Conclusions . . . 87

5 Evaluation of the Holonomic Solutions 89
5.1 Experimental Setup . . . 90
5.2 Experiments . . . 90
5.2.1 Experiment 1 . . . 91
5.2.2 Experiment 2 . . . 91
5.2.3 Experiment 3 . . . 92
5.2.4 Experiment 4 . . . 94
5.2.5 Experiment 5 . . . 96
5.2.6 Experiment 6 . . . 96
5.2.7 Experiment 7 . . . 98
5.3 Performance Measures . . . 99
5.3.1 Efficiency Metrics . . . 101
5.3.2 Oscillation Metrics . . . 102
5.3.3 Smoothness Metrics . . . 103
5.3.4 Physics-based Metrics . . . 104
5.3.5 Security Metrics . . . 105
5.4 Evaluation and Discussion . . . 105

6 Under-constrained Reactive Collision Avoidance Navigation 113

6.1 Preliminary Definitions . . . 115
6.2 Detecting Gaps . . . 116
6.2.1 Spatial Discontinuities . . . 116
6.2.2 Gaps Search . . . 117
6.2.3 Gaps Reduction . . . 121
6.3 Admissible Gap . . . 123
6.3.1 Kinematic Constraints . . . 123
6.3.2 Traversing Gaps . . . 126
6.3.3 Checking Admissibility . . . 129
6.4 AG Obstacle Avoidance Method . . . 132
6.4.1 Goal Navigability Check . . . 132
6.4.2 Gap Navigability Check . . . 134
6.4.3 Setting Motion Commands . . . 138
6.5 Experimental Results . . . 139
6.5.1 Experiment 1 . . . 140
6.5.2 Experiment 2 . . . 140
6.5.3 Experiment 3 . . . 141
6.5.4 Experiment 4 . . . 143
6.5.5 Experiment 5 . . . 145
6.5.6 Experiment 6 . . . 146
6.5.7 Experiment 7 . . . 148
6.6 Evaluation and Discussion . . . 150
6.7 Conclusions . . . 154

7 Conclusions and Future Work **155**
7.1 Conclusions . . . 155
7.2 Future Work . . . 158

Bibliography **159**

List of Notations **183**

List of Abbreviations **189**

List of Tables **191**

List of Figures **203**

List of Publications **203**

1 Introduction

We humans have long been fascinated by our capabilities to sense the environment, interact with the physical world, make decisions, and react to what is happening around us. This fascination has been expressed by directly trying to create smart machines being able to mimic the human behavior. In 1921, Karel Capek, a Czech novelist, wrote his satirical drama RUR-Rossum's Universal Robots, in which the term *robot* first appeared describing artificially-created workers smart enough to replace a human in any job. Since then, the term robot has caught on among both the scientific community and the general public, and is often used to describe any intelligent machine performing the work of humans. Nowadays, robots are becoming increasingly involved in every aspect of the modern life stepping a head towards mimicking nature and creating human-like helpers, turning the dream of Capek into reality.

Autonomous mobile robots have proven to be a powerful and very effective tool in many real-world applications [MFM18]. Perhaps the most intriguing feature of an autonomous robot is its ability to accomplish tasks under conditions where human presence is hard, unsafe, or impossible [SRD+17]; most humans would not be able to perform hazardous waste cleanup, examine an active volcano, explore the surface of planets over a long period of time, or walk through high-risk areas hit by natural disasters searching for survivors and/or rescuing them.

Designing mobile robots capable of autonomously performing tasks requires to cope with many problems; object detection, perception, control, decision making, just to name a few. Nevertheless, regardless of the mission to be executed or the application domain, at some point, the robot needs to navigate. Hence, motion planning is the heart of any robotic system. It reflects the robot's ability to generate and execute a motion so that a prescribed target is reached as efficiently

(optimal solution) and as safely (avoid collisions) as possible. This topic is, broadly speaking, the subject of the research work presented in this thesis.

1.1 Motivation and Goals

Driven by the dream of creating systems that would accomplish tasks under conditions where human existence is impossible or unsafe, robotics has gained an increasing attention in the last few decades. This research domain has been motivated by different real-world challenges, such as mining, cleaning, search and rescue, and military [MFM15]. Usually, real-world environments are unknown, unstructured, and change with time. Furthermore, unforeseen objects may obstruct the precomputed trajectories when executing tasks. This raises a wide range of challenges for introducing a robust motion planning approach [MFM16].

Traditional motion planning methods [Lat91] [LaV06] rely on accurate and static models of the environment, dealing with the navigation problem on a larger scale in which a predefined map is used to generate an efficient free path. This path is computed off-line with previously known obstacles. Therefore, these techniques often fail to function properly in dynamically changing environments.

To overcome this limitation, it is crucial to integrate the data acquired by sensors into the control system, bridging the gap between trajectory computation and motion execution. This helps in detecting dynamic changes that may occur during navigation, thus reacting to unexpected obstacles. These challenges are addressed by local reactive navigation (collision avoidance) techniques, where only a small portion of the environmental model is employed, thus achieving fast obstacle avoidance with low computational complexity [MFM15].

The majority of reactive navigation approaches present limited performance in dense, complex, and cluttered environments (as the one shown in figure 1.1). This is owing to the fact that these techniques are prone to some classical drawbacks [MM04] [MLL16] [MFM16] such as computational complexity, experiencing a local minima, difficulties of driving a robot towards obstacles (when necessary), failure of navigating a robot through narrow spaces, and the tedious parameter

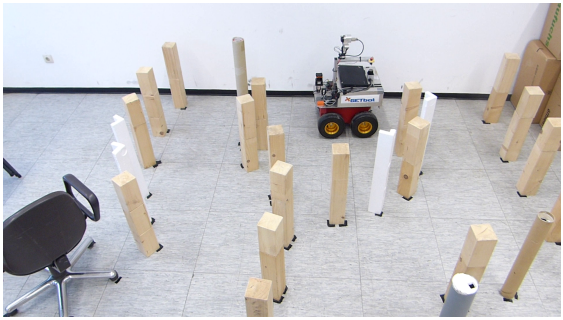


Figure 1.1: Our mobile robot, GETbot, moving through a cluttered environment.

tuning. It is still an open research problem to find an efficient technique that enables robots to safely move in such environments.

Over the past years, researchers have distinguished between approaches that generate a direction of the robot to head for and approaches that perform a search in the velocity space, selecting a steering command rather than a motion direction. The former, referred to as "directional approaches", solve the motion problem in two stages; first, a direction solution is found by analyzing the sensory information. Second, motion commands are computed in such a away that the robot navigates towards the determined direction [MFMJ10]. The latter, referred to as "velocity space approaches", map the constraints stemming from physical restrictions and obstacle information into the velocity space, and then select the speed that fulfills these constraints and optimizes an objective function [MFM15].

Velocity space approaches may allow to drive a robot at higher speeds and show a smoother behavior. Moreover, they consider the dynamic constraints of the vehicle. The Curvature Velocity Method [Sim96] [SWY10] and the Dynamic Window Approach [FBT97] [LV16] are typical representatives of this category. Unfortunately, these methods may fail to drive a robot through tight passages in dense environments. Furthermore, they are prone to local minima.

The directional approaches such as the Artificial Potential Field [Kha86] [VT12] [KST+16] and the Virtual Force Field [BK89] [NTK+11] are simple to implement and consume less computational load than velocity space methods. More-

over, experiments demonstrated that navigation in dense and cluttered environments has been successfully achieved using the Nearness-Diagram (ND) Navigation [MM04] [MFMJ10], that belongs to this category of approaches. ND-based methods avoid the limitations of most obstacle avoidance techniques mentioned above. However, they are prone to oscillations and instability, which in turn may cause a significant reduction in speed and can be unsafe in narrow spaces [MFM16]. Furthermore, ND-based methods are likely to cause deviations towards free regions, increasing the total time and distance needed to perform an assigned task. Hence, the **first goal** of this work is to develop a new collision avoidance approach thats avoids the aforementioned limitations of the ND-based techniques. In other words, our first goal is to propose a collision avoidance approach that safely guides a mobile robot through cluttered and dense environments, while enhancing the efficiency (execution time and path length), smoothness, and stability of the trajectories generated by the ND-based methods.

An additional yet significant limitation of the ND-based methods is the assumption of a holonomic disc shaped robot, which may not be valid in real-world scenarios. Neglecting the actual robot shape and its kinematics may hinder finding feasible motions or lead to collisions [MM17] [MFM18]. Considering these constraints is especially critical for robots operating in highly cluttered environments. Addressing this issue is the **second goal** of this thesis.

1.2 Contributions of this Work

A new reactive obstacle avoidance approach for autonomous mobile robots was developed and implemented. The proposed approach, called “Safe Gap” (SG) navigation, computes the motion commands based on the current sensor data rather than using a predefined/generated map. By employing the SG approach, it has been possible to steer a mobile robot in unknown, dynamic, and dense environments. The SG method follows the “closest gap” concept, that we have proposed in [MFMJ10], as a methodology to analyze the environmental structure and find out the most promising gap (opening) for navigation. Collision avoidance is carried out in such a way that the location of obstacle points between the current robot configuration and the chosen gap is taken into account, creating a

subgoal in a collision-free area. This subgoal is determined based on the position of the goal and the opening angle of the gap. By this means, a smoother and safer bridge between collision avoidance and goal approach is achieved. Moreover, unreasonable deviations towards free regions are avoided, reducing the total time and distance needed to complete the mission [MFM13a] [MFM15].

In addition, the performance of the SG method has been enhanced by performing the avoidance trajectory in such a way that the configuration of all obstacles are considered, not simply the nearest one. Moreover, the distance to obstacles on both sides of the heading direction is taken into account. By this means, the smoothness and stability of the robot's motion are increased, especially in unstructured narrow spaces. This has been possible by introducing and integrating two concepts, called "tangential" and "gap flow" navigation. Using the "tangential navigation", the robot moves tangential to the obstacles boundary. With the "gap flow navigation", the robot safely and smoothly navigates in-between closely spaced obstacles. In both concepts, avoiding collisions and approaching the target are simultaneously performed. Another important contribution is the computation of the motion controller, that drives a mobile robot towards a given goal, in such away that the stability of the system is proved in the Lyapunov sense. The enhanced approach, entitled "Tangential Gap Flow" (TGF) navigation, safely guides a mobile robot through cluttered environments with smoother and more efficient trajectories when compared to state-of-the-art techniques [MFM16].

Moreover, this work presents a new concept, the "Admissible Gap" (AG), which addresses the question of whether a given gap is traversable by performing a collision-free motion control that respects the shape and vehicle constraints. The AG concept has been successfully employed to develop a collision avoidance approach, that achieves an outstanding performance in cluttered scenarios. This has been possible by directly respecting the vehicle constraints rather than adapting a holonomic-based solution. A key idea of AG is the creation of an "admissible gap", which serves as a bridge obeying the vehicle constraints, once traversed, the robot makes progress towards the target [MM17]. To this end, a new methodology for traversing gaps has been proposed in such a way that the vehicle constraints are respected. This methodology provides a compromise between safety and efficiency. Unlike existing methods, AG is directly applied

to the workspace without having to construct an abstraction layer. Another important contribution of the AG approach is the development of a new procedure for finding out gaps. The method can be applied to full or limited field of view sensors. Moreover, it discards useless gaps, reducing the possibility of oscillation and improving the stability of navigation [MFM18].

Finally, various experiments are provided in cluttered, dense, and complex environments, utilizing our rescue mobile robot GETbot. A performance evaluation is also carried out to quantitatively assess the effectiveness of the proposed approaches. Additionally, a discussion and comparison with existing state-of-the-art techniques is conducted on the basis of the limitations mentioned in section 1.1. In addition to the experiments presented in this thesis, the proposed approaches were extensively tested while preparing for and participating in the Robocup Rescue Robot League competitions [rob19]. In these competitions, robots developed by international research teams operate in a replicated disaster environment, how it could appear after an earthquake or a terror attack. In such scenarios, it is of high priority to quickly explore the terrain and identify and locate injured people, so that they can be evacuated immediately. Since 2012 the navigation algorithms proposed in this thesis have been used as the reactive layer on our rescue mobile robots, contributing to the success of the team GETbots. During these years several honors have been achieved: third place in the overall competition at the RoboCup German Open 2012, "Best in Class Mobility" award in the RoboCup German Open competitions 2013 and 2014, and "Best in Class Manipulation" award in the RoboCup German Open 2015. However, the greatest success was the outstanding 3rd place (best European team) in our first participation at the 2016 RoboCup world championship. In this competition, employing the AG approach, proposed in chapter 6, had the greatest impact on the performance of the team. This is due to the fact that the scores of the given tasks were multiplied by two, once they were performed autonomously.

Partial results of the work presented in this thesis have been previously published in nine different peer-reviewed international conference proceedings and journals. More specifically, partial results of the "Safe Gap" (SG), "Tangential Gap Flow" (TGF), and "Admissible Gap" (AG) navigation approaches presented in chapters 3 - 6 have been published in [MFM13b] [MFM13a], [MJFM13] [MFM15] [MFM16]

[MFM17], and [MM16] [MM17] [MFM18], respectively. It is worth mentioning that all published papers were co-authored with Bärbel Mertsching. Additionally, papers [MFM13b] [MFM15] [MFM13a] [MJFM13] [MFM16] [MFM17] [MFM18] were co-authored with Dirk Fischer. Finally, [MJFM13] was co-authored with Hussein Jaddu. In all cases, the key ideas, main contributions, experimental setups, data analysis, and writing were performed by the author of this thesis.

1.3 Thesis Outline

The remaining chapters of this thesis are structured as follows:

Chapter 2 discusses the problem of autonomous mobile robot navigation and reviews some of the basic concepts used throughout the robotics literature and this thesis. In addition, a literature survey on mobile robot navigation is presented, classifying the existing techniques into path planning (global) and reactive navigation (local). At first, a brief overview of the path planning approaches is introduced. Following this presentation, we shed the light on the most popular reactive navigation approaches that motivated us to formulate this work, showing their advantages and drawbacks.

Chapter 3 introduces the "Safe Gap" (SG) navigation approach for autonomous mobile robots navigating in unknown dense environments. It presents a procedure to check if there is a safe way towards the goal. Otherwise, the robot will be directed to another location, referred to as a *subgoal*. This chapter also introduces a methodology for analyzing the structure of obstacles to locate the list of surroundings openings. It is also described how subgoals are located within free areas by making use of the gap analysis. Moreover, it is shown how to determine the steering command that drives a mobile robot towards the goal (resp. subgoal). Finally, this chapter shows different simulations and experiments, demonstrating the strength of the proposed SG navigation approach.

Chapter 4 discusses the "Tangential Gap Flow" (TGF) navigation approach that is an evolution of the SG method. At first, it describes the criteria employed to characterize the current motion situation and its corresponding action. Subsequently, the concept of "tangential navigation" is introduced, considering only

one obstacle point. The concept of "gap flow navigation" is also presented which, together with the tangential navigation, provide a foundation of the TGF approach. This chapter also explains how both concepts are integrated and how the avoidance angle is computed based on all detected obstacles. In addition, in this chapter, it is described how to set the control commands that guide a robot towards a given target. Experimental results are also introduced to demonstrate the power of the proposed TGF approach.

Chapter 5 introduces a performance evaluation to assess the significance of the developed approaches over their counterparts. Moreover, this chapter presents additional experiments using the implementation of the TGF approach in addition to three state-of-the-art methods; CG [MFMJ10], SND [DB08], and ND+ [MM04]. The TGF method has been selected since it outperforms the SG method and inherits its advantages. Before presenting these experiments, the experimental setup is shown. This chapter also describes the performance metrics that are employed to evaluate the execution of the proposed methods. Finally, this chapter discusses and compares the behavior of all discussed methods.

Chapter 6 presents an approach that considers the exact robot shape and kinematic constraints. It introduces a new procedure of finding out gaps which works for both full and limited field of view sensors. The "admissible gap" concept is then presented, where a new methodology for traversing gaps is also introduced. Moreover, this chapter shows how this concept has been successfully employed to develop a collision avoidance approach, that obeys the vehicle constraints. It also shows how to compute the control commands, which guide a mobile robot towards its goal in a kinematically constrained manner. Finally, experimental results including a discussion and comparison with existing state-of-the-art methods are introduced.

Chapter 7 draws the conclusions of this thesis and presents recommendations for future work in this field of research.

2 Autonomous Mobile Robot Navigation

2.1 Introduction

Some of the major, yet critical issues confronting robotics researchers lie in the context of navigation. In most cases, autonomous navigation determines the success of the complete mission and any failure in this module may have fatal consequences upon the robot and the environment. It is interesting to notice that this research domain has been initiated in the late 1960s with the appearance of mobile robots [MLL08]; the first navigation systems were based on seminal ideas initially published in the first International Joint Conferences on Artificial Intelligence (IJCAI) founded in California, 1969.

The general navigation methodology for any robotic system is a typical feedback process [Cao04]: the robot perceives the environment through sensors providing information to the controller, which is analyzed to understand the current environmental situation. Then, the controller sends out commands to the manipulator to carry out behaviors accordingly. As the robot moves from one place to another, the working environment keeps changing, and therefore, sensors continuously receive new information providing it to the controller and the process repeats. Several constraints may affect the robot's behavior and make it harder to develop a robust navigation algorithm [Sin97], such as: the computational power (prohibits us from achieving real-time performance), the presence of obstacles (may be dynamic ones), the inaccuracy of the perception system, and the errors caused by the robot mechanical system.

It turns out from the above mentioned constraints that autonomous navigation is a challenging and complex problem. However, the complexity of navigation can be reduced by dividing it into smaller portions (modules) to deal with them

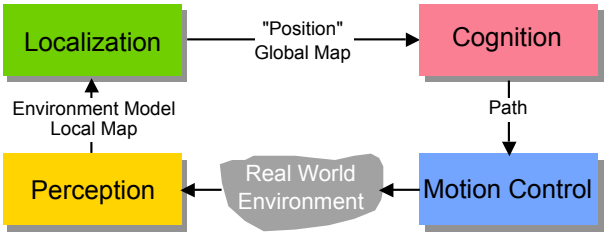


Figure 2.1: Main stages of the navigation process (originally from [SN04]).

independently and then combine the solutions. In an early development stage, Lenoard and Durrant [LDW91] explained the problem of robot navigation by three questions: "where am I?", "where am I going?", and "how should I get there?". These questions correspond to the following three problems, respectively: robot localization; determining own current position in an environment, cognition; deciding what actions are necessary to reach a target based on its specifications, and motion planning; generating a continuous path between an initial position and a prescribed goal location. Along such a path, the robot must avoid collision with obstacles. Solving the above mentioned problems requires the robot's ability to acquire information about its environment, which is usually known as robot perception. This defines the fourth portion of the navigation process as stated in [SN04] (see figure 2.1).

The third problem (motion planning) is the core of this work and will be the subject of the next sections. A discussion of the other problems is out of the scope of this thesis, and the reader may refer to [SN04] for more description. Motion planning has been thoroughly investigated in the literature and has been traditionally addressed from two distinct perspectives, the global and local motion planning. Global planners, also known as path planning methods, are based on a priori information, where an optimal path that connects a robot to a target location is computed. Local planners, on the other hand, compute one action at each time step using the information available from sensors. These algorithms, known as reactive or obstacle avoidance methods, are more realistic in real-time implementation and consists of a direct mapping between the sensor data and the motion control [Ark98].

In Section 2.2, a few basic concepts that are necessary for designing a navigation technique are discussed. Following this presentation, a brief overview of the path planning approaches is introduced in section 2.3. Finally, in section 2.4 we shed the light on the most popular reactive navigation approaches that motivated us to formulate this work, showing their advantages and drawbacks.

2.2 Basic Concepts

In this section, some of the basic concepts used throughout the robotics literature and this thesis are discussed; these include: configuration space, configuration space obstacle, notion of a path and a trajectory, non-holonomic motion planning, Lyapunov stability theory, and sensory data.

2.2.1 Configuration Space

In order to plan a safe motion for a robot, we must be able to identify its location with respect to the working environment. In the context of motion planning, a key concept for specifying the location of every point on a robotic system is known as a *configuration* q [CLH$^+$05]. For instance, the configuration of a disc-shaped robot which is only able to translate (no rotation) in a two-dimensional Euclidean space (known as *workspace*), can be described by the location of its center of gravity (x, y). If we know the robot's radius, we can easily identify the position of all points occupied by the robot from the configuration of its center. For a polygonal robot which is able to rotate and translate in a two-dimensional workspace, at least 3 parameters are required to represent its configuration; the location of a point on the robot and the orientation θ of the robot's coordinate system relative to a fixed coordinate system in the workspace.

The *configuration space* ($\mathcal{C}_{\text{space}}$) of a robotic system is described as the space of all possible robot configurations [CLH$^+$05]. Therefore, a configuration q can be seen as a point in $\mathcal{C}_{\text{space}}$. For the non-rotating robot described above, the workspace and the configuration space can be both represented by $\mathbb{R}^2$. However, for the rotating mobile robot the workspace is two dimensional $\mathbb{R}^2$, whereas the

configuration space is three dimensional $\mathbb{R}^3$ or more specifically $\mathbb{R}^2 \times \mathcal{S}^1$, where $\mathcal{S}^1$ is the unit circle. Indeed, the dimension of the configuration space equals the number of degrees of freedom of the robotic system.

2.2.2 Configuration Space Obstacle

A key concept for motion planing is the *configuration space obstacle* which is described by the set of all locations in the workspace at which an intersection between the robot and an obstacle may exist [CLH+05]. Thus, an obstacle in the workspace maps in $\mathcal{C}_{\text{space}}$ to the configuration space obstacle. More formally, let $\mathcal{W} = \mathbb{R}^m$ be the workspace, $\mathcal{O} \in \mathcal{W}$ the set of obstacles, and $\mathcal{A}(q)$ the region of $\mathcal{W}$ covered by the robot at configuration q. A configuration space obstacle, denoted by $\mathcal{CO}_\text{i}$, that corresponds to obstacle $\mathcal{O}_i \in \mathcal{O}$ is defined as follows:

$$\mathcal{CO}_i = \{q \in \mathcal{C}_{\text{space}} \mid \mathcal{A}(q) \cap \mathcal{O}_i \neq \phi\}$$

The *configuration space obstacles* is the union of all $\mathcal{CO}_i$:

$$\mathcal{C}_{\text{obstacles}} = \bigcup_{i=1}^{q} \mathcal{CO}_i$$

The *free configuration space* is the relative complement of $\mathcal{C}_{\text{obstacles}}$ in $\mathcal{C}_{\text{space}}$ (i.e locations at which no intersection between the robot and obstacles $\mathcal{O}$ exists):

$$\mathcal{C}_{\text{free}} = \mathcal{C}_{\text{space}} \setminus \mathcal{C}_{\text{obstacles}} = \left\{ q \in \mathcal{C}_{\text{space}} \;\middle|\; \mathcal{A}(q) \cap \left(\bigcup_{i=1}^{q} \mathcal{O}_i \right) = \phi \right\}$$

As an example, look at figure 2.2 which shows a triangular mobile robot that is allowed to translate in a two-dimensional workspace $\mathcal{W} = \mathbb{R}^2$ without rotation. In such a case, the configuration space is also $\mathbb{R}^2$ as we have pointed out in section 2.2.1. $\mathcal{C}_{\text{obstacles}}$ is obtained by moving the robot along the boundary of each object, figuring out the constraints the obstacle causes on the robot's configuration. In other words, the location of a reference point on the robot is

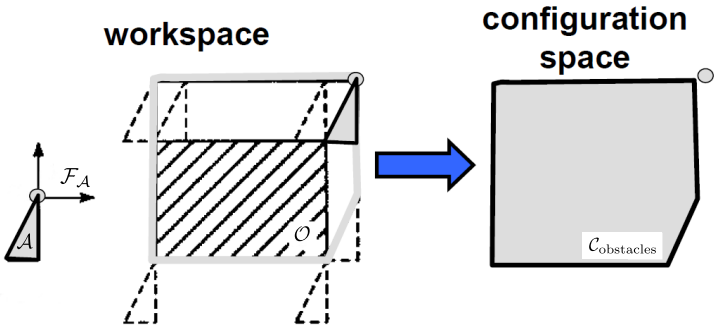


Figure 2.2: A triangular mobile robot $\mathcal{A}$ (left image) that is allowed to translate freely in a two-dimensional space at a fixed orientation. The reference point of $\mathcal{A}$ is marked as a small circle. The configuration space obstacles $\mathcal{C}_{\text{obstacles}}$ (right image) is obtained by enlarging the workspace $\mathcal{W}$ (hatched area) by the shape of $\mathcal{A}$ (middle image) (originally from [Lat91]).

marked along the boundary of each obstacle, which results in enlarging $\mathcal{O}$ by the shape of the robot as shown in figure 2.2.

2.2.3 Notion of a Path and a Trajectory

With the configuration space concept introduced above, the motion planning problem is turned out from finding a collision-free path in $\mathcal{W}$ for a complex-shaped robot to finding a path τ in $\mathcal{C}_{\text{space}}$ for a point-like robot. Hence, a path is transformed from swept volume to a one-dimensional curve (see figure 2.3). A path from an initial configuration q_{start} to a goal configuration q_{goal} is defined as a continuous mapping [Lat91]:

$$\tau : [0,1] \rightarrow \mathcal{C}_{space}, \quad \text{with: } \tau(0) = q_{\text{start}},\ \tau(1) = q_{\text{goal}}.$$

For a mobile robot that is a allowed to translate and rotate freely in the space (free-flying robot), any path defined above is feasible if the configuration space is free from obstacles. A path $\tau(s) \in [0,1]$ is a collision-free path if for all $s \in [0,1]$, $\tau(s) \in \mathcal{C}_{\text{free}}$.

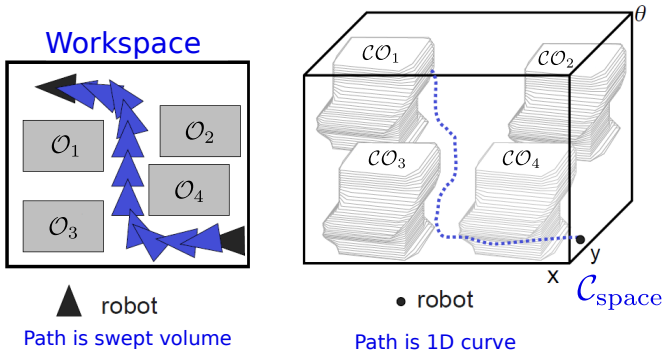


Figure 2.3: Path planning for a triangular robot that is allowed to translate and rotate in a two-dimensional space. The configuration space in such a case is $\mathbb{R}^2 \times \mathcal{S}^1$, where $\mathcal{S}^1$ is the unit circle. Planning a path for the triangular-shaped robot in the workspace (left image) is equivalent to planning a path for a point-like robot in the configuration space (right image) (originally from [Pla10]).

While a path is a pure geometric description of motion, a trajectory specifies a timing law on a path [SSV09], and therefore the velocity and/or acceleration at each point is described.

2.2.4 Non-holonomic Mobile Robots

Every mobile robot is subject to a variety of constraints which may limit its motion. Traditionally, these constraints are classified into integrable and non-integrable. An integrable constraint, referred to as *holonomic*, only depends on the position and time and can be expressed as a configuration constraint. A holonomic equality constraint can be written in the following form [Lat91]:

$$F(\mathbf{q}, t) = F(q_1, ..., q_m, t) = 0 \tag{2.1}$$

where F is a smooth function with non-zero derivative.

A mobile robot subject to holonomic constraints, called a holonomic mobile robot, can move to any location following any direction (i.e. it can move for-

ward, backward, or sideway). Therefore, no change to the basic motion planning problem (which assumes a free flying behavior) is required. An example of a holonomic system is a person walking on the ground who can instantly go towards the left or right, as well as moving forwards or backwards.

The non-integrable constraints, referred to as *non-holonomic*, cannot be expressed as a functional relationship between the configurations as in Eq. (2.1). This is due to the fact that these constraints depend on the velocity of the robot. A non-holonomic equality constraint takes the following form [Lat91]:

$$F(\mathbf{q}, \dot{\mathbf{q}}, t) = F(q_1, ..., q_m, \dot{q}_1, ..., \dot{q}_m, t) = 0 \tag{2.2}$$

where F is a non-integrable smooth function. Notice that a constraint defined by Eq. (2.2) can be formulated similar to Eq. (2.1) if all velocity parameters $(\dot{q}_1, ..., \dot{q}_m)$ can be eliminated (i.e. if it is integrable).

Non-holonomic constraints reduce the dimension of the velocity space, but not that of the configuration space (a constraint on velocity does not mean a constraint on configuration). Therefore, a mobile robot subject to non-holonomic constraints, named a non-holonomic mobile robot, can reach any configuration, but not following any trajectory [MLL08]. An example of a non-holonomic robot is a car-like mobile robot that can move forwards or backwards, and can make turns. Such a robot cannot move freely in the workspace since no direct translation to either of its sides is possible. However, it can reach any location in the workspace. Another example is a differential-drive mobile robot which may turn on spot (zero turning radius) and can only move perpendicular to the wheels axis, see figure 2.4. It has been shown in the literature that the motion of these robots is constrained by [LSL98]:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{2.3}$$

where (x, y, θ) represents the robot's configuration (location and orientation) relative to the world coordinates.

In order to describe the configuration of a non-holonomic robot, at least three parameters are required (e.g. the location of the center of the robot, and the

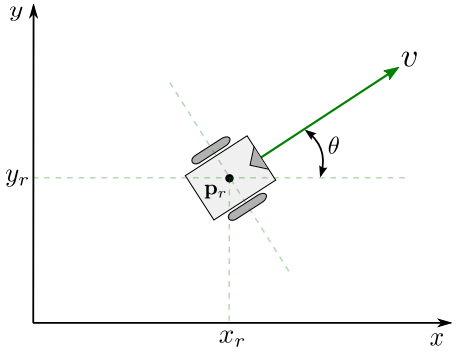


Figure 2.4: A differential-drive mobile robot, which can only move perpendicular to the wheels axis.

angle between its x-axis and the x-axis of a fixed coordinates system (global) in the workspace). However, at a specific configuration, the robot's motion is described by only two parameters (e.g. linear and angular speeds). For instance, the kinematic model of the differential-drive robot shown in figure 2.4 can be expressed by the following equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w \tag{2.4}$$

where v and ω are the translational and rotational velocities.

Planning a motion between two configurations for a non-holonomic robot is a hard task even in the absence of obstacles. The problem of motion planning for these robots can be described in the following [MLL08]: given a mobile robot subject to non-holonomic constraints, starting and goal locations, and a geometric description of the robot and obstacles, compute an admissible continuous sequence of collision-free robot configurations that connect the starting to the goal locations. Solving this problem requires to consider two types of constraints; constraints due to the existence of obstacles and constraints due to the non-holonomicity. The former are represented in the configuration space, whereas the latter are expressed in the velocity (tangent) space.

2.2.5 Lyapunov Stability Theory

Stability is the base and most important requirement of any control system, since an unstable system is normally of no use and probably risky. Loosely speaking, "a system is described as *stable* if starting the system somewhere near a desired operating point implies that it will stay around the point ever after" as stated in [SL91]. A powerful and generic methodology for checking the stability of nonlinear control systems is the theory proposed by Alexandr Lyapunov in the late 19th century . It includes two methods; the so-called *linearization* method (indirect) and the *direct* method. Lyapunov's indirect method is based on the assumption that, for small motions, the behavior of nonlinear control systems is similar to their linearized approximations. In this regard, a stable design of the linearized control system implies the stability of the original system locally. The direct Lyapunov method determines the stability characteristics of nonlinear systems by examining the variation of an energy-like function (Lyapunov function) over time without explicitly solving the differential equation. In the following, some basic definitions from the literature are introduced, followed by the main stability result of the Lyapunov theory. For more details, the interested reader can refer to a standard text, such as [SL91] [Kha14].

Throughout this section, a nonlinear dynamical system is described as follows:

$$\dot{x} = f(x, t) \tag{2.5}$$

where $f(.) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and x represents a state space vector. A solution $x(t)$ to Eq. (2.5) is represented by a curve in the state space, referred to as state or system trajectory. It is assumed that $f(x, t)$ satisfy the conditions needed to guarantee the existence and uniqueness of solutions, such as $f(x, t)$ is Lipschitz continuous with respect to x [Oeg03]. The points of interest in the context of stability analysis are the so-called equilibrium points.

Definition 2.1 (Equilibrium Point) *A state x^* is called an equilibrium point of Eq. (2.5) if once the condition $x(t) = x^*$ is fulfilled, it never changes ($x(t)$ stays equal to x^* ever after):*

$$f(x^*, t) = 0$$

We now go further by studying the stability properties of a given equilibrium state. Since it is possible to move an equilibrium point to the origin by a simple coordinate transformation, we will assume $x^* = 0$ throughout this section.

Definition 2.2 (Equilibrium Point Stability) *An equilibrium state $x^* = 0$ is described as a stable point at $t = t_0$ if a $\delta > 0$ exists for any $\sigma > 0$ such that:*

$$\|x(t_0)\| < \delta \implies \|x(t)\| < \sigma, \forall t \geq t_0$$

else, the equilibrium point is described as unstable.

Definition 2.3 (Asymptotic Stability) *An equilibrium point of Eq. (2.5) is described as an asymptotically stable point at $t = t_0$ if it is convergent and fulfills definition 2.2 (stable), where an equilibrium point is said to be convergent if:*

$$\|x(t_0)\| < \delta \implies \lim_{t \to \infty} \|x(t)\| = 0$$

In robotics, we are almost interested in asymptotically stable equilibria, since our objective is to move a robot to a given goal point, not merely remain nearby.

Definition 2.4 (Locally Positive Semi-definite Function) *A continuous function $V(x) : D_\epsilon \to \mathbb{R}$ is described as locally positive semi-definite in D_ϵ if*

(i) $V(0) = 0$

(ii) $V(x) \geq 0, \ \forall x \in D_\epsilon, x \neq 0$

where $D_\epsilon = \{x \in \mathbb{R}^n : \| x \| < \epsilon\}$ represents an open and connected subset of $\mathbb{R}^n$ centered at the origin.

Definition 2.5 (Locally Positive Definite Function) *A continuous function $V(x) : D_\epsilon \to \mathbb{R}$ is described as locally positive definite in D_ϵ if condition (ii) in definition 2.4 is replaced by: $V(x) > 0, \ \forall x \in D_\epsilon, x \neq 0$*

Definition 2.6 (Negative Definite Functions) *A continuous function $V(x) : D_\epsilon \to \mathbb{R}$ is described as locally negative definite (resp. semi-definite) in D_ϵ if $(-V(x))$ is locally positive definite (resp. semi-definite) in D_ϵ.*

Theorem 2.1 (Lyapunov Stability Theorem) *Suppose that $x^* = 0$ is an equilibrium point of the differential system defined in Eq. (2.5), and suppose that $V(x) : D_\epsilon \to \mathbb{R}$ is a continuously differentiable function, where:*

(i) $V(x)$ is locally positive definite in D_ϵ.

(ii) The derivative of $V(x)$ is locally negative semi-definite in D_ϵ.

thus, the equilibrium point $x^* = 0$ is described as stable.

Theorem 2.2 (Lyapunov Asymptotic Stability Theorem) *Considering the same conditions of the above theorem (theorem 2.1), if $\dot{V}(x)$ is locally negative definite, $x^* = 0$ is said to be asymptotically stable.*

The function $V(x)$ in both theorems above is called a Lyapunov function.

2.2.6 Sensor Data

An important task of a mobile robot is to get knowledge about itself and its surroundings. This is achieved by extracting useful information from the measurements provided by sensors. There are various types of sensors used in mobile robots ranging from those used to measure simple values like wheel load to more sophisticated sensors like those used for perceiving the environment. Generally speaking, sensors can be classified to *proprioceptive* and *exteroceptive* [SN04]. Proprioceptive sensors are used to get information about the current state of the robot itself, like its location, speed, battery voltage, etc. On the other hand, exteroceptive sensors acquire data from the outside world, such as color, distance to an obstacle, global position, etc. In the following, we briefly discuss the most common sensors used in the context of mobile robot navigation. For more information about other sensor types, the reader can refer to [LMD+98].

2.2.6.1 Shaft Encoders

Shaft encoders provide information about the distance traveled by a mobile robot based on measuring the number of revolutions of its wheels. The robot uses the

output of the shaft encoders to estimate its current position relative to a starting location, a method called *odometry*. However, this method is prone to errors due to the wheels slippage, sampling of the encoders, and the integration of velocity measurements over time. Obtaining a better position estimation requires to combine information from shaft encoders and an exteroceptive sensor, such as a camera or a laser rangefinder.

2.2.6.2 Vision-based Sensors

Vision is the most powerful sense due to the rich amount of data that can be extracted from an image [And08]. Recently, a considerable improvement in accuracy, resolution, and frame rate of vision sensors (e.g. camera) has been shown. Robots are using either global or local vision. In global vision, a camera sensor is placed externally to the robot in such a way that the robot (maybe more than one robot) and the entire environment are covered by the camera's field of view (FOV). In local vision, a camera is attached to the robot so that the area in front of it is captured. By this means, regions that are occluded from the global camera can be discovered. Despite the fact that vision-based sensors provide detailed information about an environment (which may not be available using combinations of other types of sensors [DK02]), unfortunately, obtaining accurate range information in real-time for successful obstacle avoidance does not seem to be currently possible [Hoy14].

2.2.6.3 Range Sensors

Range sensors are used to measure distances to objects surrounding the robot by making use of the propagation speed of a transmitted signal [Kuc06]. The emitted signal can be sound as in ultrasonic (sonar) sensors or light as in LiDAR (Light Detection and Ranging) . The basic principle of ultrasonic (resp. LiDAR) sensors is to emit ultrasonic pressure waves (resp. laser light beams) and calculate the time it takes to reflect and get back to the receiver. With the speed of sound (resp. light) known, the distance to the reflected surface follows immediately. A major disadvantage of an ultrasonic sensor is the inability to determine the exact

location of an obstacle. It only tells us that an obstacle is located within the area of the measurement cone. The LiDAR outperforms the ultrasonic sensor in terms of resolution, accuracy, and cost, owing to the use of laser light instead of sound waves [SN04]. Therefore, it is the most common sensor used for mobile robot navigation and obstacle avoidance.

2.3 Path Planning Techniques

Path planning is a relatively well studied research area. It is defined as a *priori* determination of the motion strategy based on a *perfect model* of the robot and a *complete knowledge* of the environment (i.e. a map). This problem has been motivated by the industrial use of manipulator arm robots operating via an end effector that can freely move in a known environment. In mobile robots operating in real-world scenarios, the a priori knowledge of an environment is mostly partial or absent [Rib05]. Moreover, it has been shown that path planning algorithms are computationally expensive, limiting their use in applications requiring real-time performance [PJK12]. In addition, incorrect data or unexpected changes in the environment (e.g. dynamic obstacles) may affect the performance since, in such a case, it is essential to recompute the path.

In this section, we review the main path planning strategies that have been developed over the past years. For a thorough description of basic methods, the reader can refer to [Lat91] [PCY+16].

2.3.1 Roadmap Path Planning

Roadmap path planning approaches capture the connectivity of the unoccupied space into a network of one dimensional curves or lines [SN04], named *roadmaps*. Once a roadmap is constructed, the path planner attempts to connect a starting configuration of the robot to the roadmap. Similarly, a given goal configuration is connected to the roadmap. Then, the roadmap is searched for a series of roads from the starting configuration to the goal. In a roadmap, the shape of an obstacle is represented as a polygon. There are several variations that follow

this strategy which differ mainly in the way the roadmap is constructed. Next, we describe two approaches, namely *visibility graph* and *Voronoi diagram*.

2.3.1.1 Visibility graph

Such a graph consists of edges connecting all visible vertices of obstacles (there exists a line of sight between both vertices) including both the start and goal configurations [KJIF06]. The shortest path from the initial location to the goal along the roads of the visibility graph is then calculated using any graph searching technique. This algorithm provides an optimal solution to the path planning problem. However, it takes the robot as close as possible to obstacles [SN04]. Moreover, it does not generalize to higher dimensions, nor it scale well with the number of obstacles.

2.3.1.2 Voronoi diagram

The Voronoi diagram consists of lines and parabolic segments that are equidistant from the two nearest obstacles, called *Voronoi edges*. Similar to the visibility graph method, a graph searching technique is used to compute the best path along the Voronoi edges that connect the start configuration to the goal. Although the distance between the robot and obstacles is maximized along the way to the goal, this method is usually far from optimal in the sense of total path length [SN04].

2.3.2 Graph and Grid based Methods

A common approach to global path planning is based on graph search, where the configuration space is discretized into a regular grid. Each cell of this grid is considered a node and each connection between two cells is an edge. The obstacle-free portion of the grid is searched for the shortest path that connects the robot's initial location to the goal. The standard graph-based techniques are the Dijkstra's search [Dij59] and A* algorithm [HNR68] [Ota09] [DTMD10]. While Dijkstra's method uses a depth-first search strategy, A* employs a heuristic

function to perform a best-first search. This function helps in selecting the most promising node by estimating the distance to the goal. Hence, A* is more efficient than Dijkstra, particularly if the size of the grid is large. The solution found by A* is optimal if the heuristic function is optimistic, i.e. it never overestimates the actual path cost (e.g. Euclidean or Manhattan distance).

Grid-based methods are considered computationally expensive, especially in complex and large environments. This is due to the grid representation of the map. Moreover, when the goal position is modified or when a dynamic obstacle obstructs the robot's working area, it is necessary to re-plan the entire path [PJK12]. However, many variants to A* have been introduced to make the searching process faster, such as Incremental A* [KL02], ARA* [LGT03], D* [Ste95], D* Lite [KL05], and Anytime D* [LFG+08]. The main idea behind these extensions is to maintain dependency information so that the previous search results are modified locally when environmental changes occur. In this regard, no need to explore the entire space at each re-planning stage.

Another problem of grid-based techniques is that the resultant path is basically aligned to the grid structure (could be 4 or 8 connected grid), which is unnatural and sub-optimal in a continuous sense. It has been shown that a path can be up to 8% longer than optimal in case of an 8 connected grid. A variant of D*, named Field D*, has been proposed in [FS06] to address this problem. Field D* modifies the search graph by assigning nodes to the corners of the grid cells rather than their centers. In this regard, edges connecting two adjacent nodes will have the same traversal cost. The path cost along any point of an edge is estimated using a linear combination (interpolation) between the nodes of the edge. Thus, in cases where the cost variation is not linear between nodes, this heuristic may fail. However, it has been shown that Field D* works quite well in practice [FS06].

2.3.3 Safe Interval Path Planning

The existence of dynamic obstacles adds a new dimension (time) to the search space, and thus increases the computational overhead of the planning problem. A

common approach to deal with the increase in the dimensionality of the planning problem is to assume that all objects are stationary and re-plan continuously as objects move (e.g. [LFG+08] [RFS09]). However, this approach scarifies completeness and optimality. Another strategy is to plan in the complete search space as has been addressed by Silver [Sil05]. By this methodology, plans can be out of time before applying them due to the increased number of states to be processed. This problem was the motivation behind developing the Safe Interval Path Planning (SIPP) approach [PL11]. The time dimension in the SIPP path planner is represented by intervals rather than time steps. A *safe interval* corresponding to a configuration of a robot is a period of time consisting of a group of contiguous time steps, during which the associated configuration is collision-free. The planner uses states defined by configurations, together with their corresponding safe intervals to construct a search-space. For planning, a modified A* algorithm is used, incorporating the time needed to attain a node. Due to the fact that the number of SIPP intervals is less than that of the time steps for each configuration, SIPP is faster and requires smaller memory as compared with those approaches that use time steps for planning.

Anytime version of SIPP [NPL12] has also been developed integrating SIPP with ARA* [LGT03]. The integrated method introduces an optimal time horizon, after which the time dimension is dropped. Thus, the path planner considers only spatial coordinates for states having a time stamp exceeding the time horizon. As reported in [Seb14], theoretically, "it is shown that in the absence of time horizon, this planner can provide guarantees on completeness as well as bounds on the sub-optimality of the solution with respect to the original space-time graph".

2.3.4 Probabalistic Roadmap

Some of the most widely used path planners are those based on the concept of a Probabilistic Roadmap (PRM) [KSLO96]. It has been shown that PRM is a successful approach to robot path planning in high-dimensional configuration spaces, which naturally arise when we control the motion of high degrees-of-freedom robots. Basically, PRM methods work in two main phases, namely a "roadmap construction phase" and a "query phase" [KSLO96]. The roadmap

construction phase captures the connectivity of the unoccupied space by first randomly sampling the configuration space and identifying those samples that lie in the free space, referred to as *free samples*. Then, nearby free samples are connected using a local planner (i.e. they are connected if there is a feasible motion command between them). The output of this stage is a graph whose vertices are the free samples and whose edges are the successful local plans. In the query phase, the start and goal locations are connected to the graph. For these two configurations, the nearby nodes are determined and edges are connected using a local planner as in the roadmap construction phase. Once the complete graph is created, a graph search algorithm (e.g. A*) is used to find the path on the graph that connects the initial and goal configurations.

The PRM approach is flexible owing to the fact that each of its main components (sampling method, roadmap construction strategy, and the local planner used) can be modified while maintaining its basic capabilities. Thus, many PRM variants have been introduced during the last two decades. For example, some variants extended the problem to non-holonomic path planning using special local planners (e.g. [SO97]). Other variants focused on speeding up the nearest neighbor search by using an approximation method (e.g. [AI08] [ML09] [RJ15]). The motivation behind developing these approaches was the fact that the nearest neighbor search is considered as a crucial part of PRM planners, since each time a new node is connected to the roadmap the planner must find the set of nearest neighbor nodes. Lately, several variants have been developed to provide *probabilistic completeness* guarantees in the sense that the probability of finding a solution, if one exists, goes to one as the number of samples goes to infinity [KF11] [JSCP15].

2.3.5 Rapidly-exploring Random Tree

Rapidly-exploring Random Trees (RRT) [LaV98] [LK01] [KFT$^+$08] are randomized planning methods that incrementally construct a search tree and attempt to quickly explore the state space. This enables them to find feasible paths in higher-dimensional search spaces more efficiently, but with less-strict optimality and completeness guarantees than search-based planners. The RRT planner is

similar in spirits to the PRM approach, but instead of constructing a complete topological graph (roadmap), a tree is grown incrementally staring from an initial configuration. It is constructed by randomly selecting a new sample at each iteration and then determining the nearest neighbor in the already created tree. A new vertex is then added to the tree by extending the nearest neighbor towards the selected sample. This procedure is repeated until having reached the goal. At the end, the nodes of the tree represent the explored locations and the edges represent the control inputs required to proceed from a node to another. Thus, the tree is constructed by a random exploration towards unexplored areas biased by motion towards the goal configuration. The efficiency of the search can be enhanced by employing a bidirectional search where the tree is extended from both the initial and goal configurations [KL00].

Several recent variants, referred to as RRT*, initially proposed in [KF11] [KF10], guarantee convergence toward optimal solutions as the number of samples grows [PPK+12] [GPPK13] [KF13]. Other variants build their analysis based on the notion of convergence in probability which provides mathematical flexibility allowing for convergence rate bounds [SJP15] [JSCP15]. Another recent variant [Hau15] attempts to speed up the sampling-based motion planners by using a lazy collision checking strategy. However, RRT-based methods are mostly limited to static environments [PPM13]. Moreover, their computed paths are post-processed to reduce the effect of randomization [DSS+13]. Although there are attempts to extend RRT-based planners to handle dynamic scenes (e.g. [PF05] [KRSV10] [Hau12]), local-reactive and control-coupled approaches are still more desirable, particularly in highly cluttered and dynamic environments.

2.4 Reactive Navigation Techniques

Besides the extensive developments in path planning, efforts were devoted towards making mobile robots operate out of the artificially created environments, so that they can share the same workspace with humans. The high computational time required to plan a path, inaccurate modeling of the environment, and the unforeseen dynamic obstacles made researchers aware of the gap between path planning and motion execution [MLL08]. This has motivated authors to develop

schemes fast enough so that robots can react to the environment and avoid collision with obstacles. These approaches goes under many names such as reactive navigation, obstacle avoidance, local navigation, and sensor-based control. Reactive navigation techniques are based on a perception-action process [MM02], where actions are generated iteratively (at each control cycle) using up-to-date model of the world. Typically, the model of the world is constructed based on feedback sensors (e.g. laser rangefinder). These approaches are computationally more efficient as compared to path planning algorithms since only a part of the world model is required, and therefore relatively less information needs to be processed at each time step. Thus, robots can respond on time to obstacles, and behave especially well in unknown environments. It is worth to mention that the work presented in this thesis belongs to this group of approaches.

One could not hope to cover all techniques that have been proposed. Hence, the focus is limited to some representative methods, including those that have proved popular across the years and those that have been introduced lately.

2.4.1 Bug Algorithms

The Bug algorithm and its variants (e.g. [LS86], [KR97], [MSZ09], [MHS13], and [MLB15]) are among the simplest and earliest reactive navigation techniques [MFM18]. The main idea of these methods is to steer the robot towards the target unless an object is met, in which case, the robot moves unidirectionally along the boundary of the object until navigation towards the target is once again possible [CLH$^+$05] (fulfilling a leaving condition). Switching the motion mode (from boundary following to goal pursuit and vice versa) follows a globally convergent criterion [YP09]. Many variants of the basic Bug algorithm have been proposed, but essentially all these methods differ in defining the rule under which the transition between both motion modes is triggered . A detailed comparison of several Bug algorithms is carried out in [NB07].

With the Bug 1 algorithm [LS86], the robot moves along the straight line connecting the robot to the goal until it encounters an obstacle at a point, referred to as a hit point H_i (see points H_1 and H_2 in figure 2.5). Then, the robot fully

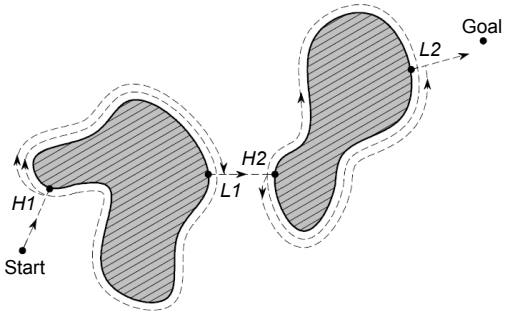


Figure 2.5: Bug 1 algorithm with (H_1, H_2) described as hit points, and (L_1, L_2) described as leave points (from [SN04]).

circumnavigates the obstacle and keeps track of the obstacle point closest to the goal, referred to as a leave point L_i. From this point, the robot leaves the obstacle and resumes the progress towards the goal again.

With the Bug 2 algorithm [LS86], the robot begins to follows the contour of the obstructed obstacle until the line to the target is crossed at a point closer to the goal than the hit point. At this point, the robot resumes the progress along the line to the goal. In general, this algorithm leads to shorter paths than Bug 1 (see figure 2.6). However, one can still face scenarios in which the path generated by Bug 2 is longer than that corresponding to Bug 1 [SN04].

Some Bug variants are classified as tangent-based (e.g. [KRR98], [LB99], [SH13]), in the sense that they consider motion towards the tangents of objects. These methods build a graph, called local tangent graph (LTG), of the robot's environment utilizing a ray-based sensor system. With the help of LTG, the robot can make a shortcut while following the contour of objects, switching back to the goal pursuit mode earlier. In this regard, a shorter path can be achieved.

Bug algorithms enable vehicles to navigate in unforeseen surroundings with guaranteed global convergence (if the goal is reachable). However, these methods are sensitive to the accuracy of the sensor and represent the robot by a point in the workspace. Furthermore, they were not examined in dynamic scenarios which is often the case in real-world applications.

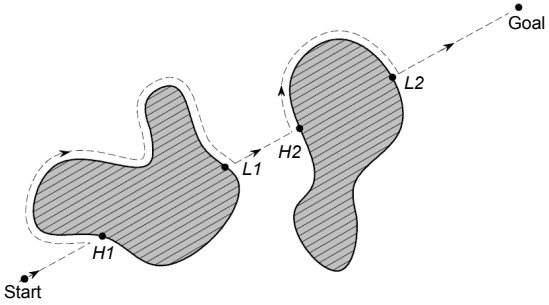


Figure 2.6: Bug 2 algorithm with (H_1, H_2) described as hit points, and (L_1, L_2) described as leave points (from [SN04]).

2.4.2 Artificial Potential Fields

A wide variety of reactive navigation methods (e.g. [MA97], [GC02], [FNA09], [RA12], [KST+16], [BGPG17], [MCL+17], [RBAM18]) are based on the concept of Artificial Potential Field (APF), firstly developed in [Kha86]. Under this concept, the robot is a particle influenced by a *gravitational force field*, where an attractive force is acted by the goal and a repulsive force is acted by obstacles (see figure 2.7a). The attraction towards the goal and the repulsion from obstacles are represented by positive and negative forces, respectively. The resultant vector sum of these forces determines the control input applied to the robot. More formally, let the attractive potential to the goal be described by the following quadratic function:

$$\mathrm{U}_{\mathrm{att}}(\mathbf{p}_r) = \frac{1}{2} k_a \|\mathbf{p}_r - \mathbf{p}_g\|^2 \tag{2.6}$$

where $k_a > 0$, and $\mathbf{p}_r$ and $\mathbf{p}_g$ are the robot and goal locations.

Analogously, for each obstacle, the associated repulsive potential is defined as:

$$\mathrm{U}^i_{\mathrm{rep}}(\mathbf{p}_r) = \begin{cases} \frac{1}{2} k_r \left(\frac{1}{r_i} - \frac{1}{r_s}\right)^2, & \text{if } r_i \leq r_s \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

where $k_r > 0$, r_i the distance from $\mathbf{p}_r$ to obstacle $\mathbf{p}_i$, and r_s the range of influence of the obstacles.

The value of the function representing the potential can be considered as an energy and therefore its negative gradient is a force. Hence, the attractive and repulsive forces resulting from $\mathrm{U}_{\mathrm{att}}(\mathbf{p}_r)$ and $\mathrm{U}^i_{\mathrm{rep}}(\mathbf{p}_r)$ can be defined as:

$$\mathrm{F}_{\mathrm{att}}(\mathbf{p}_r) = -k_a(\mathbf{p}_r - \mathbf{p}_g) \tag{2.8}$$

$$\mathrm{F}^i_{\mathrm{rep}}(\mathbf{p}_r) = \begin{cases} \frac{k_r}{r_i^2}(\frac{1}{r_i} - \frac{1}{r_s})\nabla r_i, & \text{if } r_i \leq r_s \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

Notice that the value of $\mathrm{F}_{\mathrm{att}}(\mathbf{p}_r)$ converges to zero when the location of the robot $\mathbf{p}_r$ approaches $\mathbf{p}_g$. The value of $\mathrm{F}^i_{\mathrm{rep}}(\mathbf{p}_r)$ is zero if the distance to the obstacle is greater than r_s and tends to infinity as the obstacle is approached. The resultant force field is then defined by:

$$\mathrm{F}(\mathbf{p}_r) = \mathrm{F}_{\mathrm{att}}(\mathbf{p}_r) + \sum_{i=1}^{n} \mathrm{F}^i_{\mathrm{rep}}(\mathbf{p}_r) \tag{2.10}$$

where n is the number of obstacles.

Potential field methods are efficient in terms of computational complexity and the time needed to complete a mission. Moreover, they are easy for implementation. However, with these techniques the robot is prone to oscillations in narrow spaces and may experience a deadlock due to local minima [KB91]. Figure 2.7b shows a situation where the robot experiences a local minimum while approaching a U-shaped obstacle, leading to a deadlock. This is due to the fact that, at a particular position within the obstacle, the attractive force gets symmetric to the repulsive force (i.e. $\mathrm{F}(\mathbf{p}_r) = 0$). There have been several attempts to address this drawback, such as [BL91] by utilizing a "random walk" mechanism, [RK92] [Mas12] by incorporating an extra APF function, or [HE02] by borrowing ideas from electro magnetics. This problem has also been addressed in [VTML00], by incorporating the artificial intelligence, and in [WIN15] by considering the robot's direction of motion and the front-face obstacle information. Other efforts were devoted towards achieving a smoother navigation, such as [RMP06], by utilizing an adaptation of Newton's method, or [Pan14] [HP16] by making use of two dimensional "smooth vector fields". Adapting the APF to accommodate robots of unicycle type and obstacles of arbitrary shape can

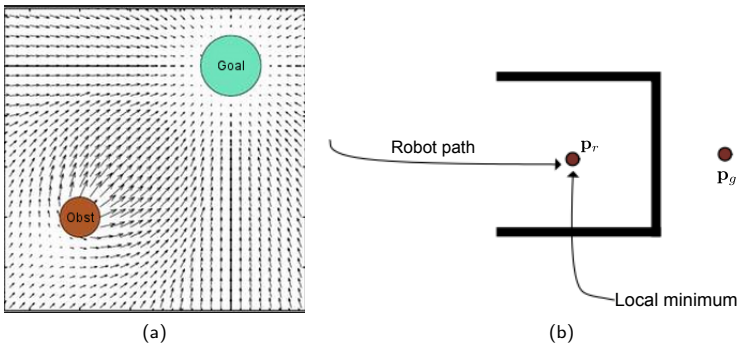


Figure 2.7: Artificial Potential Field. (a) Typical potential fields; an attractive force is acted by the goal and a repulsive force is acted by obstacles (from [Kum15]). (b) A robot experiences a local minimum while approaching a U-shaped obstacle. This problem occures if the attractive force gets symmetric to the repulsive force (originally from [Rib05]).

also be found in [RMP08], [VT12], and [WC00]. Despite the fact that these approaches may resolve the above mentioned APF problems, they are computationally inefficient, restricted to specific environments, or build upon strong assumptions as stated in [CML+15]. For example, the performance of the algorithm presented in [VTML00] is still limited by the basic model of the potential field function. Moreover, it only considers obstacles of equal sizes. The methods proposed in [WIN15], [Pan14], and [HP16] assume a point-like or a disc-shaped mobile robot and discard the motion constraints. Additionally, the techniques introduced in [BL91] and [RK92] often need long time to escape from the local minima in complex environments [CML+15].

2.4.3 Virtual Force Field

The Virtual Force Field (VFF) [BK89] is a collision avoidance method that consists of integrating two concepts; the Certainty Grid [Elf86] [ME85] for obstacle representation and the Artificial Potential Field for navigation. The certainty grid is a probabilistic representation of obstacles in a grid-like world model, es-

pecially designed to accommodate the inaccuracy of sensor data. Each cell in the grid contains a certainty value denoting the probability that an obstacle is located in this cell. The VFF method uses a similar analogy, but updating the world model, referred to as a *histogram grid*, is carried out in a different way; only one cell is required to be incremented for each new sensor reading creating a probability distribution with small computational overhead. As the robot moves, it keeps track of a window overlying a zone of the histogram grid, called *active window*, where a repulsive force acts between the robot and each cell in this zone (named *active cell*). Obviously, there is a direct proportional relationship between this repulsive force and the certainty value. Additionally, an inverse proportional relationship exists between the force and the distance to the cell. In a way similar to the APF method, the resultant force vector is then calculated.

As compared with the APF approach, the VFF method reduces the sensor uncertainties due to its probabilistic nature and provides more stable motion. Several methods build on the concept of VFF (e.g. [IO02] [LLLH08] [NTK+11]). However, neither the VFF method nor these variants are able to avoid the shortcomings inherited from the APF concept, summarized in [BL91].

2.4.4 Vector Field Histogram

The Vector Field Histogram (VFH) [BK91] constructs a 2D histogram grid using information from onboard sensors, similar to the VFF method. Then, the motion control is computed by employing two data reduction steps. In a first step, the active window of the grid is mapped into a 1D *polar histogram* surrounding the current location of the robot (see figure 2.8). This histogram comprises angular sectors, where each sector defines a polar direction relative to the center of the vehicle. In a second step, a steering direction is calculated by analyzing the polar histogram to determine open areas and then choosing the passage that optimizes a cost function. Setting this function is based on the robot's previous orientation, the target direction, and the wheels angle. The speed is then calculated as a function of the distance to obstacles.

An improved version of the VFH method [UB98], takes into account the robot's shape and kinematic limitations by enlarging obstacles so that all kinematically

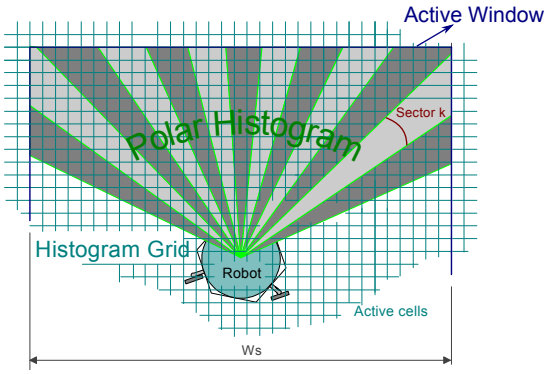


Figure 2.8: Mapping of active cells onto the polar histogram (originally from [BK91]).

blocked trajectories are avoided, an additional stage called *masked polar histogram*. The enhanced method, entitled VFH+, can generate smoother robot trajectories with greater reliability. This approach was further improved in [UB00] by introducing a lookahead procedure, thus allowing the robot to avoid the local minima problem. In addition to these two improvements, several variants have been developed during the last two decades (e.g. [AW04] [Ye07] [JXK10]). A comparison of these variants showing their pros and cons is carried out in [BDD$^+$14]. Although the VFH approach is less likely to fall in local minima and produces smoother behavior, it presents the difficulty to drive the robot in narrow passages or corridors. Moreover, it is limited by the use of arbitrary heuristics which greatly influence the behavior of the system.

2.4.5 Dynamic Window Approach

In the mid-1990s, many research efforts were devoted towards integrating the vehicle kinematics and dynamics into the collision avoidance problem, directly determining a motion control instead of a direction solution. The result of these efforts was the evolution of several techniques, later described as *velocity space approaches*. These include: the Steering Angle Field (SAF) [FBL94], Curvature Velocity Method (CVM) [Sim96], and its enhanced variant Lane-Curvature

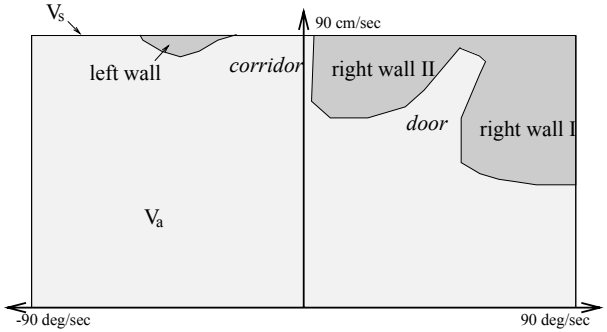


Figure 2.9: Velocity space (from [FBT97]).

Method (LCM) [KSRS98], just to name a few. However, it has been the Dynamic Window Approach (DWA) [SP07] [LV14] [LV16] [BUVRJ17], initially introduced by Fox [FBT97], that has proved popular within the robotics community. Assume a known current robot's state, the DWA models the reactive navigation problem as a constraint optimization in the *tangent space* (velocity space): the space of all possible sets of translational and rotational robot velocities (v, ω). Therefore, it can be characterized as a planning algorithm with a prediction horizon of a single time step [OL05]. The original DWA [FBT97] was designed for a synchro-drive robot, which can be summarized in two steps:

Search space: The DWA builds upon the assumption that the robot's velocity is kept constant at each time step (piecewise constant velocities). Under this assumption, the trajectory of a synchro-drive mobile robot can be approximated by a series of circular arcs, referred to as curvatures [FBT97]. By this means, the search space is reduced into a 2D velocity search space. The set of all possible velocities is denoted by V_s (see figure 2.9). The method only considers the *admissible velocities* which guarantee collision-free motion (denoted as V_a). In figure 2.9, the velocities that may cause collision with obstacles (non-admissible) are visualized by dark gray. For instance, velocities in the region labeled as "right wall II" result in a hard right turn and hence may cause collision with the wall.

Due to the limited accelerations, the search space is further restricted to a small window, referred to as a *dynamic window* (labeled V_d in figure 2.10). It includes

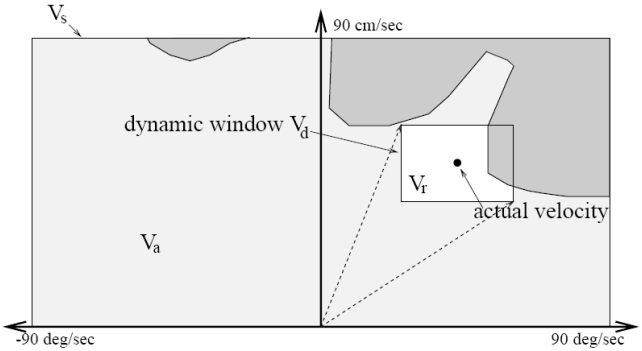


Figure 2.10: Dynamic window (from [FBT97]).

those velocities that can be reached within the acceleration limits over the next control cycle. The above mentioned constraints limit the search space into a region denoted by V_r (the white area) within the dynamic window:

$$V_r = V_s \cap V_a \cap V_d \tag{2.11}$$

Optimization: After having determined the resultant search space V_r, a motion control is determined by optimizing an objective function [FBT97]:

$$G(v,\omega) = \sigma(\alpha \cdot \text{heading}(v,\omega) + \beta \cdot \text{distance}(v,\omega) + \gamma \cdot \text{velocity}(v,\omega)) \tag{2.12}$$

This function provides a weighted sum of heading(v,ω), that favors progress towards the goal, distance(v,ω), that prefers to keep large distances to obstacles, and velocity(v,ω), which favors high speed.

The dynamic window approach has been extended to other robot shapes and models (e.g. [Sch98]). Moreover, the employment of a "navigation function" has led to optimal paths, see e.g. [BK99], [OL05], [KT12], and [MSKT13]. Some variants use information about the predicted obstacles' trajectories to better handle dynamic environments (e.g [SMP05], [SP07]). Other variants consider different trajectory shapes, where the differences in performance are mainly heuristic [FSBD04] [SWY10]. By utilizing the DWA, the efficiency and smoothness of

the robot's trajectories are enhanced. However, a failure may occur while driving a robot through narrow regions in dense environments. Moreover, the DWA is limited by the model construction as well as the parameters tuning [Pet08].

2.4.6 Velocity Obstacles

The concept of Velocity Obstacles (VO), initially proposed in [FS98], has been widely used by robotic researchers, see e.g. [SS07] [GSR09] [WH12] [KO16] [LJO17] [RGM17] [BSAP18]. Within this concept, the velocity of moving obstacles can be explicitly considered in determining the avoidance maneuver. A VO is essentially the set of velocity controls leading to collision with obstacles at a later time, once specified, a velocity beyond this set is picked out. A concept similar in spirits to VO, the Inevitable Collision States (ICS), was introduced in [FA04] and [LNWB14]. Compared to VO, ICS reasons over an infinite time horizon and discards the colliding states rather that the velocity controls.

More formally, let v_r be a given robot velocity and v_i the velocity of obstacle $\mathcal{O}_i$. The set of colliding relative velocities between the robot and obstacle $\mathcal{O}_i$, referred to as *collision cone*, is defined as follows:

$$\mathcal{CC}_i = \{v_{r,i} \mid \lambda_{r,i} \cap \mathcal{CO}_i \neq 0\} \tag{2.13}$$

where $v_{r,i} = v_r - v_i$, $\lambda_{r,i}$ is the line of $v_{r,i}$, and $\mathcal{CO}_\mathrm{i}$ is the result of mapping $\mathcal{O}_\mathrm{i}$ into the configuration space (enlarging $\mathcal{O}_i$ by the radius of the robot).

The *velocity obstacle* associated with obstacle $\mathcal{O}_i$ is obtained by adding its velocity to each relative velocity in $\mathcal{CC}_i$ (to get absolute robot velocities):

$$\mathcal{VO}_i = \mathcal{CC}_i \oplus v_i \tag{2.14}$$

where $\oplus$ represents the Minkowski vector sum.

The union of all velocity obstacles (associated with each $\mathcal{O}_i$) defines the set of forbidden robot velocities (see figure 2.11):

$$\mathcal{VO} = \bigcup_{i=1}^{n} \mathcal{VO}_i \tag{2.15}$$

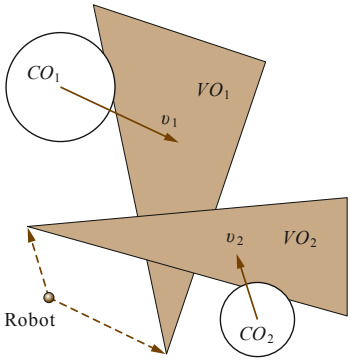


Figure 2.11: The set of forbidden robot velocities ($v_1 \cup v_2$). In order to guarantee a safe motion, a control velocity must be chosen outside of this set. (originally from [MLL08]).

where n is the number of obstacles.

The computation of the VO builds upon the assumption that the velocity of obstacles is maintained at future time, which is not realistic in real-world scenarios. In [SLS01] and [LLS05], this drawback has been addressed proposing a new concept, Non Linear Velocity Obstacles (NLVO), to deal with obstacles whose trajectory is nonlinear or arbitrary (but known). However, it has been shown that computing NLVO is computationally expensive [Wan14]. Some variants take into account the non-holonmic constraints of the obstacles (e.g. [OM05] [OM06]), but assuming that the path can be approximated by a series of short line segments. Moreover, the shape of obstacles are assumed to be either circles or polygons. The concept of velocity obstacles has been extended to multiple robot navigation and called Reciprocal Velocity Obstacles [BLM08] [BB15]. This has been possible by implicitly assuming that the other robots make a similar obstacle avoidance reasoning. Although VO-based approaches explicitly incorporate the velocity of obstacles into the motion control, they require a perfect understanding of the scene including an exact estimation of the obstacles motion. However, in real-world scenarios, it is difficult to estimate the future of the scene [JKWG15]. Moreover, it is of great importance in VO-based techniques to choose an appropriate time horizon which is often hard in cluttered and dense environments [SS12].

2.4.7 Nearness-Diagram Navigation

It is a challenging problem to safely drive an autonomous robot in cluttered, dense, and complex environments, which is usually the case in most robotic applications [Min02]. Good results in such environments have been reported using the Nearness-Diagram Navigation (ND) approach [MM04], that might be described as a methodology of designing collision avoidance algorithms rather than a method in itself [MLL08]. It is similar to the earlier developed Vector Field Histogram method presented in section 2.4.4 but solves many of its shortcomings, especially in narrow spaces. The key idea of this approach is to utilize a *divide and conquer* methodology performing a high-level information description of the environment, following the behavior-based *situated activity* paradigm [Ark98]. A predefined set of conditions is used to characterize all potential navigational states and their associated motion control. At each time step, the current situation is determined from onboard sensors and its associated action is executed. The original ND method [MM04] divides the navigation behavior into five situations. Afterwards, the motion laws were reformulated by adding another situation, leading to the ND+ method [MOM04]. In the following, we briefly discuss these situations and their corresponding actions:

Situations: The set of situations is represented using a binary decision tree whose inputs are the robot and goal locations and the set of obstacle configurations. The criteria for determining these situations is based on high-level navigation entities like a motion region, environmental characteristics, and a safety level between the robot and the obstacles (security zone). For example, one criterion is whether the security zone is free from obstacles or not. Another is whether the goal is located within the motion region or not. The result is only one situation owing to the fact that the set of situations must be complete and exclusive (it is represented by a binary decision tree), see figure 2.12.

Actions: For each situation, an associated action is executed that computes a suitable motion law. Generally speaking, an action describes the behavior required for each situation [MLL08]. For instance, one of the situations is when the goal is located within the motion region and the security zone is free from obstacles (HSGR). The action in this case is to drive the robot towards the

target. A second situation is when the security zone is free from obstacles, and the motion region is wide enough and does not include the target (HSWR). The action in this case is to drive the robot towards the side of the motion region while keeping a safe area to obstacles (see figure 2.12).

A global reasoning based on a workspace representation was added to the approach in [MMSA01]. Moreover, the performance of ND+ in wide areas has been improved in [Min05]. Another ND variant, the Smooth Nearness-Diagram (SND) [DB08], proposes a single motion control regardless of the distribution of obstacles. By this means, the resultant trajectory depends on the configuration of all obstacle points, and thus a smoother behavior is achieved. However, SND may fail to guide a robot through a tight opening if the density of obstacles on one side is much higher than the other. The Closest Gap (CG) [MFMJ10] Navigation [DB08] was then developed addressing this drawback by respecting the percentage of threats[1] on each side of the desired heading and by applying a higher avoidance against an obstacle as it gets closer to the robot's boundary. Furthermore, a novel methodology for characterizing the environmental structure was proposed, enhancing smoothness and reducing computational overhead. An approach similar in spirits to the ND-based methods is introduced in [SG12] [DS17] [OS17], where the field of view constraint is also considered.

The Nearness-Diagram Navigation overcomes several drawbacks of the collision avoidance approaches [MM04], such as experiencing a local minima (e.g. U-shaped obstacle), difficulty of driving a robot towards obstacles (when necessary), being prone to oscillations, and the tedious parameter tuning (see [MM04] for more details). Experiments demonstrated that navigation in environments cluttered with obstacles has been successfully achieved using the ND-based methods [Fra07]. However, they are prone to oscillations and instability, which in turn may reduce the speed and can be unsafe in narrow spaces. Furthermore, ND-based methods are likely to cause deviations towards free regions, increasing the path length and execution time. Another drawback is the assumption that the robot is circular and holonomic. In order to achieve a safer behavior, it is necessary to consider both the shape and kinematic constraints. Addressing all these drawbacks is the motivation of the research work presented in this dissertation.

[1] A threat is an obstacle point that falls within a predefined security zone around the robot.

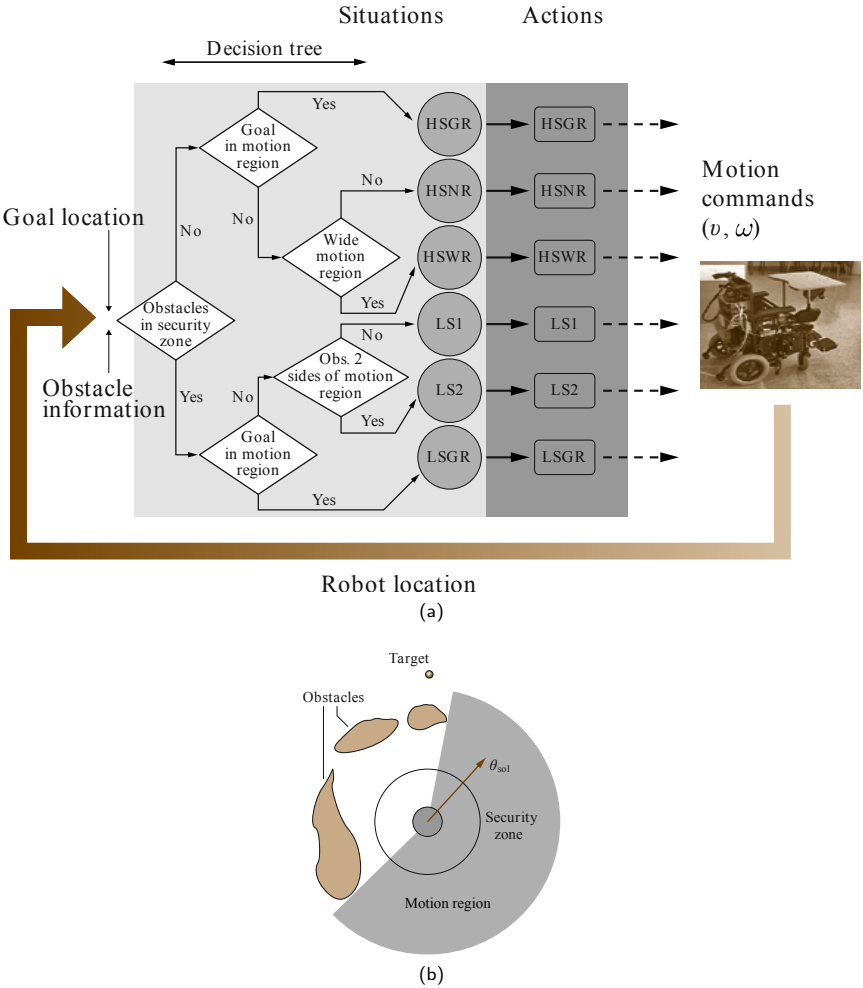


Figure 2.12: Overview of the Nearness-Diagram (ND) navigation approach (originally from [MLL08]). (a) Diagram showing the design of the ND method [MM04] following the *situated-activity* paradigm. Based on the sensor readings and the locations of the robot and the goal, one situation is chosen and the associated motion law is computed. (b) An example shows how to determine the motion direction. First, the situation is identified: the security zone is obstacle-free, the motion region is wide, and the target does not fall within the motion region. In this case, the situation is HSWR. Second, the suitable action is executed computing the most promising motion direction θ_{sol}.

3 A New Gap-based Collision Avoidance Approach - A Holonomic Solution

This chapter introduces a new obstacle avoidance approach for mobile robots navigating in dense environments. As we have pointed out in chapter 2, the majority of reactive navigation approaches present limited performance in cluttered environments. This is owing to the fact that these techniques are prone to some classical drawbacks [MM04] [MLL16] [MFM16] such as computational complexity, experiencing a local minima, difficulties of driving a robot towards obstacles (when necessary), failure of navigating a robot through narrow spaces, and the tedious parameter tuning. Driving robots in such scenarios while avoiding these drawbacks has been successfully achieved by the Nearness Diagram (ND) Navigation [MM04]. During the last decades, many ND variants have been proposed like the Smooth Nearness-Diagram [DB08], Obstacle-Restriction [Min05], and Closest Gap [MFMJ10] navigation methods. The main differences in behavior between these variants have been discussed in chapter 2.

In general, all ND variants assume static obstacles and work as follows. The data coming from sensors is analyzed to find potential free openings (referred to as gaps) surrounding the robot, once determined, the closest to the goal is selected for navigation. The robot is then driven towards the selected opening unless the clearance to obstacles gets less than a security zone (safe distance), in which case, the robot's trajectory is adjusted building upon the Artificial Potential Field (APF) concept [Kha86]. Avoiding obstacles based on APF is prone to oscillations and instability, which in turn may reduce the speed of the robot and can be unsafe in narrow spaces [MFM16]. Moreover, defining the boundaries of the security zone is not straightforward and has a big effect on the performance of the robot [MFM15]. Furthermore, adjusting the robot's trajectory only depends

on the distance to nearby obstacles, neglecting the location and field of view of the selected gap (opening angle). This may cause deviations towards free regions, increasing the total time and distance needed to perform a given task.

This chapter introduces a new reactive obstacle avoidance approach that addresses the aforementioned problems. A key idea of the proposed approach, entitled "Safe Gap (SG) Navigation", is to further analyze the environmental structure and virtually create a gap between the current robot configuration and the selected opening. This gap, referred to as a "safe gap", is determined in such a way that its opening angle (as seen by the sensors) is wide enough, maximizing the clearance to obstacles. Consequently, the safety and smoothness of the robot's motion are enhanced compared to the ND variants. Moreover, unreasonable deviations towards free regions are avoided, reducing the total time and distance needed to complete the mission. Unlike the ND-based obstacle avoidance techniques, the SG method does not require the safe distance parameter, and thus saves the parameter tuning overhead [MFM13b] [MFM13a].

This chapter introduces the SG obstacle avoidance method design in section 3.1. In sections 3.2 and 3.3, the simulation as well as the experimental results are shown. Finally, section 3.4 draws the conclusions from this study.

3.1 The Reactive Navigation Strategy

This section presents the "Safe Gap" (SG) reactive collision avoidance approach for autonomous robots navigating in dense and cluttered environments. The SG method acts as a reactive layer in a navigation system, following a "perception-action" procedure executed at a high frequency rate. The key idea is described in the following. At each time step, the sensor information is checked to find out if there is a safe way towards the goal. Otherwise, the robot will be directed to another location, referred to as a *subgoal*, as discussed in section 3.1.2. The location of the subgoal is defined depending on analyzing the environmental structure. The main aspect in this analysis is to determine the set of surrounding *gaps*. A description of our methodology for extracting gaps is introduced in section 3.1.3, and subsequently, an illustration of how subgoals are located within

free areas is presented in section 3.1.4. In section 3.1.5, it is shown how to set the motion commands that steer a robot towards the goal (resp. subgoal). This process is repeated at each sensor update.

3.1.1 Preliminary Definitions and Notations

This section presents several definitions and notations that are utilized to clarify the proposed method. Some of these definitions have been used in our publications [MFM13b] [MFM15] [MFM16] [MM16] [MFM17] [MM17] [MFM18].

The locations of the goal and the robot are denoted by $\mathbf{p}_g$ and $\mathbf{p}_r$, respectively. The radius of the robot is denoted by R and it represents the radius of a disc virtually wraps around the entire robot.

The local x axis is aligned along the longitudinal direction and the local y axis is perpendicular to it, with $+x$ points forward and $+y$ points to the left side.

Angles relative to the robot coordinate system can be positive (towards the left side) or negative (towards the right side) with a maximum absolute value of π.

Performing calculations may result in an angle θ whose absolute value exceeds π. This angle can be mapped into $[-\pi, \pi[$ using the following function:

$$\text{proj}(\theta) = ((\theta + \pi) \bmod 2\pi) - \pi \tag{3.1}$$

Sometimes, it is necessary to restrict a quantity within a certain range, say from a to b where $a < b$. For this purpose, the following function is employed:

$$\text{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x \leq a \\ x, & \text{if } a < x < b \\ b, & \text{if } x \geq b \end{cases} \tag{3.2}$$

Let $\mathbb{S}^1$ be a unit circle whose center coincides with the robot's origin, and ϕ_a and ϕ_b be two angles in $\mathbb{S}^1$, the minimum angular difference between them is:

$$\angle(\phi_a, \phi_b) = \min(\angle(\phi_a \rightarrow \phi_b), \angle(\phi_b \rightarrow \phi_a)) \tag{3.3}$$

where

$$\angle(\phi_a \to \phi_b) = (\phi_a - \phi_b) \bmod 2\pi \tag{3.4}$$

For generality, it is assumed that the sensory information is given as scan (depth) points; it is possible to transform the data returned by the majority of sensors to points. The depth points list is represented by $S = \left\{\mathbf{p}_1^{\mathrm{S}}, ..., \mathbf{p}_n^{\mathrm{S}}\right\}$. It is sorted counterclockwise relative to the sensor frame. The Cartesian and polar coordinates of a point $\mathbf{p}_i^{\mathrm{S}}$ are denoted by $(x_i^{\mathrm{S}}, y_i^{\mathrm{S}})$ and $(r_i^{\mathrm{S}}, \theta_i^{\mathrm{S}})$, respectively. Additionally, $r_{\max}$ denotes the maximum range of the sensor.

A *gap* is a potentially open area between obstacles through which the robot can fit. As will be explained in section 3.1.3, each gap g is created by two obstacles. The angle towards one of them is less than the other. The obstacle with the bigger angle creates the *left side* of the gap. Obviously, the other obstacle creates the *right side*. Similarly, the distance to one obstacle is less than the other. This obstacle forms the *closer side* of the gap while the other forms the *farther side*.

The locations of the left side, right side, closer side, and farther side of a particular gap g are denoted by $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, $\mathbf{p}_{\mathrm{cr}}(\mathrm{g})$, and $\mathbf{p}_{\mathrm{fr}}(\mathrm{g})$, respectively. Similarly, referring to the polar coordinates of a specific side requires to replace $\mathbf{p}$ by r (resp. θ), e.g. $r_{\mathrm{fr}}(\mathrm{g})$ represents the distance to the farther side of g.

The width of a particular gap g is represented by $w(\mathrm{g})$ and equals to the Euclidean distance between its both sides: $w(\mathrm{g}) = \|\mathbf{p}_{\mathrm{l}}(\mathrm{g}) - \mathbf{p}_{\mathrm{r}}(\mathrm{g})\|$.

The robot moves towards a gap g in such a way that a proper distance $\mathrm{d_s}(\mathrm{g})$ is preserved between $\mathbf{p}_{\mathrm{cr}}(\mathrm{g})$ and the robot's footprint. The value of $\mathrm{d_s}(\mathrm{g})$ is determined based on the Euclidean distance between both gap sides (gap width); for a short distance, $\mathrm{d_s}(\mathrm{g})$ is set to half of the gap width, but for a large distance, $\mathrm{d_s}(\mathrm{g})$ is restricted to $R + \mathrm{d_{safe}}$, where $\mathrm{d_{safe}} > 0$ is a desired clearance to obstacles:

$$\mathrm{d_s}(\mathrm{g}) = \begin{cases} R + \mathrm{d_{safe}}, & \text{if } w(\mathrm{g}) > 2(R + \mathrm{d_{safe}}) \\ \frac{1}{2} w(\mathrm{g}), & \text{otherwise} \end{cases} \tag{3.5}$$

Notice that $\mathrm{d_{safe}}$ provides a trade-off between efficiency and safety. A value of $2R$ was used in our experiments. In principle, $\mathrm{d_s}(\mathrm{g})$ can be set to $\frac{1}{2} w(\mathrm{g})$ always, but this may lead to a longer path if the gap is too wide.

In order to drive a robot through a gap g, an instantaneous subgoal, denoted by $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$, is assigned and the robot is directed towards it (in section 3.1.5, it is described how to define the exact location of a subgoal). The polar coordinates of $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$ are denoted by $(r_{\mathrm{s}}(\mathrm{g}), \theta_{\mathrm{s}}(\mathrm{g}))$.

For a better visibility, superscripts are eliminated from figures ($\mathbf{p}_i^{\mathrm{S}}$ becomes $\mathbf{p}_i$).

Finally, the Euclidean distance between two points, $\mathbf{p}_i^{\mathrm{S}}$ and $\mathbf{p}_j^{\mathrm{S}}$, in the workspace is denoted by $\mathrm{d}(\mathbf{p}_i^{\mathrm{S}}, \mathbf{p}_j^{\mathrm{S}})$.

3.1.2 Selecting the Direction of Motion

If it is unsafe to directly head the robot from its current location $\mathbf{p}_r$ towards the goal $\mathbf{p}_g$, it is essential to locate a subgoal as an intermediate step and drive the robot towards it. Motion towards the goal is resumed once the "direct path" from $\mathbf{p}_r$ to $\mathbf{p}_g$ is collision free. In [MFM13b], we have defined a "direct path" towards a given position $\mathbf{p}_x$ as the path followed by a robot to attain $\mathbf{p}_x$ by executing a single motion control . Determining whether this path is collision free or not depends on the obstacle distribution, the vehicle constraints, and the robot's footprint. This section presents an algorithm (previously published in [MFM13b]) to check this condition for a free-flying (holonomic) mobile robot.

The algorithm extracts the set of obstacle points $\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_x]$ that may cause collision with the robot while driving it along the direct path towards $\mathbf{p}_x$. Obviously, a holonomic mobile robot follows a path made up of straight segments to reach $\mathbf{p}_x$. Assume that the robot's footprint can be represented by a polygon whose edges are denoted by E_i, where $i = 1, ..., m$. Also, let $\overrightarrow{\mathbf{p}_r\mathbf{p}_x}$ be the vector that connects $\mathbf{p}_r$ to $\mathbf{p}_x$, called "base vector" in [MFM13b]. For each $\mathbf{p}_i^{\mathrm{S}} \in \mathrm{S}$, it is checked if any of the robot edges intersects the line that passes through $\mathbf{p}_i^{\mathrm{S}}$ and parallel to the base vector. If no intersection exists, $\mathbf{p}_i^{\mathrm{S}}$ is collision-free, and hence it is discarded. Otherwise, we do the following: assume that $\mathbf{p}_e$ represents the point in the robot's boundary, at which the intersection occurs. Whenever point $\mathbf{p}_x$ is reached, the coordinates of $\mathbf{p}_e$ with respect to the current robot coordinate system can be expressed as $\mathbf{p}'_e = \mathbf{p}_e + \overrightarrow{\mathbf{p}_r\mathbf{p}_x}$. If $\mathbf{p}_i^{\mathrm{S}}$ is located on the line segment $\overline{\mathbf{p}_e\mathbf{p}'_e}$, it causes collision, and hence it is added to $\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_x]$.

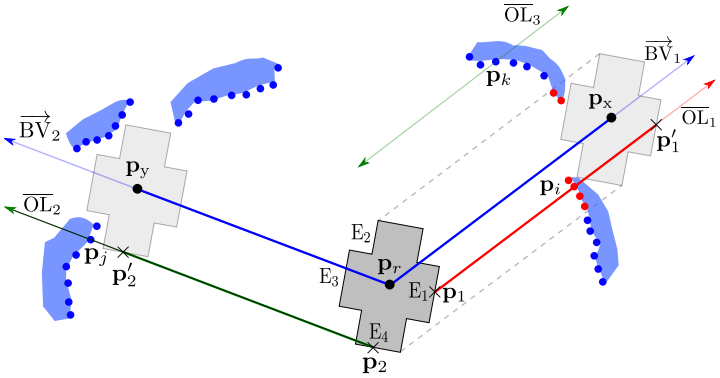


Figure 3.1: Collision check along the "direct path" towards two locations, $\mathbf{p}_x$ and $\mathbf{p}_y$. Line $\overline{\mathrm{OL}}_1$ that passes through $\mathbf{p}_i$ intersects E_1 in $\mathbf{p}_1$. Obstacle $\mathbf{p}_i$ causes collision with the direct path towards $\mathbf{p}_x$ since it is located on the dark red line segment $\overline{\mathbf{p}_1\mathbf{p}'_1}$. Line $\overline{\mathrm{OL}}_2$ that goes through $\mathbf{p}_j$ hits E_4 in $\mathbf{p}_2$. Obstacle $\mathbf{p}_j$ is collision-free while traveling towards $\mathbf{p}_y$ since it is not located on $\overline{\mathbf{p}_2\mathbf{p}'_2}$. None of the robot edges intersects $\overline{\mathrm{OL}}_3$, so $\mathbf{p}_k$ is collision-free while traveling towards $\mathbf{p}_x$. The path towards $\mathbf{p}_y$ is free, while the path towards $\mathbf{p}_x$ is in collision with the red obstacle points (adapted from [MFM13b] with permission from IEEE).

Otherwise, $\mathbf{p}_i^{\mathrm{S}}$ is collision-free, thus discarded. Apparently, the collision check can be reduced to one step if the robot's boundary can be represented by one equation (e.g. an eclipse) rather than approximating it by a polygonal shape.

Figure 3.1 shows an example where the task is to extract the set of obstacle points that may cause collision with the "direct path" towards two locations $\mathbf{p}_x$ and $\mathbf{p}_y$. The robot has a polygonal shape and consists of 12 edges. It is apparent that line $\overline{\mathrm{OL}}_1$, which is parallel to $\overrightarrow{\mathrm{BV}}_1$ (the base vector) and passes through $\mathbf{p}_i$, intersects edge E_1 in $\mathbf{p}_1$. Also, line $\overline{\mathrm{OL}}_2$, which goes through $\mathbf{p}_j$ and parallel to $\overrightarrow{\mathrm{BV}}_2$ hits edge E_4 in $\mathbf{p}_2$. Obstacle $\mathbf{p}_i$ is located on $\overline{\mathbf{p}_1\mathbf{p}'_1}$, while $\mathbf{p}_j$ is not located on $\overline{\mathbf{p}_2\mathbf{p}'_2}$. Therefore, the direct path towards $\mathbf{p}_x$ is in collision with $\mathbf{p}_i$, but the direct path towards $\mathbf{p}_y$ is collision-free with $\mathbf{p}_j$. Apparently, obstacle $\mathbf{p}_k$ is not in collision with the direct path towards $\mathbf{p}_x$, since none of the robot edges intersects $\overline{\mathrm{OL}}_3 \parallel \overrightarrow{\mathrm{BV}}_1$. In summary, the path towards $\mathbf{p}_y$ is collision-free, while the path towards $\mathbf{p}_x$ is in collision with the 6 obstacle points visualized by red.

If there is a direct free path towards $\mathbf{p}_g$ (i.e. $\mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_g] = \phi$), the robot is directly driven towards it. Otherwise, a subgoal is located in a collision free area and the robot is directed towards it. In the next sections, it is shown how to identify the position of this subgoal.

3.1.3 Extracting Gaps

As mentioned previously, the main aspect in analyzing the environmental structure is the determination of the set of surrounding gaps. For this purpose, our strategy developed in [MFMJ10] and utilized in [MFM16] is followed. This strategy is performed in two steps: in a first step, all gaps that can be seen from the current robot's view are found out. In a second step, useless or unnecessary gaps are identified and discarded. The major part in extracting gaps is the detection of spatial discontinuities in the sensor data, which can be of two types:

"Edge discontinuity": takes place if there is a spatial distance between two adjacent scan measurements more than the diameter of the robot, i.e.

$$\mathrm{d}(\mathbf{p}_r, \mathbf{p}_j^{\mathrm{S}}) - \mathrm{d}(\mathbf{p}_r, \mathbf{p}_i^{\mathrm{S}}) > 2R \tag{3.6}$$

"Max-range discontinuity": takes place if one of two adjacent scan measurements returns the maximum range of the sensor, i.e.

$$\mathrm{d}(\mathbf{p}_r, \mathbf{p}_j^{\mathrm{S}}) = r_{\max} \wedge \mathrm{d}(\mathbf{p}_r, \mathbf{p}_i^{\mathrm{S}}) < r_{\max} \tag{3.7}$$

If $r_j^{\mathrm{S}} > r_i^{\mathrm{S}}$ a "rising discontinuity" takes place at i; else, a "descending discontinuity" occurs at j. The condition of an "edge discontinuity" is checked before that of a "max-range discontinuity" (has a higher priority). In the following, these definitions are utilized to explain our strategy.

Step 1: Seeking for gaps: in a first step, a "forward search" from the first (1) to the final (n) scan measurements is performed. In a second step, a "backward search" is carried out in the reverse order (from n to 1). In the forward search, the depth measurement at which a rising discontinuity is detected creates the "first side" of a gap (for instance, at points A and D in figure 3.2). Assume that i

is the index of the measurement creating the first side. Determining the "second side" is based on the type of the discontinuity:

1) For a discontinuity of type edge: Let $S^+ = \{\mathbf{p}^S_{i+1}, ..., \mathbf{p}^S_n\}$ be the list of scan points coming after $\mathbf{p}^S_i$. The second side is created by the obstacle point (assume $\mathbf{p}^S_j$) closest to $\mathbf{p}^S_i$ and contained in S^+ such that the angular difference between both sides is less than π (as an example, see point B in figure 3.2).

$$d(\mathbf{p}^S_i, \mathbf{p}^S_j) \leq d(\mathbf{p}^S_i, \mathbf{p}^S_k) \wedge \angle(\theta^S_j \rightarrow \theta^S_i) \leq \pi, \quad \forall\, \mathbf{p}^S_k \in S^+ \tag{3.8}$$

2) For a discontinuity of type max-range: The second side is created by the scan measurement (assume j) forming the initial descending discontinuity that comes after the first side. e.g. point I in figure 3.2.

Seeking for the remaining gaps is performed starting from index j. The forward search produces gaps 1 - 4 in figure 3.2.

In the backward search, the second side of a gap occurs at the depth measurement creating a descending discontinuity (e.g. points F and I in figure 3.2). Let j be the index of this depth measurement, the first side is determined as follows:

1) For an edge discontinuity: Let $S^- = \{\mathbf{p}^S_{j-1}, ..., \mathbf{p}^S_1\}$ be the list of scan points coming before point $\mathbf{p}^S_j$. The first side is created by the obstacle point (assume $\mathbf{p}^S_i$) falling in S^- and closest to $\mathbf{p}^S_j$ such that the angular difference between both sides is less than π (for example, see point G in figure 3.2).

$$d(\mathbf{p}^S_i, \mathbf{p}^S_j) \leq d(\mathbf{p}^S_k, \mathbf{p}^S_j) \wedge \angle(\theta^S_j \rightarrow \theta^S_i) \leq \pi, \quad \forall\, \mathbf{p}^S_k \in S^- \tag{3.9}$$

2) For a max-range discontinuity: The first side is created by the scan measurement (assume i) forming the initial rising discontinuity that comes before the second side. See point H in figure 3.2.

Seeking for the remaining gaps is performed starting from index i. The backward search generates gaps 5 - 8 in figure 3.2.

Step 2: The result of performing both searches (forward and backward) is the set of all surrounding gaps, denoted as G. Among the assembled gaps, we remove

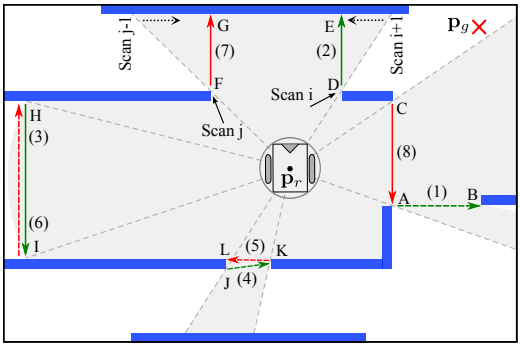


Figure 3.2: Finding out gaps. Firstly, the gaps marked as 1 - 4 and visualized by green arrows are found out by the "forward search" and the gaps marked as 5 - 8 and visualized by red arrows are extracted by the "backward search". Secondly, gaps 1, 4, and 6 are discarded as they are located within gaps 8, 5, and 3, respectively. Gap 5 is also discarded since its width is less than $2R$. Then, the closest gap (gap 8) is selected for navigation (adapted from [MFMJ10] with permission from IEEE).

those falling within other gaps and denote the set of remaining ones as G' (e.g. gaps 1, 4, and 6 in figure 3.2 are removed). Let $\mathbf{p}_f(\mathrm{g})$ represents the first side and $\mathbf{p}_s(\mathrm{g})$ be the second side of a gap $\mathrm{g} \in \mathrm{G}$, G' is then determined as follows:

$$\mathrm{G}' = \mathrm{G} \setminus \mathrm{A},\ \mathrm{A} = \{x \mid x \in \mathrm{G}, y \in \mathrm{G}, x \neq y, \theta_f(x) \geq \theta_f(y), \theta_s(x) \leq \theta_s(y)\} \quad (3.10)$$

where $\theta_f(\mathrm{g})$ and $\theta_s(\mathrm{g})$ are the angles towards $\mathbf{p}_f(\mathrm{g})$ and $\mathbf{p}_s(\mathrm{g})$, respectively.

The final step is to discard each gap having a width (Euclidean distance between both sides) less than the diameter of the robot (e.g. gap 5 in figure 3.2). We denote the list of remaining gaps by G'':

$$\mathrm{G}'' = \mathrm{G}' \setminus \mathrm{B},\ \mathrm{B} = \left\{\mathrm{g} \;\middle|\; \mathrm{g} \in \mathrm{G}',\ w(\mathrm{g}) < 2R\right\} \quad (3.11)$$

3.1.4 Locating the Subgoal

Once the set of gaps seen from the current robot's view (G'') is determined, the closest to the goal is picked out. This gap, called "closest gap", must satisfy two

conditions [MFM16]: first, the angular difference between one of its sides and the goal must be the minimum compared to that of the other gaps. Second, it has to be navigable [1]. All gaps are checked for both conditions until the closest gap is detected, or it is decided that no closest gap exists. The closest gap in figure 3.2 is labeled 8, where the side closest to the goal takes place at point C.

Then, a another gap is virtually created between the closest gap and the robot, referred to as a "safe gap". As pointed out in section 3.1.1, a subgoal is assigned to each gap and the robot is directed towards it. The safe gap is created in such away that the direct path towards its corresponding subgoal is collision-free. In the following, we show how to locate the safe gap based on investigating the environmental structure between the closest gap and the robot's location.

Before presenting details, it is useful to classify each gap based on two criteria: its position with respect to the current robot's location and its angular width with respect to the current robot's view [MFM13b].

"Gap location state": This criterion is concerned with specifying the location of gaps, relative to the current robot's position: if the angular distance between both sides of a gap g exceeds π, it is called a "rear gap". Otherwise, it will be a "front gap". Notice that the angular distance is measured here counterclockwise, i.e. from the right to the left of g. In figure 3.3, gaps 1 - 3 are front gaps, while gap 4 is a rear gap. The following function distinguishes front from rear gaps:

$$\Gamma(\mathrm{g}) = \begin{cases} 1, & \text{if } \theta_{\mathrm{l}}(\mathrm{g}) - \theta_{\mathrm{r}}(\mathrm{g}) \leq \pi \\ -1, & \text{otherwise} \end{cases} \tag{3.12}$$

"Gap vision state": With this criterion, the angular width of gaps with respect to the current robot's view is measured, reflecting the safety of navigation. If a given gap g is wide enough, it has a "good vision state". Otherwise, it will have a "weak vision state". The vision state of g is determined as follows: let $\overline{GL}$ be the line segment that connects both sides of g, and $\overleftrightarrow{L}$ the line that passes through the robot's origin and orthogonal to $\overline{GL}$. If $\overleftrightarrow{L}$ goes through $\overline{GL}$, the distance

[1] To determine the navigability status of a given gap, we do the following: if the goal is located in between both sides of the gap, we investigate whether there is a path towards the goal or not, following [MM04]. Otherwise, the path to be checked is towards the gap center, i.e. towards the point between its both sides.

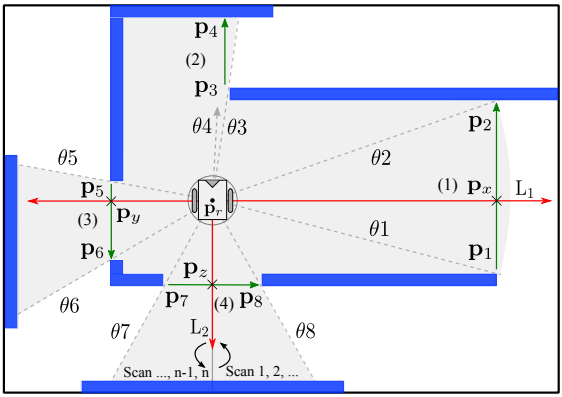


Figure 3.3: Classifying gaps based on their location and angular width. Gaps 1-3 are front gaps, whereas gap 4 is a rear gap. Gaps 1 and 4 are in a good vision state, while gaps 2 and 3 are in a weak vision state (adapted from [MFM13b] with permission from IEEE).

between $\mathbf{p}_{\mathrm{cr}}(\mathrm{g})$ and the intersection point is checked. If it is more than $\mathrm{d_s}(\mathrm{g})$, g has a good vision state. But, if it is less than $\mathrm{d_s}(\mathrm{g})$ or if $\overleftrightarrow{L}$ does not intersect $\overline{GL}$, g will have a weak vision state. The condition to be checked is defined as:

$$(\overleftrightarrow{L} \cap \overline{GL} \neq \phi) \wedge (\|\mathbf{p}_x - \mathbf{p}_{\mathrm{cr}}(\mathrm{g})\| \geq \mathrm{d_s}(\mathrm{g})) \tag{3.13}$$

where $\mathbf{p}_x$ is the point at which $\overleftrightarrow{L}$ intersects $\overline{GL}$ and ϵ is any small value. g has a good vision state if Eq. (3.13) is met. Else, it will have a weak vision state.

As an example, look at figure 3.3 which shows an environmental structure consists of 4 gaps, labeled 1 - 4. The line segments that connect both sides of these gaps are labeled $\overline{\mathbf{p}_1\mathbf{p}_2}$, $\overline{\mathbf{p}_3\mathbf{p}_4}$, $\overline{\mathbf{p}_5\mathbf{p}_6}$, and $\overline{\mathbf{p}_7\mathbf{p}_8}$. Line $\overleftrightarrow{\mathrm{L}_1}$ is orthogonal to $\overline{\mathbf{p}_1\mathbf{p}_2}$, $\overline{\mathbf{p}_3\mathbf{p}_4}$, and $\overline{\mathbf{p}_5\mathbf{p}_6}$, whereas line $\overleftrightarrow{\mathrm{L}_2}$ is orthogonal to $\overline{\mathbf{p}_7\mathbf{p}_8}$. The points of intersection associated with gaps 1, 3, and 4 are marked respectively as $\mathbf{p}_x$, $\mathbf{p}_y$, and $\mathbf{p}_z$. Let $\mathrm{d_{safe}}$ in Eq. (3.5) equals to R. In this case, $w(\mathrm{g1}) > 4R$, and hence, $\mathrm{d_s}(\mathrm{g1}) = 2R$. It can be seen that $\|\mathbf{p}_x - \mathbf{p}_1\| > \mathrm{d_s}(\mathrm{g1})$, which implies that the vision state of gap 1 is good. The same is true for gap 4, having in mind that $\mathrm{d_s}(\mathrm{g2}) = w(\mathrm{g2})/2$. When it comes to gaps 3 and 2, the distance between $\mathbf{p}_y$ and $\mathbf{p}_5$ is small, and $\overleftrightarrow{\mathrm{L}_1}$ does not pass through $\overline{\mathbf{p}_3\mathbf{p}_4}$. So, they both have a weak vision state.

Let $\mathcal{C}$ and $\mathcal{S}$ denote the closest gap and the safe gap, respectively. Determining the "closer side" of $\mathcal{S}$, $\mathbf{p}_{\mathrm{cr}}(\mathcal{S})$, is based on checking the "direct path" towards $\mathcal{C}$: assume that $\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})]$ represents the set of obstacle points that may cause collision with the direct path towards $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$. If this set is empty, $\mathbf{p}_{\mathrm{cr}}(\mathcal{S})$ is set to the closer side of $\mathcal{C}$. Otherwise, it is set to the obstacle point belonging to $\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})]$ and closest to the robot [MFM13b]:

$$\mathbf{p}_{\mathrm{cr}}(\mathcal{S}) = \begin{cases} \mathbf{p}_{\mathrm{cr}}(\mathcal{C}), & \text{if } \mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})] = \phi \\ \mathbf{p}_x, & \text{otherwise} \end{cases} \tag{3.14}$$

with:

$$\mathbf{p}_x = \underset{\mathbf{p}_i^{\mathrm{S}}}{\operatorname{argmin}} \|\mathbf{p}_i^{\mathrm{S}} - \mathbf{p}_r\| \tag{3.15}$$

where $\mathbf{p}_i^{\mathrm{S}} \in \mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})]$.

Figure 3.4 shows an example where only one gap $\mathcal{C}$ is detected, and hence it is the closest gap. As will be explained in section 3.1.5, the subgoal associated with $\mathcal{C}$, $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$, is located at the center point between its both sides. The direct path towards $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$ is checked by applying the algorithm presented in section 3.1.2. It is clear that all obstacles lying between $\mathbf{p}_r$ and $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$, visualized by dark gray, are included in the set of colliding obstacle points $\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})]$. Among these points, the closest to the robot (labeled $\mathbf{p}_x$ in figure 3.4) defines $\mathbf{p}_{\mathrm{cr}}(\mathcal{S})$.

The *farther side* $\mathbf{p}_{\mathrm{fr}}(\mathcal{S})$ is determined in two steps: in a first step, we identify the most appropriate obstacle point to locate $\mathbf{p}_{\mathrm{fr}}(\mathcal{S})$, denoted by $\mathbf{p}'_x$. Then, the location of this point is adjusted if the opening angle (angular width) between $\mathbf{p}'_x$ and $\mathbf{p}_x$ is small with respect to the current robot's view.

Stage 1: The workspace is divided into two regions; one to the left and the other to the right of the line that connects $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$ to $\mathbf{p}_r$. They are denoted by $\mathcal{W}^+$ and $\mathcal{W}^-$, respectively, as visualized in figure 3.4. Assume that r and l represent the indexes of the obstacle points forming the right and left sides of $\mathcal{C}$. Also, let $\mathrm{S}^- = \left\{\mathbf{p}_r^{\mathrm{S}}, \mathbf{p}_{r-1}^{\mathrm{S}}, ..., \mathbf{p}_1^{\mathrm{S}}\right\}$ and $\mathrm{S}^+ = \left\{\mathbf{p}_l^{\mathrm{S}}, \mathbf{p}_{l+1}^{\mathrm{S}}, ..., \mathbf{p}_n^{\mathrm{S}}\right\}$ be the lists of obstacle points falling to the right and left of $\mathcal{C}$, respectively. (i.e. $\mathrm{S}^+ \subset \mathrm{S}$ and $\mathrm{S}^- \subset \mathrm{S}$, where $\theta_i^{\mathrm{S}^+} \geq \theta_{\mathrm{l}}(\mathcal{C})$ and $\theta_j^{\mathrm{S}^-} \leq \theta_{\mathrm{r}}(\mathcal{C})$ for all $i \in \mathrm{S}^+$, $j \in \mathrm{S}^-$). In figure 3.4, S^- and

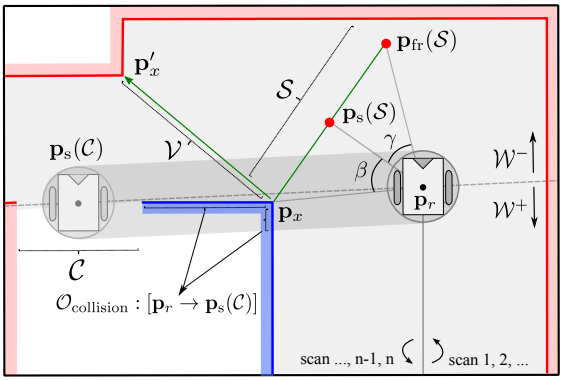


Figure 3.4: Determining the safe gap. The closest gap and the safe gap are denoted by $\mathcal{C}$ and $\mathcal{S}$. The robot is driven towards $\mathcal{S}$ rather than $\mathcal{C}$, ensuring a safer behavior and providing a gradual change in the steering angle (adapted from [MFM13b] with permission from IEEE).

S^+ are composed of the obstacles visualized by red and blue colors, respectively. We refer to the list located in the region that does not contain $\mathbf{p}_x$ as S':

$$\mathrm{S}' = \begin{cases} \mathrm{S}^+, & \text{if } \mathbf{p}_x \in \mathcal{W}^- \\ \mathrm{S}^-, & \text{otherwise} \end{cases} \tag{3.16}$$

For an empty S', we set the required obstacle point $\mathbf{p}'_x$ to the farther side of $\mathcal{C}$ (i.e. $\mathbf{p}'_x := \mathbf{p}_{\text{fr}}(\mathcal{C})$). Otherwise, S' is searched for $\mathbf{p}'_x$ sequentially until the angular distance between the accessed element and $\mathbf{p}_x$ gets more than π. Among the visited points in S', the closest to the robot is selected (see figure 3.4):

$$(\|\mathbf{p}'_x - \mathbf{p}_x\| \leq \|\mathbf{p}_i^{\mathrm{S}'} - \mathbf{p}_x\|) \wedge (\zeta \leq \pi), \quad \mathbf{p}_i^{\mathrm{S}'} \in \mathrm{S}' \tag{3.17}$$

where ζ is defined as:

$$\zeta = \begin{cases} \angle\mathbf{p}'_x - \angle\mathbf{p}_x, & \text{if } \angle\mathbf{p}_x < \theta_{\text{s}}(\mathcal{C}) \\ \angle\mathbf{p}_x - \angle\mathbf{p}'_x, & \text{otherwise} \end{cases} \tag{3.18}$$

where $\angle\mathbf{p}_x$ and $\angle\mathbf{p}'_x$ represent the direction towards $\mathbf{p}_x$ and $\mathbf{p}'_x$, respectively.

Stage 2: A virtual gap $\mathcal{V}$ is constructed whose sides are set to $\mathbf{p}_x$ and $\mathbf{p}'_x$ (i.e. $\mathbf{p}_{\mathrm{cr}}(\mathcal{V}) := \mathbf{p}_x$, $\mathbf{p}_{\mathrm{fr}}(\mathcal{V}) := \mathbf{p}_x{}'$). Then, the opening angle of $\mathcal{V}$ is checked following the "gap vision state" criterion. If it has a wide opening angle (i.e. in a good vision state), the farther side of $\mathcal{S}$ is set to $\mathbf{p}'_x$ ($\mathbf{p}_{\mathrm{fr}}(\mathcal{S}) := \mathbf{p}'_x$). Otherwise, it is set in such a way that the resultant $\mathcal{S}$ has a good vision state (see figure 3.4):

$$\theta_{\mathrm{fr}}(\mathcal{S}) = \begin{cases} \mathrm{proj}\left(\theta_{\mathrm{cr}}(\mathcal{V}) - \vartheta\right), & \text{if } \theta_{\mathrm{cr}}(\mathcal{V}) \geq \theta_{\mathrm{fr}}(\mathcal{V}) \\ \mathrm{proj}\left(\theta_{\mathrm{cr}}(\mathcal{V}) + \vartheta\right), & \text{otherwise} \end{cases} \tag{3.19}$$

$$r_{\mathrm{fr}}(\mathcal{S}) = r_{\mathrm{cr}}(\mathcal{S}) \,.\, \frac{\cos(\beta)}{\cos(\gamma)} \tag{3.20}$$

where ϑ, γ, and β are defined as:

$$\beta = \arcsin\left(\frac{\mathrm{d_s}(\mathcal{V})}{r_{\mathrm{cr}}(\mathcal{V})}\right) \tag{3.21}$$

$$\gamma = \mathrm{atan2}\left(w(\mathcal{V}) - \mathrm{d_s}(\mathcal{V}),\ r_{\mathrm{cr}}(\mathcal{V})\cos(\beta)\right) \tag{3.22}$$

$$\vartheta = \Gamma(\mathcal{V}) \,.\, (\beta + \gamma) \tag{3.23}$$

It can be deduced from the above equations that $\mathcal{S}$ is created by rotating $\mathcal{V}$ around $\mathbf{p}_{\mathrm{cr}}(\mathcal{V})$, so that its vision state gets good (the angle of rotation is ϑ) [MFM13b]. Driving the robot towards $\mathcal{S}$ provides a safer and smoother bridge to the goal. This is because the subgoal corresponding to $\mathcal{S}$ is located within a *free area* and provides a *gradual change* in the steering angle while progressing towards $\mathcal{C}$. Using the ND variants, on the other hand, the robot is directly driven towards $\mathcal{C}$, and only if the clearance to obstacles gets small the trajectory is adjusted based on the APF. This may lead to sudden turns which is unsafe in tight spaces.

3.1.5 Determining Motion Commands

Up to now, we have seen how to determine the most appropriate gap ($\mathcal{S}$) for navigation. As pointed out in section 3.1.1, a gap g is traversed by assigning a subgoal $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$ and driving the robot towards it. In the following, we show how to compute the exact location of $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$. Let GL be the line segment that connects both sides of g. In order to achieve a safe navigation, $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$ is located

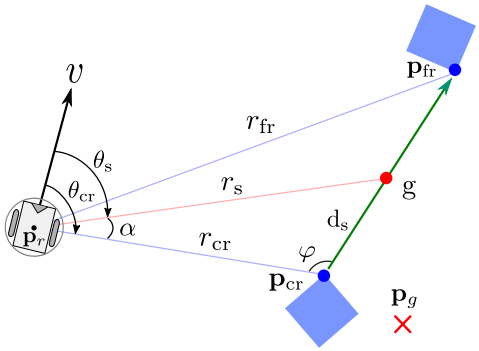


Figure 3.5: Determining the subgoal corresponding to gap g. For clarity, the notation representing the gap, (g), is dropped, e.g. $r_s(g)$ becomes r_s.

at a distance of $d_s(g)$ from $\mathbf{p}_{cr}(g)$ (the closer side of g). Therefore, the polar coordinates of $\mathbf{p}_s(g)$ are computed as follows (see figure 3.5) [MFM13b]:

$$r_s(g) = \sqrt{(d_s(g))^2 + (r_{cr}(g))^2 - 2d_s(g) \,.\, r_{cr}(g) \,.\, \cos(\varphi)} \tag{3.24}$$

$$\theta_s(g) = \begin{cases} \text{proj}\,(\theta_{cr}(g) - \alpha)\,, & \text{if } \theta_{cr}(g) \geq \theta_{fr}(g) \\ \text{proj}\,(\theta_{cr}(g) + \alpha)\,, & \text{otherwise} \end{cases} \tag{3.25}$$

where φ and α are defined as:

$$\varphi = \arccos\left(\frac{(w\,(g))^2 + (r_{cr}(g))^2 - (r_{fr}(g))^2}{2w\,(g) \cdot r_{cr}(g)}\right) \tag{3.26}$$

$$\alpha = \arccos\left(\frac{(r_s(g))^2 + (r_{cr}(g))^2 - (d_s(g))^2}{2r_s(g) \cdot r_{cr}(g)}\right) \cdot \Gamma(g) \tag{3.27}$$

where $\Gamma(g)$ is used to direct the robot to the right direction (that leads to g).

The SG method considers controlling the robot's speed based on the clearance to obstacles, achieving a safer behavior. This is achieved by specifying v_{limit} which is set in such a way that the robot slows down as it gets close to obstacles, coming to a full stop once a collision occurs [MFMJ10] [MFM13b] [MFM15] [MFM16]:

$$v_{limit} = \left(\sqrt{1 - \text{sat}_{[0,1]}\left(\frac{D_{vs} - r_{min}}{D_{vs}}\right)}\right) \,.\, v_{max} \tag{3.28}$$

where $r_{\min}$ is the distance to the closest obstacle, $v_{\max}$ the maximum limit of the linear speed, and D_{vs} a parameter, which we refer to in [MFM15] as a "velocity safe distance". The value of D_{vs} defines the size of a zone around the robot in which the velocity is limited. A higher D_{vs} increases safety but reduces speed. The sat operator in Eq. (3.28) caps the velocity at $v_{\max}$ if all obstacles are outside D_{vs} and at 0 if the robot ever touches an obstacle. Obviously, there is a direct proportional relationship between the value of v_{limit} and the distance to the obstacle that falls within D_{vs} and closest to the robot.

After having the current target (goal or subgoal) determined, we can compute the motion control which drives a mobile robot towards its location. For the sake of a fair comparison in sections 3.2 and 3.3, we use the same equations proposed in [MM04] (also used in [DB08] and [MFMJ10]):

$$v = \text{sat}_{[0,1]} \left(\frac{\pi/4 - |\theta_{\text{traj}}|}{\pi/4} \right) . v_{\text{limit}} \tag{3.29}$$

$$\omega = \text{sat}_{[-1,1]} \left(\frac{\theta_{\text{traj}}}{\pi/2} \right) . w_{\max} \tag{3.30}$$

where θ_{traj} can be θ_{g} or $\theta_{\text{s}}(\mathcal{S})$ based on investigating the "direct path" towards $\mathbf{p}_g$ (see section 3.1.2), and $\omega_{\max}$ the maximum limit of the rotational speed.

3.2 Simulation Results

This section demonstrates the differences in behavior between the ND variants and the proposed SG method. Two different scenarios are shown; the objective of the first scenario, previously presented in [MFM13b], is to verify the increased safety and smoothness of the trajectories generated by the SG method compared to those generated by the ND variants. Here, the behavior of the SG method is compared to one ND variant, the Closest Gap (CG) [MFMJ10]. The second scenario shows how irrational robot's deflections towards free spaces occurring in the ND variants have been avoided by applying the SG method. In this scenario, the behavior of the SG method is compared to the ND+ [MM04], SND [DB08], and CG techniques [MFMJ10]. All discussed methods have been implemented using the well-known Robot Operating System (ROS) [QCG+09].

3.2.1 Scenario 1 Simulations

For this scenario, the map shown in figure 3.6 was created, where the task was to successfully drive a mobile robot from a start to a goal location. Our objective was to verify the impact of employing the safe gap, introduced in section 3.1.4, in enhancing the robot's trajectory. The simulated robot has a rectangular shape with a width of $0.48\,m$ and a length of $0.52\,m$ and works in a differential-driven mode. The sensing system is a laserscanner which delivers 1024 measurements over 360° and covers a range of $10\,m$. The maximum limits of the linear and rotational speeds were set to $0.5\,m/s$ and $0.5\,rad/s$, respectively, whereas the velocity safe distance (D_{vs}) was set to $0.7\,m$. In the CG technique, the parameters defining the safe distance D_s and the weight of deflection associated with nearby obstacles (k) were set to $0.7\,m$ and 1.0, respectively.

The path generated by the CG approach is visualized in figure 3.6a, where the progress of the robot is depicted by drawing red rectangles at uniform time intervals. It can be deduced from the density of these rectangles that CG suffers from rapid changes in the direction of motion θ_{traj}. By employing SG, the route is safely and smoothly traversed as can be seen from figure 3.6b. Figures 3.6d - 3.6g show snapshots of the simulation taken at successive time stamps. With the CG approach, the robot is directly driven towards the closest gap $\mathcal{C}$ unless the clearance to obstacles gets less than the security zone D_s, in which case, the robot's trajectory is re-planned, resulting is sharp trajectory changes. For instance, at the starting location the distance to obstacles is greater than D_s. Therefore, the robot is directly driven towards $\mathcal{C}$, although the direct path is occupied by obstacles. Whenever point 1 is reached (figure 3.6a), the distance to the obstacles marked as A gets less than D_s, and hence, the direction of motion is adjusted by an angle D_{net} [MFMJ10]. The SG approach, on the other hand, employs a safer and smoother bridge (the safe gap $\mathcal{S}$) between both cases (existence or absence of obstacles within D_s). It can be deduced from figures 3.6d - 3.6g that subgoals generated within safe gaps provide a gradual change in the steering angle while traversing the open starting region. Once the robot travels through the area between the points labeled 2 and 3, CG and SG behave fairly similar. This is because the direct path towards $\mathcal{C}$ is collision-free and D_s

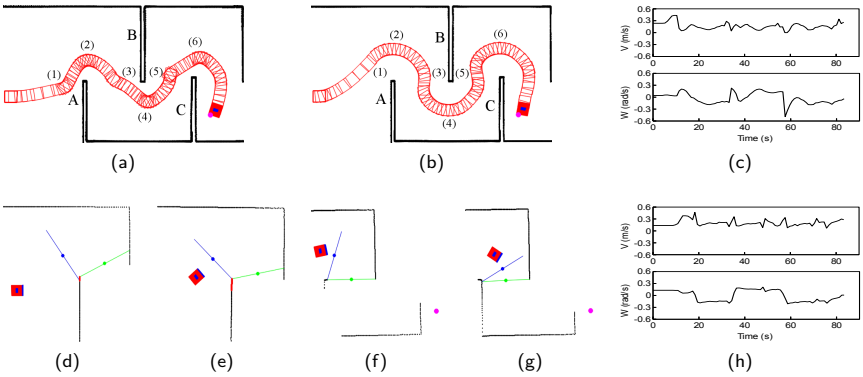


Figure 3.6: Scenario 1 simulations. (a) Path generated by the CG method, where rapid changes in the direction of motion occurs. (b) Smoother path generated by SG. (c) Speed profile of the CG method. (d-g) Snapshots of the SG simulation. The goal and obstacles are visualized by magenta circle and black lines, respectively. The closest gap $\mathcal{C}$ and the safe gap $\mathcal{S}$ are shown by green and blue line segments, where subgoals are represented by small circles on the center of gaps. The obstacle points in collision with the direct path towards $\mathcal{C}$ are visualized by red color. (h) Speed profile of the SG method (reprinted from [MFM13b] with permission from IEEE).

is always occupied by the A obstacles (figures 3.6f, 3.6g). Similar improvements in performance can be seen looking at the trajectory next to the points labeled 3 - 6. We confirm our visualization by recording the translational and rotational speeds and plotting them versus time in figures 3.6c and 3.6h.

3.2.2 Scenario 2 Simulations

For this scenario, an environmental structure with tight passages was created (the width of the passages is within $[0.75 - 1.15]\,m$). It is shown in figure 3.7, where the initial and goal configurations are also visualized. In order to reach the assigned goal, the robot had to negotiate several curvy roads. The main objective of this scenario was to verify the capability of the SG method to achieve better

safety, efficiency, and smoothness compared to the ND-based variants. Another objective was to show how SG avoids irrational deflections towards free spaces. The simulated robot is a differential drive whose shape is rectangular. Its length and width are $0.53\,m$ and $0.49\,m$, respectively. The linear and angular speeds were limited to the ranges $[-0.5, 0.5]$ and $[-1.0, 1.0]$, respectively. The adopted laser scanner delivers 683 scan points and covers a range of $5.6\,m$ with a field of view of $240°$. The parameter which determines the speed limit D_{vs} is set to $1\,m$. The security zone in the ND variants and the weight of deflection in the CG method were set to $D_s = 1\,m$ and $k = 1$, respectively.

3.2.2.1 Simulations for ND+, SND and CG

Figures 3.7a - 3.7c show the trajectories generated by the ND+, SND and CG techniques, respectively. The robot managed to reach the goal in 135 seconds by applying the SND Approach. By employing ND+ and CG, the robot was faster as it reached the goal in 129 seconds. Figures 3.7e, 3.7f, and 3.7g show the robot velocities visualized against the time for ND+, SND and CG, respectively. ND+ suffers from suddenly changing the angular speed (see the large spikes in figure 3.7e). According to the SNG and CG methods, no real differences in behavior can be seen except at one location where, using the SND method, the robot almost touched the obstacle creating the side marked as D. A common behavior for all these methods is the unnecessary deflection of the robot towards unoccupied regions while making progress towards the target, see for example the locations labeled 1-6 in figures 3.7b and 3.7c, and the locations labeled 1-3 in figure 3.7a. The reason behind this drawback is the usage of the artificial potential field concept, where repulsive forces are exerted onto the robot from nearby obstacles. Another reason is the determination of the avoidance trajectory without respecting the location and field of view of the closest gap.

3.2.2.2 Simulation for SG

The path generated by running the SG method is visualized in figure 3.7d. The time required by the robot to reach the goal was only 94 seconds. At the starting

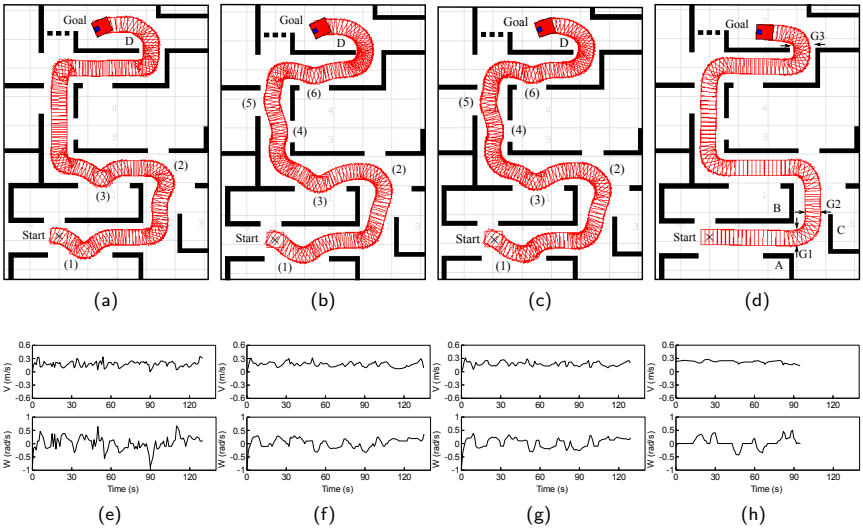


Figure 3.7: Scenario 2 simulations. (a-d) Paths generated using the implementation of ND+, SND, CG, and SG, respectively. (e-h) Linear and angular velocities visualized against the time elapsed for ND+, SND, CG, and SG, respectively.

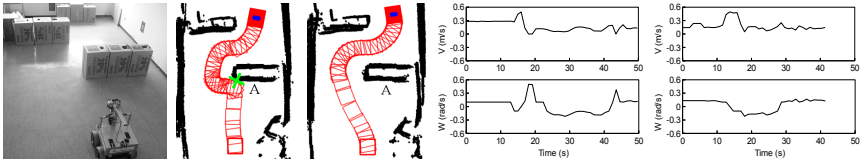
location, the gap closest to the goal is created by A and B obstacles (labeled G1), where the direct path towards it is unoccupied by obstacles. In such a case, the safe gap is identical to the closest gap, and therefore the robot moves directly towards the subgoal corresponding to G1 (located at the center of G1 here). As soon as the robot navigates through G1, the closest gap gets G2 that is created by B and C obstacles. At that moment, the direct path towards G2 is occupied by obstacles, and thus the robot smoothly and safely navigates towards the safe gap created by the closest obstacle (on the side labeled B here). This procedure continues until having moved through all passages and eventually having attained the target. It is obvious from the path next to G3 that the SG method avoids the drawback of approaching the wall having less density of obstacle points, occurring in SND. Additionally, it can be deduced from the density of the red rectangles that the SG method enhances the safety, efficiency, and smoothness

of the trajectories generated by the above discussed ND variants. Looking at the velocity profile in figure 3.7h supports our visualization.

3.3 Experimental Results

The improved performance of the SG method has been demonstrated utilizing our Pioneer 3-AT mobile robot GETbot, whose dimensions are (0.52×0.48 m). GETbot works in a skid-steering mode and subject to nonholonomic constraints. It is controlled using a 2.6 GHz Intel core i7-620M CPU and equipped with a Hokuyo-UTM-30LX laserscanner. The maximum speeds of GETbot are ($v_{\max} = 0.7\,m/s$, $w_{\max} = 2.4\,rad/s$). While performing the experiments, these velocities were restricted to ($v_{\max} = 0.5\,m/s$, $w_{\max} = 0.5\,rad/s$). In this section we show one of our experiments (figure 3.8), which was previously published in [MFM13b]. For the sake of comparison, this experiment was executed using SG and CG. The values of D_s and k in the CG method were set to $0.7\,m$ and 1.0, respectively.

Figures 3.8b and 3.8c show the paths generated by CG and SG, respectively. It can be deduced from these figures that, in terms of smoothness, the difference between both methods is roughly similar to that presented in section 3.2.1. However, when it comes to safety the difference becomes clearer as the robot requires time for deceleration and turning once obstacles appear inside D_s. This can be depicted from figure 3.8b where the CG-controlled GETbot almost touched the obstacle marked as A. The improved safety of SG over CG can be clearly seen from figure 3.8c. The speed profiles for both methods are shown in figures 3.8d and 3.8e. After performing several experiments, it was observed that the safety of CG can be improved by decreasing the value of $v_{\max}$ or increasing the value of D_s. However, decreasing $v_{\max}$ is unfavorable, especially in environments having a mixture of wide and narrow passages. In these environments, it is desirable to use the maximum possible velocity. Increasing the value of D_s causes another problem that is the failure of guiding the robot through a tight opening if the density of obstacles on one side is much higher than the other. This problem can be reduced by increasing the value of k. However, tuning k is not straightforward; increasing its value leads to oscillations and unstable behavior. Decreasing its value, on the other hand, leads to collision. In summary, with SG this tedious



(a) (b) (c) (d) (e)

Figure 3.8: Experiments (reprinted from [MFM13b] with permission from IEEE). (a) Environment setup (b) Path generated by applying the CG approach, where the robot almost touched the obstacle marked as A. (c) Path generated by SG achieving smoother and safer behavior. (d) Speed profile for CG. (e) Speed profile for SG.

parameter tuning ($v_{\max}$, D_s, or k) is avoided either in wide or in tight spaces. Additionally, SG improves the safety, smoothness, and efficiency of the robot's motion. Notice that the next chapters include *additional experiments* where the performance of the discussed methods is quantitatively estimated.

3.4 Conclusions

In this chapter, the "Safe Gap" (SG) approach for reactive collision avoidance has been addressed. With SG, the smoothness and safety of the robot's motion have been improved compared to the ND-based variants. This has been achieved by incorporating an additional step in analyzing the sensory data, locating a virtual gap in a collision-free area, referred to as a "safe gap". The location of this gap is determined based on its opening angle and the configuration of the goal, providing a smoother and safer bridge between obstacle avoidance and goal approach. This also helps in evading unreasonable deviations towards free spaces, reducing the total time and distance required to complete the mission. Unlike the ND-based obstacle avoidance techniques, the SG method does not require the safe distance parameter, and thus saves the parameter tuning overhead. An additional, yet important aspect of the SG approach is the simplicity of the problem formulation and implementation.

4 Smooth Navigation in Unstructured Narrow Spaces - A Holonomic Solution

In chapter 3, we have seen how the trajectories generated by the ND navigation methods have been enhanced by introducing a safer and smoother bridge between collision avoidance and target approach. This bridge, referred to as a "safe gap", is determined by analyzing the environmental structure and extracting the set of obstacle points in collision with the "gap trajectory" (the trajectory followed by the robot to reach the closest gap [MFM15]). Among these points, only the closest to the robot is utilized to create the safe gap. In real-world scenarios, the sensor measurements are prone to instability and uncertainty. Moreover, the robot environments are often unstructured and subject to changes. In this regard, the location of the closest obstacle creating the safe gap may vary rapidly over time. This can lead to oscillatory motion, especially at relatively higher speeds. Another yet significant drawback occurs in narrow spaces, in which the obstacle creating the safe gap may rapidly alternate between being located to the right or left of the direction of motion. These sudden changes cause successive right and left turns, which in turn may lead to oscillations and instability [MFM16].

The "Tangential Gap Flow" (TGF) navigation presented in this chapter is specially developed to cope with the above mentioned limitations. The key idea of the TGF method is the computation of the motion control based on the configuration of all obstacles causing collision with the "gap trajectory", not simply the nearest one. Moreover, the clearance to obstacles on both sides of the heading direction is taken into account. Adjusting the gap trajectory is, in general, based on two concepts, namely "tangential" and "gap flow" navigation [MFM16]. Using the "tangential navigation", the robot moves tangential to the obstacles boundary. With the "gap flow navigation", the robot safely and smoothly navigates among

closely spaced obstacles. In both concepts, avoiding collisions and approaching the target are simultaneously performed. By employing these enhancements, the smoothness of the generated trajectories is increased. Furthermore, a much more reliable yet stable motion is achieved. Last but not least, the motion commands, that drive a mobile robot towards a given target, is computed in such a way that the stability of the system is guaranteed in the Lyapunov sense.

In a nutshell, the TGF approach can be described in the following: in a first step, the data coming from sensors is analyzed to characterize the distribution of surrounding obstacles and identify the current motion situation. Once determined, an action is carried out as explained in section 4.1. With this step, the desired heading direction that guarantees both collision avoidance and target approach is computed. Section 4.2 describes how to set the motion control which drives a robot towards the desired heading. This process is repeated at each sensor update. Experimental results including a discussion and comparison with the SG method as well as with existing ND variants are introduced in section 4.3. In section 4.4, our conclusions are highlighted. Notice that the definitions and notations introduced in section 3.1.1 are also used here.

4.1 Motion Situations and Corresponding Actions

In order to avoid the risk of collision with obstacles, it is essential to adjust the direction of motion based on the distribution of surrounding obstacles. Hence, once an obstacle obstructs the robot's path, a temporary rotation for the goal position is performed until the risk is passed. As we have reported in [MFM16] [MJFM13] [Muj10], the degree of rotation is determined based on two criteria:

Criterion 1: "Path-to-goal". This criterion considers two situations: "Free-path" and "Dangerous-path". For determining which situation is currently active, the configuration space is created and the holonomic path towards the goal is checked for collision. If it is collision-free, the situation is a "Free-path". Otherwise, it is a "Dangerous-path". The former situation does not require any action as can be depicted from figure 4.1a, while the latter imposes a rotation to the goal position by an angle Ψ_{sg}, referred to as a "gap rotation angle". This rotation heads the

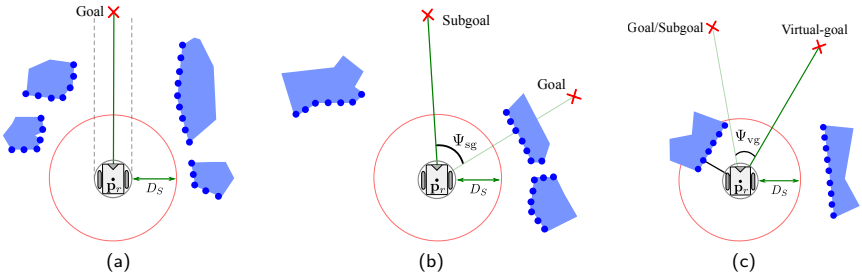


Figure 4.1: Visualization of three motion situations based on the "Path-to-goal" and "Safety" criteria. (a) "Free-path" and "High-safety": this situation does not require any action. (b) "Dangerous-path" and "High-safety": in this case, a rotation to the goal position is performed, driving the robot towards a subgoal within the closest gap. (c) "Low-safety": a temporary rotation to the goal (resp. subgoal) position is performed so that the robot avoids collisions with obstacles (adapted from [Muj10] and from [MJFM13] with permission from IEEE).

robot towards a *subgoal* within the *closest gap* as visualized in figure 4.1b. For constructing the closest gap, see section 3.1.3 from chapter 3.

Criterion 2: "Safety". This criterion considers two situations: "High-safety" and "Low-safety". Determining the currently active situation is dependent on the existence (Low-safety) or absence (High-safety) of a potential threat from obstacles. Traditionally, an obstacle is considered a threat if the distance to it measured from the robot's boundary is less than a predefined safe distance D_s, as visualized in figure 4.1. However, the proposed TGF method considers an obstacle as a threat if it causes collision with the robot's boundary while traveling *directly* towards the goal/subgoal (see section 3.1.2), i.e. an obstacle point $\mathbf{p}_i^{\mathrm{S}}$ is a threat if the following condition is fulfilled:

$$\mathbf{p}_i^{\mathrm{S}} \in \mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_t] \tag{4.1}$$

where $\mathbf{p}_t$ can be the given goal or the computed subgoal based on investigating the "Path-to-goal" situations. This definition is more reasonable and eliminates the need for the D_s parameter.

Hence, to determine the current situation, the "direct path" towards $\mathbf{p}_t$ is checked for collision. If it is collision-free (i.e. $\mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_t] = \phi$), the situation is a "High-safety". Otherwise, it is a "Low-safety". The former situation does not require any action, while the latter imposes a rotation to the goal/subgoal location by an angle Ψ_{vg}, referred to as a "collision avoidance rotation angle". The result of this rotation is a new goal location, called a "virtual-goal", as shown in figure 4.1c. The actions associated with the Path-to-goal criterion are executed before those associated with the safety criterion (they have a higher priority).

At first, the computation of the "gap rotation angle" Ψ_{sg} is described in section 4.1.1. Then, in section 4.1.2 the "tangential navigation" concept is introduced and employed to determine the "collision avoidance rotation angle" Ψ_{vg}. Subsequently, in section 4.1.3 the "gap flow navigation" concept is explained, where Ψ_{vg} is computed in such a way that the clearance to obstacles on both sides of the heading direction is considered. Both concepts compute the rotation angle considering only one obstacle point (to simplify understanding). Later, in section 4.1.4, the smoothness of the generated trajectories is increased by integrating both concepts and by considering all threats surrounding the robot.

4.1.1 Gap Rotation Angle

The first step in determining Ψ_{sg} is to extract the set of surrounding gaps and identify the navigable one closest to the goal. For this purpose, the procedure presented in section 3.1.3 is followed. The outcome of this step is the closest gap $\mathcal{C}$. Setting Ψ_{sg} is, in principle, based on the goal location and the angular distance between both sides of $\mathcal{C}$, as we have proposed in [MFM16] [MJFM13] [Muj10]; if the goal falls within $\mathcal{C}$, Ψ_{sg} is set to zero, since the objective is to drive the robot towards the goal and it is inappropriate to drive it somewhere else. Otherwise, Ψ_{sg} is computed based on the angular width of $\mathcal{C}$ (look at figure 4.2).

$$\Psi_{\text{sg}} = \begin{cases} \theta_{\text{mid}} - \theta_g, & \text{if } \angle(\theta_{\text{cg}}(\mathcal{C}), \theta_{\text{mid}}) < \angle(\theta_{\text{cg}}(\mathcal{C}), \theta_{\text{scs}}) \\ \theta_{\text{scs}} - \theta_g, & \text{otherwise} \end{cases} \tag{4.2}$$

where θ_g is the angle towards the goal, $\theta_{\text{cg}}(\mathcal{C})$ the direction towards the side of $\mathcal{C}$ closer to the goal, and θ_{mid} and θ_{scs} are defined as (originally from [DB08]):

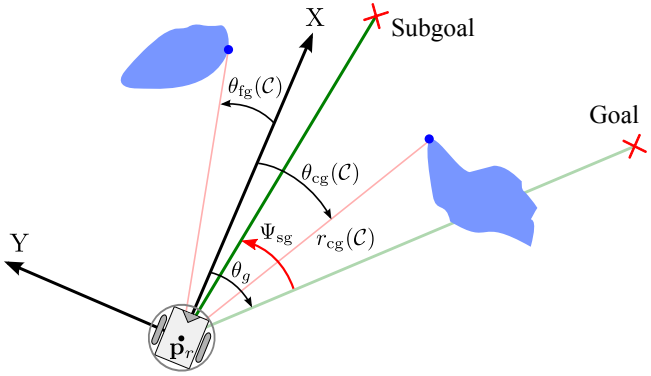


Figure 4.2: Computing the gap rotation angle Ψ_{sg}. The closest gap $\mathcal{C}$ is assumed narrow. Therefore, Ψ_{sg} is determined in such a way that the robot passes through the gap center $\mathcal{C}$.

$$\theta_{\mathrm{mid}} = \begin{cases} \theta_{\mathrm{cg}}(\mathcal{C}) - \frac{\left(\theta_{\mathrm{cg}}(\mathcal{C}) - \theta_{\mathrm{fg}}(\mathcal{C})\right)}{2}, & \text{if } \theta_{\mathrm{cg}}(\mathcal{C}) > \theta_{\mathrm{fg}}(\mathcal{C}) \\ \theta_{\mathrm{cg}}(\mathcal{C}) + \frac{\left(\theta_{\mathrm{fg}}(\mathcal{C}) - \theta_{\mathrm{cg}}(\mathcal{C})\right)}{2}, & \text{otherwise} \end{cases} \tag{4.3}$$

$$\theta_{\mathrm{scs}} = \begin{cases} \theta_{\mathrm{cg}}(\mathcal{C}) - \arcsin\left(\frac{R + \mathrm{d_{safe}}}{r_{\mathrm{cg}}(\mathcal{C})}\right), & \text{if } \theta_{\mathrm{cg}}(\mathcal{C}) > \theta_{\mathrm{fg}}(\mathcal{C}) \\ \theta_{\mathrm{cg}}(\mathcal{C}) + \arcsin\left(\frac{R + \mathrm{d_{safe}}}{r_{\mathrm{cg}}(\mathcal{C})}\right), & \text{otherwise} \end{cases} \tag{4.4}$$

where $r_{\mathrm{cg}}(\mathcal{C})$ is the distance to the side $\mathbf{p}_{\mathrm{cg}}(\mathcal{C})$ of gap $\mathcal{C}$ closer to the goal, $\theta_{\mathrm{fg}}(\mathcal{C})$ the direction towards the side of $\mathcal{C}$ farther from the goal, and $\mathrm{d_{safe}}$ a desired clearance to obstacles as defined in section 3.1.1. It can be deduced from Eq. (4.2) that, for a narrow $\mathcal{C}$, the robot passes through the gap center. However, if $\mathcal{C}$ is wide, $\mathrm{d_{safe}}$ is preserved between $\mathbf{p}_{\mathrm{cg}}(\mathcal{C})$ and the robot's footprint.

4.1.2 Tangential Rotation Angle

Up to now, we have seen how the instantaneous goal[1] is determined based on analyzing the holonomic path towards the goal. For brevity, from now on the

[1] The instantaneous goal can be the goal or the subgoal based on investigating the "Path-to-goal" criterion (criterion 1).

instantaneous goal is named the target and denoted by $\mathbf{p}_t$. As mentioned previously, for a "Low-safety" situation, TGF considers rotating $\mathbf{p}_t$ by an angle, referred to as a "collision avoidance rotation angle" Ψ_{vg}. Within the "tangential navigation" concept, previously published in [MFM16] [MJFM13] [Muj10], Ψ_{vg} is computed in such a way that the robot navigates tangential to the boundary of the closest obstacle (considered as a threat) in the direction of the target:

$$\Psi_{\text{vg}} = \begin{cases} -\text{sgn}(\theta_{\text{c}})\frac{\pi}{2} - \gamma, & \text{if } |\gamma| < \pi \wedge \text{sgn}(\theta_t) \neq \text{sgn}(\theta_{\text{c}}) \\ -\text{sgn}(\theta_{\text{c}})\frac{3\pi}{2} - \gamma, & \text{if } |\gamma| \geq \pi \wedge \text{sgn}(\theta_t) \neq \text{sgn}(\theta_{\text{c}}) \\ -\text{sgn}(\theta_{\text{c}})\frac{\pi}{2} - \gamma, & \text{if } |\theta_{\text{c}}| \geq |\theta_t| \wedge \text{sgn}(\theta_t) = \text{sgn}(\theta_{\text{c}}) \\ +\text{sgn}(\theta_{\text{c}})\frac{\pi}{2} - \gamma, & \text{if } |\theta_{\text{c}}| < |\theta_t| \wedge \text{sgn}(\theta_t) = \text{sgn}(\theta_{\text{c}}) \end{cases} \tag{4.5}$$

where θ_t is the angle towards $\mathbf{p}_t$, θ_{c} the direction towards $\mathbf{p}_{\text{c}}$ (the obstacle point closest to the robot), and $\gamma = \theta_t - \theta_{\text{c}}$.

The Tangential Escape (TE) approach [FPV+08] builds upon a similar concept. However, this method can only work properly in simple environments, as the TE-controlled robot always seeks the goal without considering the environmental structure (creating subgoals within free gaps). It has been shown in [MJFM13] that, by employing the TE method, the robot can get stuck in different scenarios (e.g U-shaped obstacles). Furthermore, it is not stated in [FPV+08] how navigation towards the goal is resumed after following the contour of an obstacle and when it is necessary to circumnavigate the boundary of a different obstacle, i.e. setting a *leaving condition*. By creating subgoals in free areas (gaps), the "tangential navigation" determines Ψ_{vg} so that local trap situations are avoided.

In figure 4.3 different scenarios are shown, where the virtual target (the result of rotating the target $\mathbf{p}_t$ by Ψ_{vg}) is marked as $\mathbf{p}_{\text{vg}}$. The corresponding collision avoidance rotation angle Ψ_{vg} for each scenario is also visualized. By implementing the TE approach [FPV+08], the robot may move far away from $\mathbf{p}_t$. For instance, the dashed red arrows in figures 4.3b, 4.3e, and 4.3f show the directions towards which the robot is driven using TE. The proposed "tangential navigation" concept avoids this limitation by creating subgoals within free areas, guiding the robot towards the direction leading to the goal.

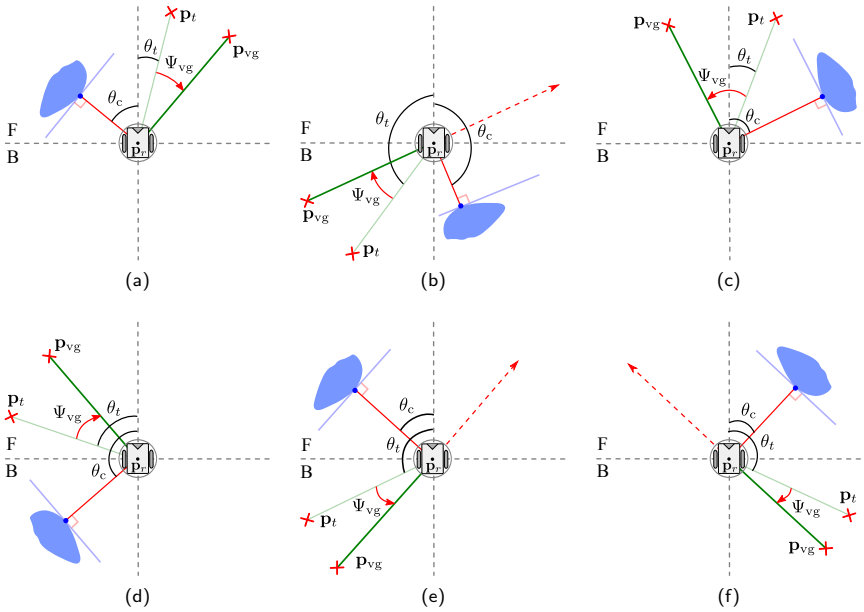


Figure 4.3: Computing Ψ_{vg} based on the tangential navigation concept for different cases. (a, b) $\text{sgn}(\theta_t) \neq \text{sgn}(\theta_c)$ where $|\gamma| < \pi$ for (a) and $|\gamma| \geq \pi$ for (b). (c, d): $\text{sgn}(\theta_t) = \text{sgn}(\theta_c)$ and $|\theta_c| \geq |\theta_t|$ where $\theta_c < 0$ for (c) and $\theta_c > 0$ for (d). (e, f): $\text{sgn}(\theta_t) = \text{sgn}(\theta_c)$ and $|\theta_c| < |\theta_t|$ where $\theta_c > 0$ for (e) and $\theta_c < 0$ for (f). (adapted from [Muj10] and from [MFM16] [MJFM13] with permissions from Elsevier and IEEE).

The *leaving condition* is met if the angular distance between θ_c and θ_t exceeds $\frac{\pi}{2}$, i.e. The value of Ψ_{vg} calculated above is set to zero if $\angle(\theta_t, \theta_c) > \frac{\pi}{2}$.

4.1.3 Gap Flow Rotation Angle

In section 4.1.2, we have seen how the collision avoidance rotation angle Ψ_{vg} is computed in such a way that the robot navigates tangential to the contour of the closest obstacle i.e. the angular difference between the direction of motion and the closest obstacle $\angle(\theta_t, \theta_c)$ is maintained at 90°. In this regard, approaching

one side of a *narrow passage* causes the robot to do a sharp turn so that it moves parallel to this side (regardless of the location of obstacles on the other side). As soon as the robot starts to turn, there is a big chance that it gets closer to the other side of the passage causing a sharp turn towards the opposite direction, and the process repeats. These sudden turn changes may lead to oscillations and instability. Figure 4.4 shows an example where the robot is supposed to pass through the narrow gap created by obstacles (sides) A and B. Let us assume that the current situation according to criteria 1 and 2 is a "Dangerous-path" and "Low-safety". Since we have only one gap, it will be the *closest gap* and the target $\mathbf{p}_t$ (subgoal here) is located between its both sides. At the starting point (marked as 1 in figure 4.4a), the obstacle point closest to the robot is located on side A. This imposes a rotation to $\mathbf{p}_t$ by Ψ_{vg} and the robot navigates tangential to side A accordingly. Whenever the location labeled 3 in figure 4.4b is reached, the robot gets closer to side B. At that point, the new rotation angle causes the robot to move tangential to side B instead. Once location 4 is approached, the robot gets closer to side A again, moving tangential to it as can be depicted from figure 4.4c. As soon as the angular distance between the closer side (here side B) and $\mathbf{p}_t$ approaches $\frac{\pi}{2}$ (figure 4.4d), the *leaving condition* is fulfilled, guiding the robot directly towards $\mathbf{p}_t$. It is worth to mention that for a wide gap, these oscillatory transitions are reduced as the leaving condition may be fulfilled earlier.

The "gap flow navigation" concept, previously published in [MFM16] [MFM15], addresses this limitation by performing the avoidance maneuver in such a way that the clearance to obstacles on both sides of $\mathbf{p}_t$ is taken into account. In the following, it is shown how to compute the avoidance trajectory associated with any threat $\mathbf{p}_i^{\text{S}}$, using this concept. In order to enhance the readability, the superscript S is removed (i.e. $\mathbf{p}_i^{\text{S}}$ is abbreviated $\mathbf{p}_i$).

Let $\mathbf{p}_r$ be the location of the robot's origin and $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$ the line segment representing the holonomic path towards $\mathbf{p}_t$. The workspace is divided into two parts; one to the right ($\mathcal{R}^-$) and the other to the left ($\mathcal{R}^+$) of $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$. Let $\mathbf{p}_i$ be any threat and $\mathcal{R}^*$ the region that does not include $\mathbf{p}_i$. The avoidance trajectory is determined in such a way that the clearance between $\mathbf{p}_i$ and the obstacles falling on $\mathcal{R}^*$ is considered. Among the obstacle points located in $\mathcal{R}^*$, the closest to $\mathbf{p}_i$, denoted by $\mathbf{p}_i^*$, is selected. Since our objective is to drive the robot towards $\mathbf{p}_t$, we exclude

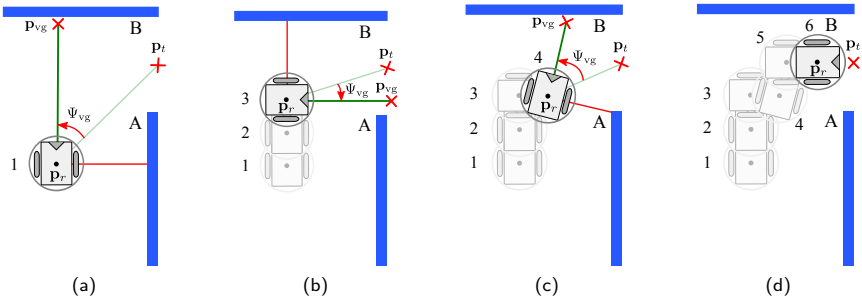


Figure 4.4: Oscillations that may occur in a tight passage by employing the "tangential navigation" concept. (a) The closest obstacle point is located on side A. This imposes a rotation to $\mathbf{p}_t$ by Ψ_{vg} and the robot navigates tangential to A accordingly. (b) The robot gets closer to side B. At that point, the new Ψ_{vg} causes the robot to move tangential to B. (c) The robot gets closer to side A again, moving tangential to it. (d) Fulfilling the *leaving condition*, and in turn guiding the robot directly towards $\mathbf{p}_t$ (adapted from [MFM16] with permission from Elsevier).

all obstacles making an angular distance more than π with $\mathbf{p}_i$, traveling in the direction of $\mathbf{p}_t$. This is due to the fact that these obstacles do not pose a collision risk, as they are located behind the robot while driving it towards $\mathbf{p}_t$ [MFM15]:

$$\mathbf{p}_i^* = \underset{\mathbf{p}_k \in S}{\operatorname{argmin}} \|\mathbf{p}_k - \mathbf{p}_i\|, \ \beta < \alpha < \pi \tag{4.6}$$

with β and α given by:

$$\alpha = \begin{cases} \text{proj}\,(\theta_i - \theta_t) - \text{proj}\,(\theta_i^* - \theta_t), & \text{if } \mathbf{p}_i \text{ is located in } \mathcal{R}^+ \\ \text{proj}\,(\theta_i^* - \theta_t) - \text{proj}\,(\theta_i - \theta_t), & \text{otherwise} \end{cases} \tag{4.7}$$

$$\beta = \begin{cases} \text{proj}\,(\theta_i - \theta_t), & \text{if } \mathbf{p}_i \text{ is located in } \mathcal{R}^+ \\ -\text{proj}\,(\theta_i - \theta_t), & \text{otherwise} \end{cases} \tag{4.8}$$

where θ_i^* is the angle towards $\mathbf{p}_i^*$.

As an example, look at figure 4.5, where $\mathbf{p}_i$ is located in the left region $\mathcal{R}^+$, and therefore $\mathcal{R}^* = \mathcal{R}^-$. The obstacle points visualized by red are excluded as they are not falling in $\mathcal{R}^*$ (they violate the condition $\alpha > \beta$ from Eq. (4.6)).

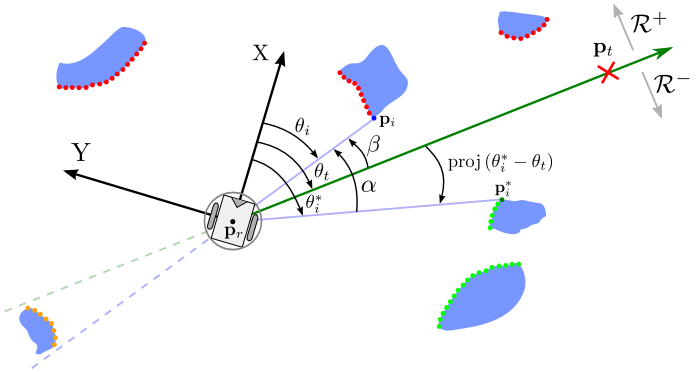


Figure 4.5: Considering the clearance to both sides of a target $\mathbf{p}_t$ while computing the avoidance trajectory. The line towards $\mathbf{p}_t$ divides the workspace into two regions, $\mathcal{R}^+$ and $\mathcal{R}^-$. It is obvious that $\mathbf{p}_i$ is located in $\mathcal{R}^+$. Hence, while computing Ψ_{vg} associated with $\mathbf{p}_i$ the clearance between $\mathbf{p}_i$ and the obstacles located in $\mathcal{R}^-$ (visualized by green and orange) is considered. Since our objective is to drive the robot towards $\mathbf{p}_t$, all obstacles making an angular distance $> \pi$ with $\mathbf{p}_i$ are excluded (such as those visualized by orange). Among the remaining obstacles, the closest to $\mathbf{p}_i$ is selected (denoted $\mathbf{p}_i^*$ and visualized by dark green).

Additionally, the orange obstacle points are excluded as they make an angular distance more than π with $\mathbf{p}_i$ (they violate the condition $\alpha < \pi$ from Eq. (4.6)). The remaining obstacle points are visualized by green. Among these valid points, $\mathbf{p}_i^*$ is set to the closest to $\mathbf{p}_i$ (visualized by dark green in figure 4.5).

We now introduce an angle, referred to as "gap flow angle" $\Delta(\mathbf{p}_i)$ that defines the angular distance between a threat $\mathbf{p}_i$ and the desired avoidance trajectory associated with it [MFM16]. Notice that this angle is maintained at 90° by using the tangential navigation concept, discussed in section 4.1.2. Our objective here is to restrict this angle, so that the obstacles located in $\mathcal{R}^*$ are considered.

Let θ_{center} be the angle towards the center point between $\mathbf{p}_i$ and $\mathbf{p}_i^*$, i.e.:

$$\theta_{\text{center}} = \text{atan2}\left(\frac{y_i + y_i^*}{2}, \frac{x_i + x_i^*}{2}\right) \tag{4.9}$$

where (x_i, y_i) and (x_i^*, y_i^*) denote the Cartesian coordinates of $\mathbf{p}_i$ and $\mathbf{p}_i^*$.

The "gap flow angle" corresponding to $\mathbf{p}_i$ is defined as follows:

$$\Delta(\mathbf{p}_i) = \arccos\left(\frac{1}{2} \,.\, \frac{A^2 + r_i^2 - B^2}{A \,.\, r_i}\right) \tag{4.10}$$

where r_i is the distance to $\mathbf{p}_i$, and B and A are given by:

$$B = \begin{cases} \min\left(\frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_i^*\|, \mathrm{d_{safe}}\right), & \text{if } r_i <= r_i^* \\ \min\left(|\, r_i \sin\left(\angle\left(\theta_i, \theta_{\mathrm{center}}\right)\right)|\,, \mathrm{d_{safe}}\right), & \text{otherwise} \end{cases} \tag{4.11}$$

$$A = \sqrt{B^2 + r_i^2 - 2\,B \,.\, r_i \,.\, \cos\left(\frac{\pi}{2} - \angle\left(\theta_i, \theta_{\mathrm{center}}\right)\right)} \tag{4.12}$$

where r_i^* is the distance to $\mathbf{p}_i^*$. It is obvious from Eq. (4.10) that $\Delta(\mathbf{p}_i)$ distinguishes two cases based on whichever of $\mathbf{p}_i$ and $\mathbf{p}_i^*$ is closer to the robot; if $\mathbf{p}_i$ is closer ($r_i <= r_i^*$), $\Delta(\mathbf{p}_i)$ points the robot so that $\frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is maintained to $\mathbf{p}_i$, see figure 4.6a. For a large distance between $\mathbf{p}_i$ and $\mathbf{p}_i^*$, the maintained clearance is limited to $\mathrm{d_{safe}}$, see figure 4.6b. On the other hand, if $r_i > r_i^*$, $\Delta(\mathbf{p}_i)$ points the robot towards the center point between $\mathbf{p}_i$ and $\mathbf{p}_i^*$, see figure 4.7a. Similarly, if this makes the clearance to $\mathbf{p}_i$ relatively large, the distance between the robot and $\mathbf{p}_i$ is capped to $\mathrm{d_{safe}}$, see figure 4.7b. Having in mind all $\Delta(\mathbf{p}_i)$ on both regions ($\mathcal{R}^+$ and $\mathcal{R}^-$), the avoidance trajectories associated with all $\mathbf{p}_i$ can be visualized as vector flow fields between all $\mathbf{p}_i$ and $\mathbf{p}_i^*$, where the flow direction points towards the target (could be the "closest gap" or the goal based on investigating the "Path-to-Goal" criterion) and an appropriate distance to obstacles is maintained. That is why $\Delta(\mathbf{p}_i)$ is called the "gap flow angle" [MFM16].

With this, we can now define the rotation angle imposed on $\mathbf{p}_t$ to avoid $\mathbf{p}_i$ as:

$$\Psi_{\mathrm{vg}} = \Gamma(\mathbf{p}_i)\left[-\Delta(\mathbf{p}_i) + \mathrm{sat}_{[0,\Delta(\mathbf{p}_i)]}\left(\angle\left(\theta_t, \theta_i\right)\right)\right] \in [-\frac{\pi}{2}, \frac{\pi}{2}[\tag{4.13}$$

where $\Gamma(\mathbf{p}_i)$ is given by:

$$\Gamma(\mathbf{p}_i) = \begin{cases} +\mathrm{sgn}\left(\theta_i - \theta_t\right), & \text{if } |\theta_i - \theta_t| \leq \pi \\ -\mathrm{sgn}\left(\theta_i - \theta_t\right), & \text{otherwise} \end{cases} \tag{4.14}$$

Notice that $\Gamma(\mathbf{p}_i)$ in Eq. (4.13) is used to point the robot towards the direction closer to $\mathbf{p}_t$ rather than using the 4 cases in Eq. (4.5).

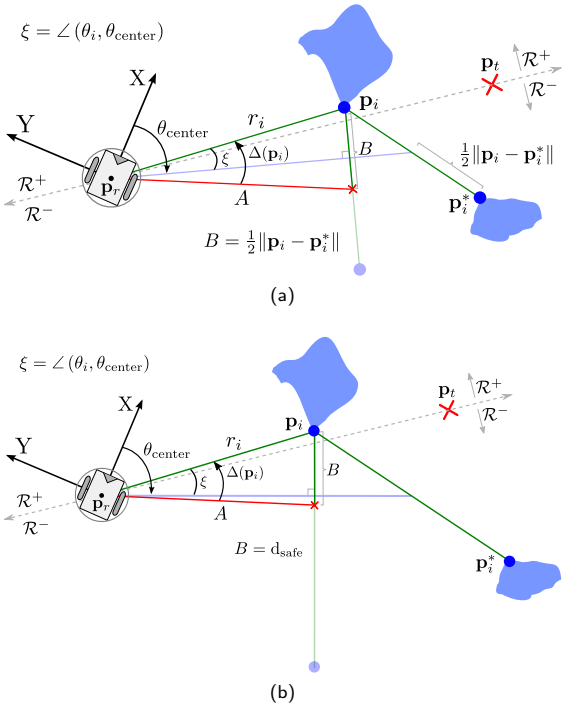


Figure 4.6: Computing $\Delta(\mathbf{p}_i)$. $\mathbf{p}_i$ is closer to $\mathbf{p}_r$ than $\mathbf{p}_i^*$. Hence, $\Delta(\mathbf{p}_i)$ is set such that $\frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is maintained to $\mathbf{p}_i$ as visualized in (a). If $\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is high, the maintained distance is limited to d_{safe}, see (b).

4.1.4 Tangential Gap Flow Rotation Angle

As mentioned at the beginning of this chapter, the "Tangential Gap Flow (TGF)" navigation determines the motion control based on the configuration of all surrounding threats, achieving a better performance in unstructured environments. Next, we show how the "tangential" and "gap flow" navigation concepts are employed to achieve this goal, following our work in [MFM15] [MFM16] [MFM17].

The foundation of TGF is the determination of the collision risk acted by each of the surrounding obstacles. It consists of the following steps. In a first step, we extract all obstacle points (threats) that may cause collision with the robot

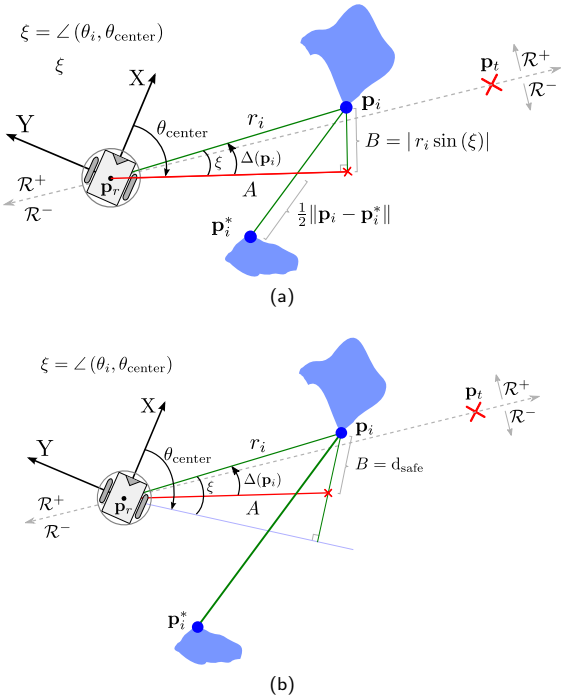


Figure 4.7: Computing $\Delta(\mathbf{p}_i)$. $\mathbf{p}_i^*$ is closer to $\mathbf{p}_r$ than $\mathbf{p}_i$: $\Delta(\mathbf{p}_i)$ is set such that the robot moves towards the center point between $\mathbf{p}_i$ and $\mathbf{p}_i^*$, see (a). If $\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is high, the maintained distance is set to d_{safe}, see (b).

while guiding it towards $\mathbf{p}_t$. In a second step, the rotation angle ψ_i associated with each of these threats is computed. The "weighted average rotation angle" Ψ_{vg} is then calculated based on the degree of risk posed by each threat.

For determining the set of threats, we perform the following steps. First, the current situation is determined according to criterion 1. If it is a "Free-path" situation, no action is required. Otherwise, $\mathbf{p}_t$ is translated to a safer location within $\mathcal{C}$ (figure 4.8). The distance r_t to the new location of $\mathbf{p}_t$ is computed as:

$$r_t = \begin{cases} d(\mathbf{p}_r, \mathbf{p}_g), & \text{if } \Psi_{\text{sg}} = 0 \\ \frac{r_{\text{cg}}(\mathcal{C})\, \sin(\zeta)}{\sin(\pi - (\zeta + \eta))}, & \text{otherwise} \end{cases} \tag{4.15}$$

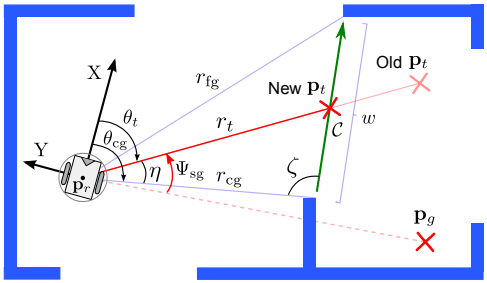


Figure 4.8: The target $\mathbf{p}_t$ is translated to a safer location between both sides of the closest gap $\mathcal{C}$ if the holonomic path towards $\mathbf{p}_t$ is unsafe. This step is necessary to determine the set of obstacles that may cause collision with the robot while guiding it towards $\mathbf{p}_t$. For clarity, the notation representing the closest gap ($\mathcal{C}$) is removed, e.g. $\theta_{cg}(\mathcal{C})$ is abbreviated as θ_{cg} (adapted from [MFM16] with permission from Elsevier).

where ζ and η are given by:

$$\zeta = \arccos\left(\frac{w^2(\mathcal{C}) + r_{cg}^2(\mathcal{C}) - r_{fg}^2(\mathcal{C})}{2w(\mathcal{C})r_{cg}(\mathcal{C})}\right) \tag{4.16}$$

$$\eta = \begin{cases} \angle\left(\theta_{cg}(\mathcal{C}) \rightarrow \theta_t\right), & \text{if } \theta_{cg}(\mathcal{C}) > \theta_{fg}(\mathcal{C}) \\ \angle\left(\theta_t \rightarrow \theta_{cg}(\mathcal{C})\right), & \text{otherwise} \end{cases} \tag{4.17}$$

where $r_{fg}(\mathcal{C})$ is the distance to the side $\mathbf{p}_{fg}(\mathcal{C})$ of gap $\mathcal{C}$ farther from the goal and $w(\mathcal{C})$ the width of $\mathcal{C}$.

In a second step, the "direct path" from $\mathbf{p}_r$ to $\mathbf{p}_t$ is checked for collision, following the algorithm presented in section 3.1.2. The outcome $N = \mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_t]$[2] is the set of obstacles that may cause collision with the robot while driving it from its current location towards $\mathbf{p}_t$.

In a third step, a virtual reference frame, named "robot-target" frame (denoted by $\mathcal{F}_{rt}$), is created by performing a rotation to the robot coordinate system so that it points towards $\mathbf{p}_t$ (performing a rotation by θ_t) [MFM16]. Throughout

[2]Notice that the direct path can rapidly change between being free or occupied. To provide a smoother and safer transition between both states, we enlarge the robot's footprint to both sides of $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$ while extracting $\mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_t]$. In our experiments, the robot's footprint was inflated by a value of $2R$.

this section, the Cartesian coordinates of an obstacle point $\mathbf{p}_i^{\mathrm{S}}$ with respect to the "robot-target" frame $\mathcal{F}_{rt}$ is denoted by $(T(x_i^{\mathrm{S}}),\, T(y_i^{\mathrm{S}}))$:

$$T(x_i^{\mathrm{S}}) = x_i^{\mathrm{S}} \cos(-\theta_t) - y_i^{\mathrm{S}} \sin(-\theta_t) \tag{4.18}$$

$$T(y_i^{\mathrm{S}}) = x_i^{\mathrm{S}} \sin(-\theta_t) + y_i^{\mathrm{S}} \cos(-\theta_t) \tag{4.19}$$

With the "robot-target" frame $\mathcal{F}_{rt}$, the list of colliding obstacle points (N) is divided into two sublists; one contains those obstacles lying to the left of $\mathcal{F}_{rt}$ (to the left of its x-axis) while the other includes those obstacles located to the right. The first is called a left-sublist and represented by N_{L}, whereas the other is called a right-sublist and represented by N_{R} :

$$N_{\mathrm{R}} = \left\{ \mathbf{p}_i^{\mathrm{S}} \in N \;\middle|\; T(y_i^{\mathrm{S}}) < 0 \right\} \tag{4.20}$$

$$N_{\mathrm{L}} = \left\{ \mathbf{p}_i^{\mathrm{S}} \in N \;\middle|\; T(y_i^{\mathrm{S}}) > 0 \right\} \tag{4.21}$$

In order to avoid unreasonable deviations towards free regions, threats located within U-shaped objects are discarded. For this purpose, we eliminate from both sublists each obstacle point whose absolute y-coordinate (relative to $\mathcal{F}_{rt}$) is more than that of the closest to the robot. By this means, the efficiency and safety of the generated trajectories are enhanced. Let $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_R}}$ be the obstacle point belonging to N_{R} and closest to the robot. Similarly, $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_L}}$ represents the obstacle point included in N_{L} and closest to the robot:

$$\mathbf{p}_{\mathrm{c}}^{\mathrm{N_R}} = \underset{\mathbf{p}_j^{\mathrm{S}}}{\operatorname{argmin}} \|\mathbf{p}_j^{\mathrm{S}} - \mathbf{p}_r\|, \quad \mathbf{p}_j^{\mathrm{S}} \in N_{\mathrm{R}} \tag{4.22}$$

$$\mathbf{p}_{\mathrm{c}}^{\mathrm{N_L}} = \underset{\mathbf{p}_j^{\mathrm{S}}}{\operatorname{argmin}} \|\mathbf{p}_j^{\mathrm{S}} - \mathbf{p}_r\|, \quad \mathbf{p}_j^{\mathrm{S}} \in N_{\mathrm{L}} \tag{4.23}$$

The modified right $(\hat{N}_{\mathrm{R}})$ and left $(\hat{N}_{\mathrm{L}})$ subsets are then defined as follows:

$$\hat{N}_{\mathrm{R}} = N_{\mathrm{R}} \setminus \left\{ \mathbf{p}_i^{\mathrm{S}} \in N_{\mathrm{R}} : \left| T(y_i^{\mathrm{S}}) \right| > \left| T(y_{\mathrm{c}}^{\mathrm{N_R}}) \right| \right\} \tag{4.24}$$

$$\hat{N}_{\mathrm{L}} = N_{\mathrm{L}} \setminus \left\{ \mathbf{p}_i^{\mathrm{S}} \in N_{\mathrm{L}} : \left| T(y_i^{\mathrm{S}}) \right| > \left| T(y_{\mathrm{c}}^{\mathrm{N_L}}) \right| \right\} \tag{4.25}$$

where $y_{\mathrm{c}}^{\mathrm{N_R}}$ and $y_{\mathrm{c}}^{\mathrm{N_L}}$ are the y-coordinates of $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_R}}$ and $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_L}}$, respectively.

Each obstacle point $\mathbf{p}_i^{\mathrm{S}}$ included in either N_{L} or N_{R} imposes a rotation to the target $\mathbf{p}_t$ by an angle, denoted as ψ_i. Setting ψ_i is based on the tangential and gap flow navigation concepts. Let H represents the sublist including threat $\mathbf{p}_i^{\mathrm{S}}$ and H^* the other sublist[3]. Also, denote threat $\mathbf{p}_i^{\mathrm{S}}$ by $\mathbf{p}_i^{\mathrm{H}}$ and the threat closest to $\mathbf{p}_i^{\mathrm{S}}$ and contained in H^* by $\mathbf{p}_i^{\mathrm{H}^*}$:

$$\mathbf{p}_i^{\mathrm{H}^*} = \underset{\mathbf{p}_j^{\mathrm{S}}}{\operatorname{argmin}} \|\mathbf{p}_j^{\mathrm{S}} - \mathbf{p}_i^{\mathrm{H}}\|, \quad \mathbf{p}_j^{\mathrm{S}} \in H^* \tag{4.26}$$

The rotation angle imposed by $\mathbf{p}_i^{\mathrm{H}}$ is defined as follows:

$$\psi_i = \Gamma(\mathbf{p}_i^{\mathrm{H}}) \left[-\lambda + \mathrm{sat}_{[0,\lambda]} \left(\angle \left(\theta_t, \theta_i^{\mathrm{H}} \right) \right) \right] \in [-\frac{\pi}{2}, \frac{\pi}{2}[\tag{4.27}$$

where θ_i^{H} is the angle towards $\mathbf{p}_i^{\mathrm{H}}$, and $\Gamma(\mathbf{p}_i^{\mathrm{H}})$ and λ are given by:

$$\Gamma(\mathbf{p}_i^{\mathrm{H}}) = \begin{cases} +\mathrm{sgn}\left(\theta_i^{\mathrm{H}} - \theta_t\right), & \text{if } \left|\theta_i^{\mathrm{H}} - \theta_t\right| \leq \pi \\ -\mathrm{sgn}\left(\theta_i^{\mathrm{H}} - \theta_t\right), & \text{otherwise} \end{cases} \tag{4.28}$$

$$\lambda = \begin{cases} \frac{\pi}{2}, & \text{if } H^* = \phi \\ \Delta(\mathbf{p}_i^{\mathrm{H}}), & \text{otherwise} \end{cases} \tag{4.29}$$

In Eq. (4.27), we distinguish two cases based on whether H^* is empty or not. If it is empty, ψ_i is set in such a way that the robot moves tangential to $\mathbf{p}_i^{\mathrm{H}}$ ("tangential navigation"). In such a case, if $\mathbf{p}_i^{\mathrm{H}}$ is the threat closest to the robot, the result of Eq. (4.27) is equivalent to Eq. (4.5). However, if H^* is not empty (there are threats on the other side of the "robot-target" frame), ψ_i is determined such that the angular distance between $\mathbf{p}_i^{\mathrm{H}}$ and its avoidance trajectory is maintained at the "gap flow angle" $\Delta(\mathbf{p}_i^{\mathrm{H}})$ ("gap flow navigation"). Notice that Eq. (4.27) already includes the leaving condition; whenever the angular distance between $\mathbf{p}_t$ and $\mathbf{p}_i^{\mathrm{H}}$ gets greater than $\frac{\pi}{2}$ (for a tangential navigation) or $\Delta(\mathbf{p}_i^{\mathrm{H}})$ (for a gap flow navigation), the value of ψ_i is capped at 0 (using the sat operator), and therefore, the robot is directly driven towards $\mathbf{p}_t$. By this means, a larger clearance to $\mathbf{p}_i^{\mathrm{H}}$ is maintained. Moreover, avoiding $\mathbf{p}_i^{\mathrm{H}}$ and making progress towards the target $\mathbf{p}_t$ are simultaneously considered [MFM16] [MFM17].

[3]The left-sublist if $\mathbf{p}_i^{\mathrm{S}} \in \hat{N}_{\mathrm{R}}$ and the right-sublist if $\mathbf{p}_i^{\mathrm{S}} \in \hat{N}_{\mathrm{L}}$.

As mentioned above, it is required to compute the degree of risk posed by each $\mathbf{p}_i^{\mathrm{H}}$. This is reflected by the relative proximity of $\mathbf{p}_i^{\mathrm{H}}$ to the robot and can be expressed as:

$$w_i = \left(\mathrm{sat}_{[0,1]} \frac{\left(r_t + R - r_i^{\mathrm{H}}\right)}{\left(r_t + R - r_{\mathrm{c}}^{\hat{\mathrm{N}}}\right)} \right)^2 \tag{4.30}$$

where r_i^{H} is the distance to $\mathbf{p}_i^{\mathrm{H}}$ and $r_{\mathrm{c}}^{\hat{\mathrm{N}}}$ the distance to the obstacle posing the highest risk (the closest to the robot and included in $\hat{N} = \hat{N}_{\mathrm{R}} \cup \hat{N}_{\mathrm{L}}$). The risk measure w_i is the maximum (1) if $\mathbf{p}_i^{\mathrm{H}}$ is the closest threat and the minimum (0) if $r_i^{\mathrm{H}} \geq r_t + R$, where $r_t + R$ is the largest possible distance to a threat belonging to $\hat{N}$. Notice that w_i gets higher as the distance to the threat gets smaller.

The "weighted average rotation angle" Ψ_{vg} is determined for the left and right sides (sublists) separately. By this means, the net Ψ_{vg} will not be affected by the number of threats in each sublist [MFM16]. As a consequence, the computed trajectory will not be biased towards the side having the least number of threats (for more details about this issue, the reader may refer to [MFMJ10]). Let $\mathrm{T_R}$ be the total number of obstacle points included in the right sublist $\hat{N}_{\mathrm{R}}$, we define the "weighted average rotation angle" caused by these obstacles (called "right-side rotation angle") as the weighted sum of their corresponding rotation angles:

$$\Psi_{\mathrm{R}} = \frac{1}{\mathrm{W_R}} \sum_{i=1}^{\mathrm{T_R}} w_i \psi_i \in [-\frac{\pi}{2}, \frac{\pi}{2}[, \quad \mathbf{p}_i^{\mathrm{S}} \in \hat{N}_{\mathrm{R}} \tag{4.31}$$

where $\mathrm{W_R}$ is the total weight corresponding to the obstacles contained in $\hat{N}_{\mathrm{R}}$:

$$\mathrm{W_R} = \sum_{i=1}^{\mathrm{T_R}} w_i, \quad \mathbf{p}_i^{\mathrm{S}} \in \hat{N}_{\mathrm{R}} \tag{4.32}$$

Analogously, the "left-side rotation angle" Ψ_{L} is computed. Both Ψ_{R} and Ψ_{L} are weighted, reflecting the relative risk posed by each side. The weight corresponding to Ψ_{R} can be expressed as [MFM15] [MFM16]:

$$\mathrm{W}(\Psi_{\mathrm{R}}) = \underbrace{w_{\max}^{\mathrm{R}}}_{\text{left-term}} \cdot \underbrace{\left(1 - \frac{\Psi_{\max} - |\Psi_{\mathrm{R}}|}{\Psi_{\max}}\right)}_{\text{right-term}} \tag{4.33}$$

where $w^{\mathrm{R}}_{\max}$ is the weight corresponding to the obstacle ($\mathbf{p}_{\mathrm{c}}^{\hat{\mathrm{N}}_{\mathrm{R}}}$) closest to the robot among those contained in N_{R} and $\Psi_{\max}$ the maximum absolute rotation angle among both Ψ_{R} and Ψ_{L} ($\Psi_{\max} = \max(|\Psi_{\mathrm{R}}|, |\Psi_{\mathrm{L}}|)$). It is apparent that the value of $\mathrm{W}(\Psi_{\mathrm{R}})$ depends on two factors: first, the distance between $\mathbf{p}_{\mathrm{c}}^{\hat{\mathrm{N}}_{\mathrm{R}}}$ and the robot reflected by the maximum weight $w^{\mathrm{R}}_{\max}$ (left-term). Notice that $\mathbf{p}_{\mathrm{c}}^{\hat{\mathrm{N}}_{\mathrm{R}}}$ is considered among those contained in $\hat{N}_{\mathrm{R}}$ as it poses the highest threat. The second factor is expressed by the relative difference between $\Psi_{\max}$ and $|\Psi_{\mathrm{R}}|$ (right-term). With this factor, more weight is given to the side imposing a higher rotation to $\mathbf{p}_t$. For instance, if $|\Psi_{\mathrm{R}}|$ is greater than $|\Psi_{\mathrm{L}}|$, the right term evaluates to 1. Otherwise, its value depends on the difference between $|\Psi_{\mathrm{R}}|$ and $|\Psi_{\mathrm{L}}|$: it increases as $|\Psi_{\mathrm{R}}|$ get closer to $|\Psi_{\mathrm{L}}|$. The idea here is to make the side imposing a larger rotation angle (posing a higher risk) the dominant side. This term reduces successive turn changes (oscillations) occurring as a result of changing the dominant side between being left Ψ_{L} or right Ψ_{R} , resulting in a smooth variation of Ψ_{vg}. The weight associated with Ψ_{L}, $\mathrm{W}(\Psi_{\mathrm{L}})$, is calculated in the same manner.

Finally, the net rotation angle corresponding to all $\mathbf{p}_i^{\mathrm{S}} \in \hat{N}$ ("tangential gap flow" rotation angle Ψ_{vg}) can be expressed as the weighted average of Ψ_{R} and Ψ_{L}:

$$\Psi_{\mathrm{vg}} = \frac{\mathrm{W}(\Psi_{\mathrm{R}})\Psi_{\mathrm{R}} + \mathrm{W}(\Psi_{\mathrm{L}})\Psi_{\mathrm{L}}}{\mathrm{W}(\Psi_{\mathrm{R}}) + \mathrm{W}(\Psi_{\mathrm{L}})} \in [-\frac{\pi}{2}, \frac{\pi}{2}[\tag{4.34}$$

4.2 Determining Motion Commands

Up to now, we have determined both angles of rotation, Ψ_{sg} and Ψ_{vg}. As we have pointed out in section 4.1, The goal location is rotated by these angles to achieve a collision free motion, while progressing towards the goal. Therefore, at each sensor update, the instantaneous goal location is computed as [MFM16]:

$$\hat{\mathbf{p}}_g = \begin{bmatrix} \cos(\Psi) & \sin(\Psi) \\ -\sin(\Psi) & \cos(\Psi) \end{bmatrix} \mathbf{p}_g \tag{4.35}$$

where $\Psi = \Psi_{\mathrm{sg}} + \Psi_{\mathrm{vg}}$ is the total rotation angle.

Next, we show how to determine the motion control that guides a mobile robot towards $\hat{\mathbf{p}}_g$. A unicycle type mobile robot is considered whose control inputs are

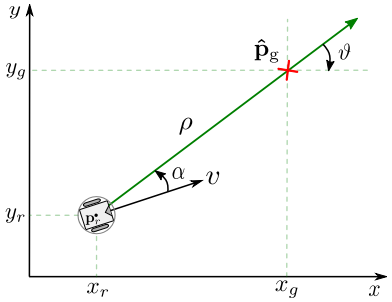


Figure 4.9: A robot navigates towards a given goal location (adapted from [MJFM13] with permission from IEEE).

the linear and angular velocities (v and ω). Its configuration with respect to the global (world) coordinate frame is illustrated in figure 4.9. It has been shown in the literature that this robot obeys the following kinematic model (given in polar coordinates) [ACBB95] [PSV11]:

$$\dot{\rho} = -v\cos(\alpha) \tag{4.36}$$

$$\dot{\alpha} = -w + v\frac{\sin(\alpha)}{\rho} \tag{4.37}$$

$$\dot{\vartheta} = -v\frac{\sin(\alpha)}{\rho} \tag{4.38}$$

where ρ is the distance to the goal $\hat{\mathbf{p}}_g$, α the angle towards $\hat{\mathbf{p}}_g$ (orientation error), and ϑ the angular distance between the horizontal axis and the line connecting $\hat{\mathbf{p}}_g$ to the robot's origin. Notice that the values of $\dot{\alpha}$ and $\dot{\vartheta}$ are indefinite in case that ρ in Eq. (4.37) and Eq. (4.38) is zero. Therefore, it is assumed that the robot reaches $\hat{\mathbf{p}}_g$ if the value of ρ gets below a small threshold ϵ.

The goal is to determine a state dependent controller which guarantees that both α and ρ asymptotically go to zero. For this system, the following control inputs, previously published in [MFM16] [MJFM13] [Muj10], are proposed:

$$v = k_b v_{\text{limit}}\cos(\alpha), \quad k_b > 0 \tag{4.39}$$

$$\omega = k_m\alpha + \frac{v\sin(\alpha)}{\rho}, \quad k_m > 0 \tag{4.40}$$

This final pose controller is adapted from [FPV+08]. If the current situation according to criterion 1 is a "Free-path", the value of k_b in Eq. (4.39) is set to $\tanh(\rho)$. Otherwise, it is set 1. This helps in achieving smooth breaking whenever the distance to p_g approaches zero. The parameter k_m in Eq. (4.40) is utilized to restrict ω to its maximum limit. It is set to $k_m = \frac{2\omega_{max}}{\pi}$ in the experiments presented in this thesis. Next, we describe how to control the speed of the robot (determine v_{limit}), and subsequently, we analyze the stability of the system.

4.2.1 Limiting Speed

The TGF method considers limiting the robot's speed based on the distance to nearby obstacles, similar to the SG method presented in chapter 3. This has been addressed by specifying v_{limit} (Eq. (3.28)) whose value is based on two factors; the distance to the closest obstacle and a parameter D_{vs}, named "velocity safe distance". The value of this parameter defines the size of a region around the robot in which the velocity is restricted. Here, we propose to determine the value of D_{vs} based on the physical and dynamical properties of the robot. For this purpose, we utilize the work in [MLO+98], which proposes a lookahead distance d_l for assessing the performance requirements of range sensors and for measuring the quality of obstacle detection algorithms. One of those requirements is the ability of range sensors to detect an obstacle with sufficient resolution at the proposed distance. Mainly, the lookahead distance is composed of three terms:

$$d_l = \underbrace{d_b}_{\text{first-term}} + \underbrace{v_i(2t_c + t_a)}_{\text{second-term}} + \underbrace{v_i^2/(2\mu\hat{\mathrm{g}})}_{\text{third-term}} \tag{4.41}$$

The first term is a buffer distance which accounts for the distance between the sensor and the robot's boundary plus any desired safety margin from obstacles. The second term is a reaction distance which is the distance traveled by the robot before the obstacle avoidance maneuver starts; it is based on the initial velocity v_i, the computation time t_c, and the actuation latency time t_a. The third term is the breaking distance, which is the distance the robot travels before coming to a full stop once the brake is engaged. μ and $\hat{\mathrm{g}}$ denote the coefficient of friction and the gravitational acceleration, respectively.

For our mobile robot, the values of the buffer distance, the combined reaction time, and the coefficient of friction are set experimentally to $0.5\,m$, $0.5\,s$, and 0.7, respectively. In case of hills, the coefficient of friction is reduced by a function of the slope angle [MLO+98].

With this, we define the value of v_{limit} as follows:

$$v_{\text{limit}} = \left(\sqrt{1 - \text{sat}_{[0,1]} \left(\frac{D_{vs} - r_{\text{c}}^{\hat{\text{N}}}}{D_{vs}} \right)} \right) v_{\max} \tag{4.42}$$

where $D_{\text{vs}} = 0.5 + 0.5\,|v_i| + 0.073 v_i^2$.

4.2.2 Stability Analysis

In order to investigate the stability of the proposed final pose controller, the following Lyapunov function candidate is considered:

$$V = \frac{1}{2}\alpha^2 + \frac{1}{2}\rho^2 \tag{4.43}$$

It is apparent that V is a positive definite. Its time derivative is:

$$\dot{V} = \alpha\dot{\alpha} + \rho\dot{\rho} \tag{4.44}$$

Now, by substituting the values of $\dot{\rho}$ and $\dot{\alpha}$ from Eq. (4.36) and Eq. (4.37) into Eq. (4.44), we obtain:

$$\dot{V} = \alpha \left(-w + v\frac{\sin(\alpha)}{\rho} \right) - \rho v \cos(\alpha) \tag{4.45}$$

Finally, by replacing v and ω from Eq. (4.39) and Eq. (4.40) into Eq. (4.45), $\dot{V}$ can be rewritten in the following form:

$$\dot{V} = -k_m \alpha^2 - k_b v_{\text{limit}} \rho \cos^2(\alpha) \tag{4.46}$$

It is obvious from Eq. (4.46) that $\dot{V}$ is a negative definite, thus demonstrating the asymptotic stability of the proposed controller (i.e. $[\alpha \quad \rho] \to [0 \quad 0]$ as $t \to \infty$).

4.3 Experimental Results

Several experiments were conducted using our mobile robot GETbot. The objective of these experiments was to demonstrate the improved stability and smoothness of TGF and to compare its performance to that of the ND variants as well as SG. In the following, three experiments are presented. These experiments were performed using the implementation of two ND variants (ND+ [MM04] and CG [MFMJ10]) in addition to the proposed SG and TGF methods. The experimental setup is introduced in chapter 5, where additional experiments including a performance evaluation are also provided.

In experiment 1, it was supposed to drive GETbot through relatively tight gaps made up of boxes and chairs as can be seen in figure 4.10a. Experiment 2 (figure 4.11a) had two challenges: first, the robot had to avoid obstacles forming a U-like shape to reach the goal. Second, the obstacle course included tight openings having one side with higher density of obstacles compared to the other (see, for instance, passages P2 and P4). The environmental structure of experiment 3 consisted of various narrow gaps where the available clearance at some locations did not exceed $5\,cm$ to both robot sides (figure 4.12a). The value of D_s in the ND variants was set to "$0.7\,m$" in experiments 1 and 3 and to "$1.0\,m$" in experiment 2. According to the "weight strength" k in CG, it was set to "0.6", "0.3", "0.4" in experiments 1, 2, and 3, respectively. The translational and rotational speeds were limited to "$0.5\,m/s$" and "$1.0\,rad/s$" while carrying out all experiments.

It is important to mention that the ND variants as well as the SG method guide the robot towards a goal using a motion controller (called ND-controller) different from that introduced in section 4.2 (called TGF-controller). For the sake of a fair comparison, the presented experiments were performed using the ND-controller. A comparison of both motion controllers is then presented in chapter 5.

4.3.1 Experimental Results for ND+ and CG

The trajectories generated by ND+ in experiments 1 - 3 are shown in figures 4.10b, 4.11b, and 4.12b, respectively. The robot successfully passed through the

obstacle structure of experiments 1 and 2 in 56 s and 90 s, respectively. In the last experiment, the robot failed to reach the goal as it pushed over the obstacle labeled E after hitting/touching obstacles A, B and C. With the CG technique, the routes of the first and second experiments were traversed in 56 seconds and 86 seconds, respectively (see figures 4.10c and 4.11c). However, in experiment 2, the robot touched obstacle A and collided with the thin obstacles marked as C. In the last experiment, the mission was aborted after touching obstacle A, colliding with obstacle B, and finally overturning obstacle D (see figure 4.12c). It can be deduced from the generated trajectories that both methods (especially ND+) were prone to oscillations as a result of unreasonable deviations towards free regions and rapid changes in the direction of motion. For instance, see the trajectory while passing through the openings marked as P1 - P3 in figures 4.11b and 4.11c and the openings labeled P1 and P2 in figures 4.12b and 4.12c. The reason behind this behavior is the computation of the avoidance trajectory only based on the distance to threats (obstacles falling within the security zone) regardless of the location and field of view of the closest gap. This visualization has been supported by recording the translational and rotational speeds and plotting them against the elapsed time. Figures 4.10f, 4.10g, 4.11f, 4.11g, 4.12f, and 4.12g show these speeds using ND+ and CG for experiments 1 - 3, respectively[4].

4.3.2 Experimental Results for SG and TGF

By employing SG, GETbot was successfully navigated through the obstacle structure of experiments 1 - 3 in 48 seconds, 72 seconds, and 76 seconds, respectively (see figures 4.10d, 4.11d, and 4.12d). However, the robot moved close to the obstacles labeled A and B in figures 4.10d and 4.11d, and touched the obstacle marked as F in figure 4.12d. The speed profiles of experiments 1 - 3 are shown in figures 4.10h, 4.11h, and 4.12h, respectively. It can be deduced that the performance of the "SG method" is better than that of the ND+ and CG techniques. Despite this fact, however, rapid changes in the direction of motion can be seen along the trajectories generated by SG, especially at tight spaces. For example, while passing through passage P1 in experiment 3, GETbot smoothly traversed

[4]Videos of all experiments presented in this chapter are available at: "`http://getwww.uni-paderborn.de/research/videos/tgf-conf`"

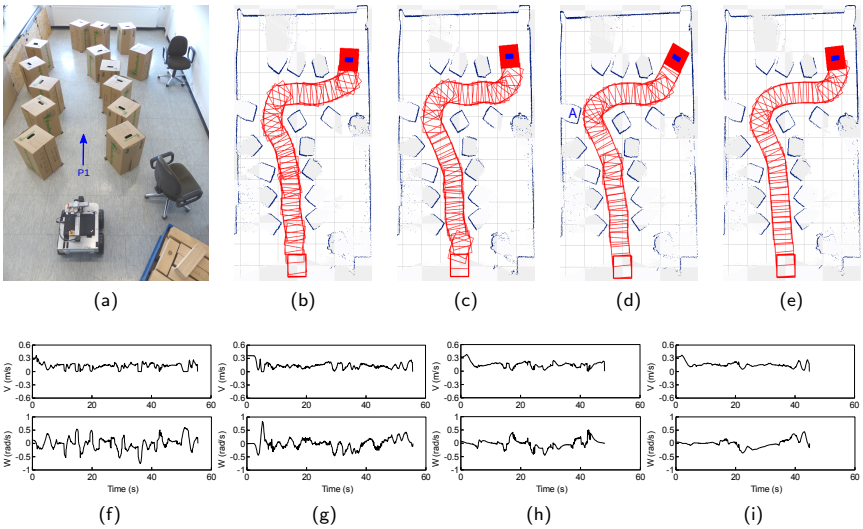


Figure 4.10: Test 1. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE).

the wide starting area, but it began to oscillate once the relatively tight end of P1 was reached. A similar behavior can be seen in experiments 1 and 2 (e.g. while passing through passages P1 and P3 in figures 4.10d and 4.11d, respectively). As pointed out in section 1, these rapid changes in the direction of motion is a result of creating the "safe gap" based only on the obstacle point closest to the robot. In narrow spaces, the location of this point varies rapidly over time and may frequently alternate between being on the right or left of the heading direction. The TGF method avoided this limitation by integrating the tangential and gap flow navigation concepts (Eq. (4.27)) and by computing Ψ_{vg} based on all threats belonging to $\mathcal{O}_{\text{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_t]$, while still emphasizing the available clearance to both sides of $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$ (Eq. (4.29)), the closest obstacle on each side (Eq. (4.30), Eq. (4.33)), and the location of the target (Eq. (4.28)) [MFM15]. The trajectories shown in figures 4.10e, 4.11e, and 4.12e, and the corresponding velocity profiles (figures 4.10i, 4.11i, and 4.12i) verify the improved stability and smooth-

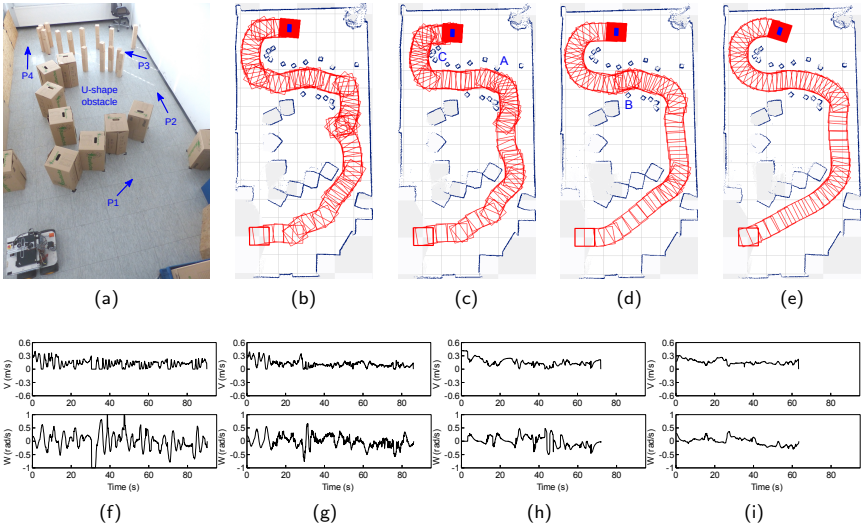


Figure 4.11: Test 2. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE).

ness of the proposed "TGF approach". Moreover, the time required to reach the goal was less than that corresponding to all discussed methods (45 seconds, 63 seconds, and 66 seconds in experiments 1 - 3, respectively).

4.4 Conclusions

This chapter presents the "Tangential Gap Flow" (TGF) navigation approach for reactive collision avoidance. TGF improves the navigation performance in narrow, unstructured, and cluttered environments. This is reflected by generating smoother and more stable avoidance maneuvers and by avoiding turn changes in tight gaps (occurs as a result of switching between circumnavigating/avoiding obstacles located to the right or left of the heading direction). This improvement is a result of computing the motion control based on the configuration of all obstacles

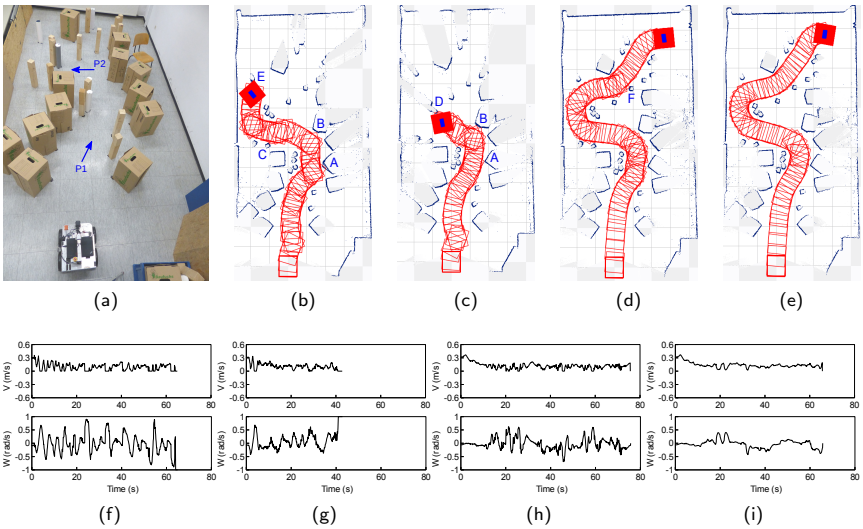


Figure 4.12: Test 3. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE).

causing collision with the "gap trajectory", while still accounting for the available clearance to obstacles on both sides of the heading direction [MFM15]. Performing the avoidance maneuver is based on two concepts; the "tangential" and "gap flow" navigation. A key idea of both concepts is to make use of the data extracted from the environmental structure in computing the avoidance maneuver. The "tangential navigation" drives the robot tangential to the obstacles boundary. With the "gap flow navigation" the robot safely and smoothly navigates among closely spaced obstacles. In both concepts, avoiding collisions and approaching the target are simultaneously performed. Finally, the motion commands, that drive a mobile robot towards a given target, is computed in such a way that the stability of the system is guaranteed in the Lyapunov sense [MFM16].

5 Evaluation of the Holonomic Solutions

We have seen how a mobile robot can be successfully driven through unknown dense environments by employing the "SG" and "TGF" approaches presented in chapters 3 and 4, respectively. It has been shown that both methods improve the robot's behavior when compared with the Nearness-Diagram (ND) Navigation variants, which are especially developed to accommodate these environments. It has also been shown that the trajectories generated by the TGF method, presented in chapter 4, are smoother and much more stable than those generated by the SG method, presented in chapter 3.

Several experiments were carried out and presented in chapter 3 where the differences in execution between the ND variants, the SG method, and the TGF approach have been subjectively discussed. In this chapter, we introduce a performance evaluation to quantitatively assess the effectiveness of the developed approaches over their counterparts. Moreover, we present additional experiments which were carried out using the TGF method and three ND variants; ND+ [MM04], SND [DB08], and CG [MFMJ10]. The TGF approach was selected as it outperforms the SG method and inherits its advantages.

The aim of these experiments (partially published in [MFM16]) is to demonstrate that the proposed "TGF approach" fulfills the major objective of this work: *to successfully guide an autonomous robot through unknown dense environments, while enhancing the efficiency (execution time and path length), smoothness, safety, and stability of the trajectories generated by state-of-the-art methods.* The experiments are presented in section 5.2 while the experimental setup is described in section 5.1. Section 5.3 introduces the performance metrics that are utilized to evaluate the execution of the proposed approaches. Finally, section 5.4 discusses and compares the behavior of all discussed methods.

5.1 Experimental Setup

The proposed methods were tested using our Pioneer 3-AT mobile robot GETbot, whose length and width are $0.52\,m$ and $0.48\,m$, respectively. GETbot works in a skid-steering mode and subject to nonholonomic constraints. A 2.6 GHz Intel core i5-3320M CPU laptop is placed on top of the robot, which fills in the role of a main controller. According to the sensing system, it consists of two laser scanners; one is placed on the front of the robot and the other is attached to the back. The front laser scanner is a Hokuyo UTM-30LX having a resolution of 0.25° and covering a range of $30\,m$ over 270° field of view. The rear laser scanner is a Hokuyo URG-04LX having a resolution of 0.35° and covering a range of $5.6\,m$ over 240° field of view. The information acquired by both laser scanners were fused, creating a virtual laser rangefinder having a full (360°) field of view. The maximum speeds of GETbot are $v = 0.7\,m/s$ and $\omega = 2.4\,rad/s$.

5.2 Experiments

This section presents seven experiments[1] performed in very dense scenarios where the robot did not have any previous knowledge about the obstacle distribution (Experiments 1 - 5 have been previously presented in [MFM16]). In order to compare the behavior of TGF to that of the ND techniques, each experiment (except experiment 7) was additionally performed using the implementation of ND+ [MM04], SND [DB08], and CG [MFMJ10]. While performing these experiments, the maximum robot speeds (v, ω) were restricted to ("$0.4\,m/s$", "$0.8\,rad/s$") in experiment 1 and to ("$0.5\,m/s$", "$1.0\,rad/s$") in the other experiments. The reason behind this restriction was the limited ability of the ND methods to generate a collision-free motion at high speed (see section 5.4). According to the D_s parameter of the ND variants, it was set to "$1\,m$". The CG approach has two additional parameters, D_{vs} and k. The value of D_{vs} was set to "$0.9\,m$", while k in experiments 1 - 6 was respectively set to "0.9", "0.5", "0.6", "0.8", "0.8", and "0.5". In order to determine the appropriate parameter value,

[1] Videos of these experiments can be found at: "`http://getwww.uni-paderborn.de/research/videos/tgf`"

each experiment was performed using various parameter settings. Notice that all tested methods are pure reactive, and hence they are prone to cyclic loops. To deal with this issue, gaps lying ahead of the robot were assigned higher priorities than those located to the rear. ROS [QCG+09] [ros19] was used as a middleware for implementation. For the sake of visualization, maps of the surroundings were created by employing an open source SLAM system [Gc10]. As mentioned in chapter 4, for having a fair comparison, the presented experiments were executed utilizing the ND-controller. Comparing the execution of the ND-controller to that of the TGF-controller is then presented in section 5.4.

5.2.1 Experiment 1

In this experiment, it was supposed to guide GETbot through an environmental structure made up of boxes, as depicted in figure 5.1a. The paths generated by all implemented techniques are visualized in figures 5.1b - 5.1e. Figure 5.1b demonstrates that ND+ suffered from rapid changes in the direction of motion, leading to oscillations (e.g. see the trajectory near the points marked as 1 - 7). By running SND and CG, GETbot got close to obstacles while navigating through tight spaces. For instance, see the path near the obstacles labeled A - C in figures 5.1c and 5.1d. By employing TGF, GETbot safely and smoothly traversed the obstacle course as shown in figure 5.1e. The translational and rotational speeds (v and ω) were recorded and plotted against the elapsed time. Figures 5.1f, 5.1g, 5.1h, and 5.1i show these speeds for ND+, SND, CG, and TGF, respectively.

5.2.2 Experiment 2

The environmental structure of this experiment consisted of different sized obstacles forming a large U-like shape, as shown in figure 5.2a. GETbot had to pass through an obstacle course including various narrow gaps where the available clearance at some locations did not exceed $10\,cm$ to both robot sides. By employing CG, GETbot moved close to obstacles like those labeled A - D in figure 5.2d. Furthermore, a touch with the obstacle marked as F occurred by using both ND+ and CG (see figures 5.2b and 5.2d). By applying SND, GETbot failed

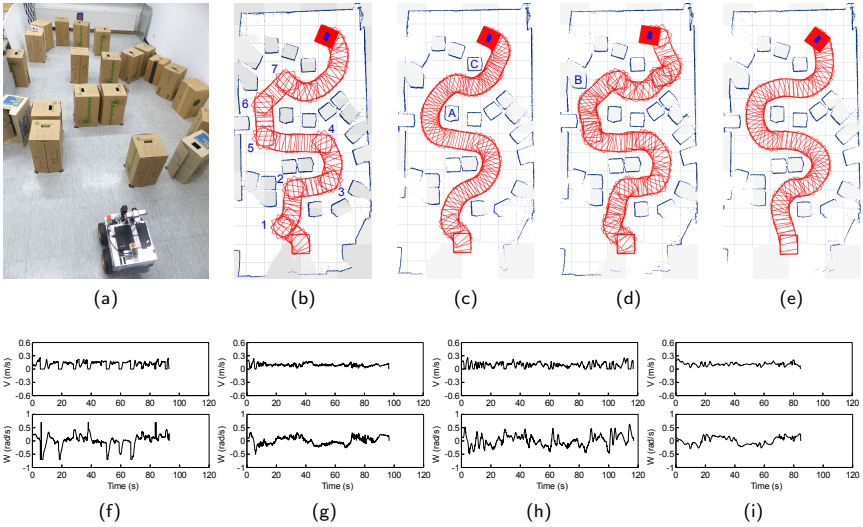


Figure 5.1: Scenario 1 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF.

to reach the goal as it collided with obstacle D and then ran over the tight gap formed by obstacles E and A (see figure 5.2c). Figures 5.2b - 5.2d demonstrate that the ND variants were prone to rapid changes in the direction of motion and unreasonable deviations towards free regions (e.g. see the points marked as 1 - 3). It can be deduced from figure 5.2e that TGF was able to drive GETbot with improved efficiency, safety, and smoothness compared the ND methods. This has been confirmed by plotting the velocity profiles in figures 5.2f - 5.2i.

5.2.3 Experiment 3

This experiment had three challenges (see figure 5.3a). First, the obstacle course consisted of several consecutive tight and curvy passages (e.g. P2, P3, and P4). Second, some tight gaps had one side with higher density of obstacles compared to

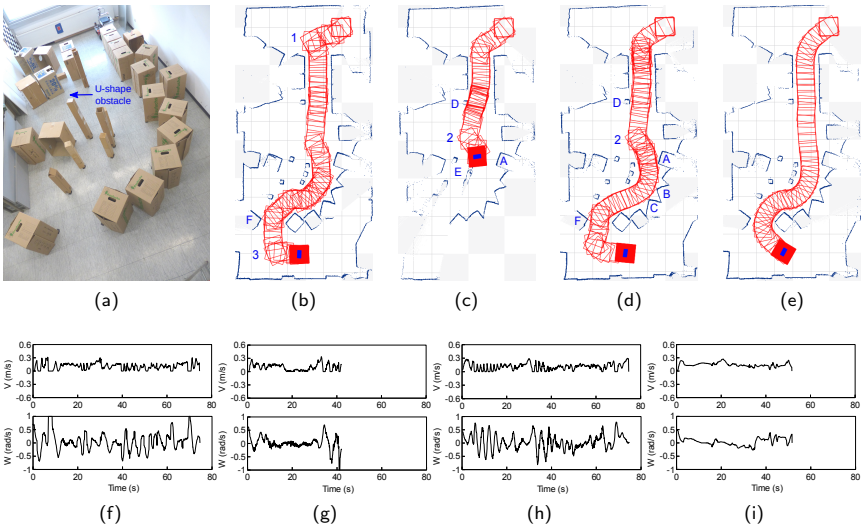


Figure 5.2: Scenario 2 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF.

the other (e.g. P1 and P2). Finally, the traversed route included many openings through which the robot did not fit (between each two obstacles). The goal was successfully reached by using both ND+ and TGF (figure 5.3b and figure 5.3e). However, with ND+ oscillations occurred along the traversed route and the chair marked as A was touched as can be seen from figure 5.3b. Moreover, the motion was unstable while traversing the first passage (marked as 1 in figure 5.3b). By employing CG and SND, GETbot pushed over the thin obstacles located to the right side of passage 1, coming to a full stop after 68 s and 26 s, respectively. Furthermore, the robot's movement was oscillatory and unstable as shown in figures 5.3c and 5.3d. Although the environment was very complex and contained several tight passages, the TGF approach managed to safely and smoothly guide GETbot towards the goal as shown in figure 5.3e. Figures 5.3f - 5.3i depict the recorded speeds plotted versus the elapsed time for all techniques.

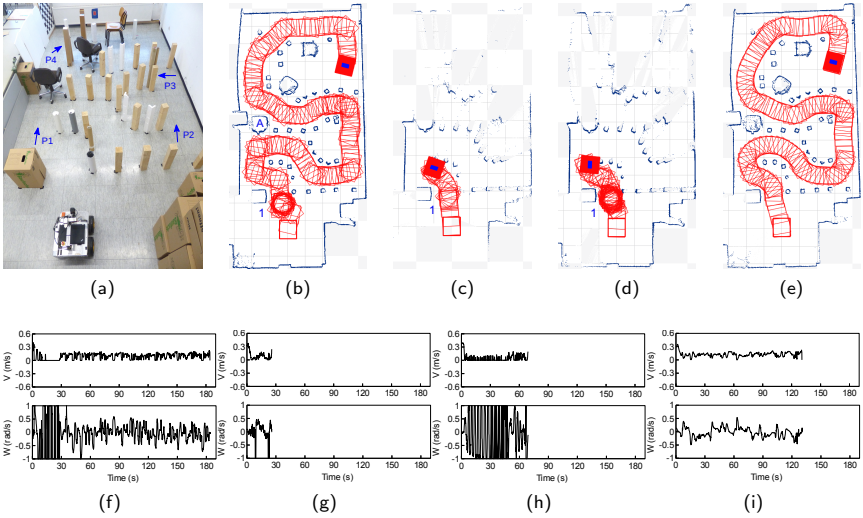


Figure 5.3: Scenario 3 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF.

5.2.4 Experiment 4

In this experiment, it was supposed to guide GETbot through two obstacle structures. Both structures were created in GET Lab; one in the entrance corridor (see figure 5.4a) and the other inside P 1.6.18, one of GET Lab rooms (see figure 5.4d). The major challenge in this experiment was the existence of consecutive passages with large difference in width between them (e.g. P1 and P2, P3 and P4). In such a situation, a considerable reduction in speed was necessary to achieve safe navigation while driving GETbot towards a relatively narrow passage, coming from a wide opening. An additional challenge was the presence of obstacles forming U-like shapes (e.g. see the large U-shape obstacle in figure 5.4d). The goal was reached by employing all techniques, as can be seen from figures 5.4b, 5.4c, 5.4e, and 5.4f. Similar to experiments 1 - 3, the ND methods were prone to rapid changes in the direction of motion (oscillations). For

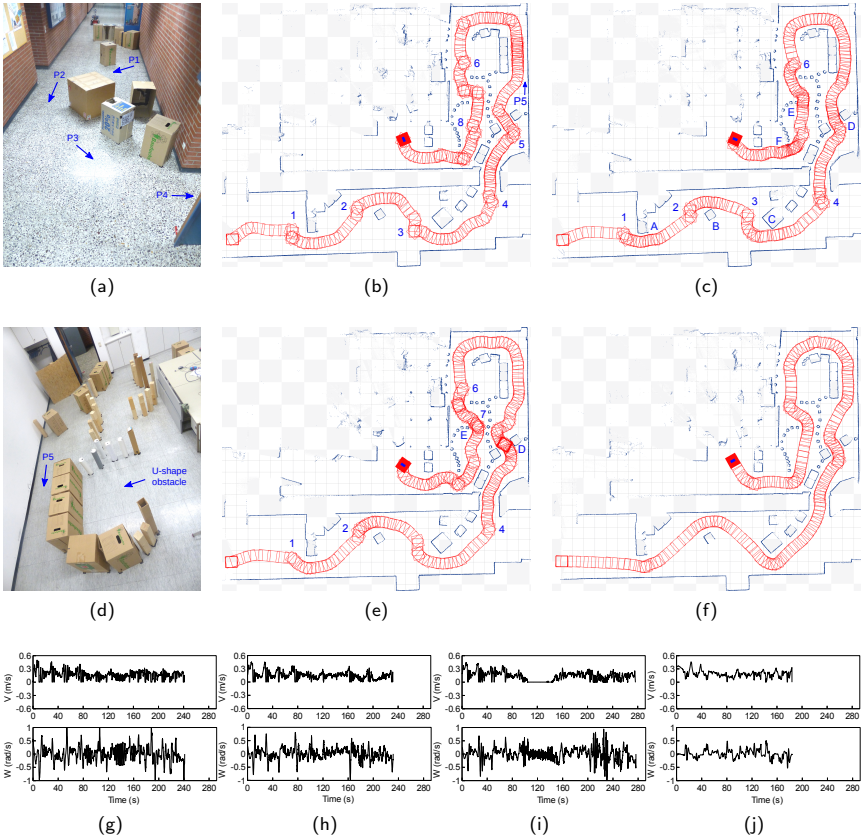


Figure 5.4: Scenario 4 (reprinted from [MFM16], with permission from Elsevier). (a, d) Environmental setup. (b, c, e, and f) Paths generated by (b) ND+, (c) SND, (e) CG, and (f) TGF. (g-j) Speed profiles for (g) ND+, (h) SND, (i) CG, and (j) TGF.

instance, see the generated paths near points 1 - 8 in figures 5.4b - 5.4e, as well as the large spikes in figures 5.4g - 5.4i. Moreover, by applying ND+, successive turn changes occurred while driving GETbot through the passage labeled P5, as can be deduced from figure 5.4b. By employing CG, GETbot got close to the obstacle marked as E and overturned obstacle D. With the SND technique, the

performance was worse as GETbot ran over the obstacles marked as D, E, and F and moved close to obstacles A - C. Figures 5.4f, and 5.4j demonstrate the improved performance of the proposed TGF method.

5.2.5 Experiment 5

The objective of this experiment was to test the behavior of TGF in the existence of moving objects. At the start of the mission, two boxes were located in front of the robot, as shown in figure 5.5a. At that moment, the gap labeled G1 was detected and selected to navigate through. Once the line marked as L was crossed, three boxes were pushed across the path towards G1. Notice that line L was approximately $50\,cm$ away from the boxes at that time (figure 5.5b). It was recognized that the path towards G1 was blocked and a new gap was created between the cupboard and the boxes, denoted as G2 in figure 5.5b. At that moment, the robot turned sharply to avoid collision with the boxes and then smoothly proceeded towards gap G2. The trajectory and the speed profile are shown in figures 5.5f and 5.5j, respectively. The ND variates were tested using the same setup. However, it was impossible to replicate the same scenario with each method, pushing the boxes at the exact time, velocity, and orientation. Furthermore, with each algorithm line L was reached with different velocity and heading. For example, by running ND+, SND, CG, and TGF, the robot's velocity was $0.11m/s$, $0.045m/s$, $0.033m/s$, and $0.21\,m/s$, respectively. Therefore, the performance of the tested methods cannot be fairly compared. Nevertheless, just to give an impression about the behavior of the ND variants, the generated paths and the speed profiles for ND+, SND, and CG are visualized in figures 5.5c - 5.5e and 5.5g - 5.5i, respectively. Although with TGF line L was reached at a speed higher than that of the ND variants, GETbot was able to avoid collision on time and smoothly proceed towards the goal.

5.2.6 Experiment 6

The route chosen for this experiment consisted of three obstacle courses as shown in figures 5.6a - 5.6c. The robot had to pass through the door labeled D1 in figure

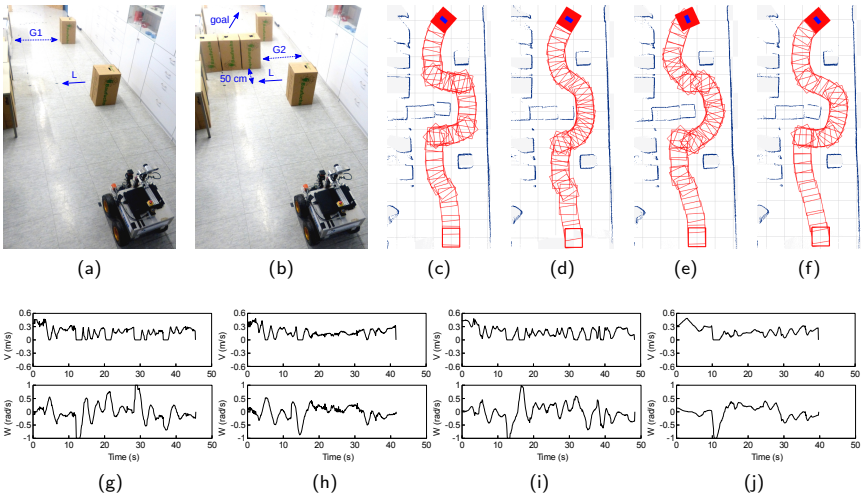


Figure 5.5: Scenario 5 (reprinted from [MFM16], with permission from Elsevier). (a, b) Environmental setup where (a) Depicts the start of the mission at which two boxes were located in front of the robot and (b) Mimics the moment at which three boxes were pushed towards the corridor once line L was crossed. (c-e) Paths generated by (c) ND+, (d) SND, and (e) CG, where oscillations in motion can be observed. (f) TGF avoided collision on time and smoothly proceeded towards the goal. (g-j) Speed profiles for (g) ND+, (h) SND, (i) CG, and (j) TGF.

5.6a, navigate towards the door marked as D2 in figure 5.6b, and finally reach the goal location shown in figure 5.6c. The obstacle courses were constructed in such a way that, at each time step, the robot may have multiple solutions (more than one navigable gap) and it had to decide which one to consider. For example, at the starting location, six navigable gaps were detected, the gaps labeled 1 - 6 in figure 5.6a. Similarly, at the locations labeled P1 and P2, the navigable gaps 7 - 9 and 10 - 12 were detected, see figures 5.6b and 5.6c. As expected, the robot selected the gap closest to the goal in each case (gaps 1, 7, and 12, respectively). The paths generated by ND+, SND, CG, and TGF are shown in figures 5.6d - 5.6g. Similar to the previous experiments, oscillation in motion can be observed along the paths taken by the ND variants (for instance, see the generated path

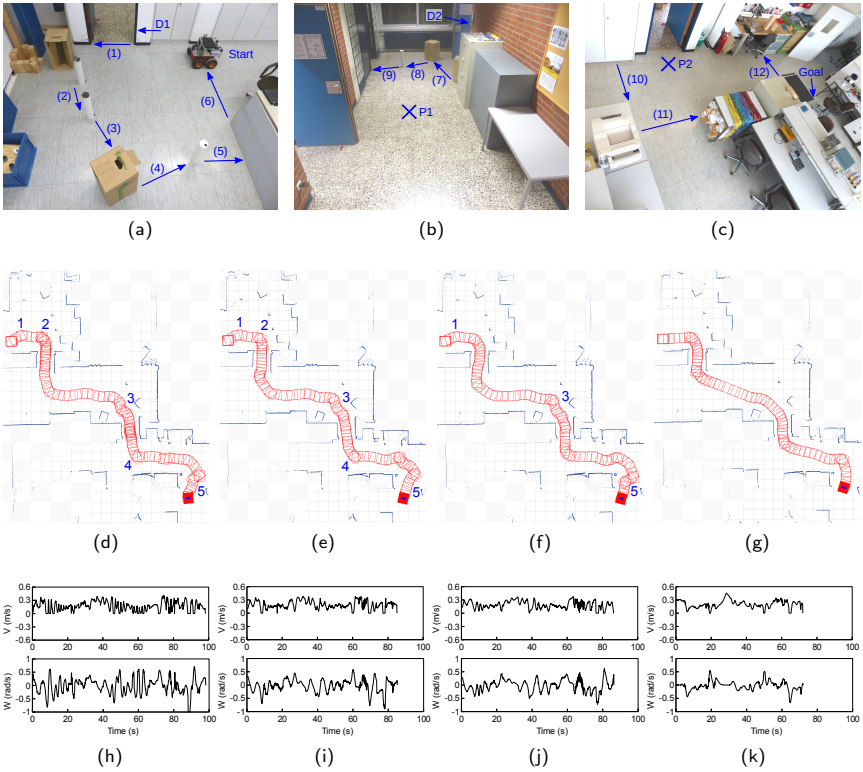


Figure 5.6: Scenario 6. (a-c) Environmental setup. (d-g) Paths generated by (d) ND+, (e) SND, (f) CG, and (g) TGF. (h-k) Speed profiles for (h) ND+, (i) SND, (j) CG, and (k) TGF.

at the points labeled 1 - 5 in figures 5.6d - 5.6f). We confirm our visualization by plotting the recorded velocities versus time in figures 5.6h - 5.6k.

5.2.7 Experiment 7

The objective of this experiment was to verify the capability of the TGF method to drive a mobile robot in unknown environments occupied by a crowd of moving

persons. The experiment was carried out in the ME building at the university of Paderborn[2]. This area is always crowded by students at the launch time. That is why we conducted the experiment at around 12:00 noon. The mission was started in front of the "Mensa" as shown in figure 5.7a. It was planned that the robot moves through the connection between buildings ME and B to reach the goal given near the elevator of building B. The connection and the location of the goal are shown in figures 5.7b and 5.7c, respectively. At the beginning of the mission, it was detected that the door labeled D1 in figure 5.7d is free, and therefore the robot moved directly towards it. Once the location shown in figure 5.7e is reached, some students wanted to pass through the door, closing the way of the robot. The situation was detected on-line, and therefore the robot decided to move towards the free area to the right-hand side, as shown in figure 5.7f. However, the students suddenly changed their decision and moved towards the door marked as D2 in figure 5.7g, freeing the D1 door. At that moment, the robot decided to pass through door D1 again, proceeding towards the free gap labeled G1 in figure 5.7g. In a similar situation, whenever the robot reached the location shown in figure 5.7h, a student closed the path planned across the door marked as D3. As result, gap G2 was selected to navigate through instead. However, the student kept moving, closing this gap as well, see figure 5.7i. At that moment, the robot escaped towards the free area to the left-hand side as shown in figure 5.7j. While moving towards that direction, another student stepped in front of the robot, leaving the D3 door behind him free, see figure 5.7k. Hence, the robot decided to pass through door D3 again, progressing towards the goal location as depicted in figure 5.7l. The trajectory followed by the robot and the velocity profiles are visualized in figure 5.8.

5.3 Performance Measures

Performance evaluation of robot motion planning techniques is a challenging issue, since assessing a method is often based on the application where it is being used. Furthermore, the robot motion is active, in the sense that the action

[2] A video of this experiment can be found at: `https://getwww.uni-paderborn.de/research/videos/dynamic-obstacles`

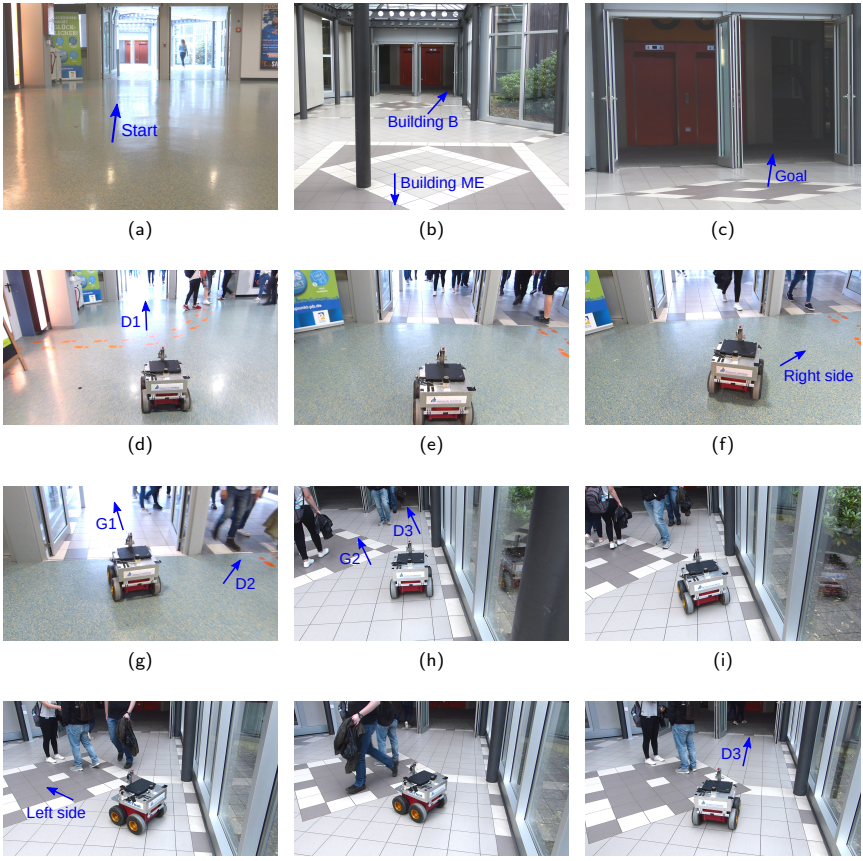


Figure 5.7: Scenario 7. (a-c) Environmental setup. (d-g) Snapshots of the experiment taken whenever the robot passed through door D1, at which several students closed the robot's path for a while. (h-i) Snapshots of the experiment show how the robot navigated through door D3.

performed affects the world state, e.g. the robot's configuration and speed. That is why it is not possible to produce benchmarking datasets of log-files, such as those used for testing localization and mapping algorithms [CN09].

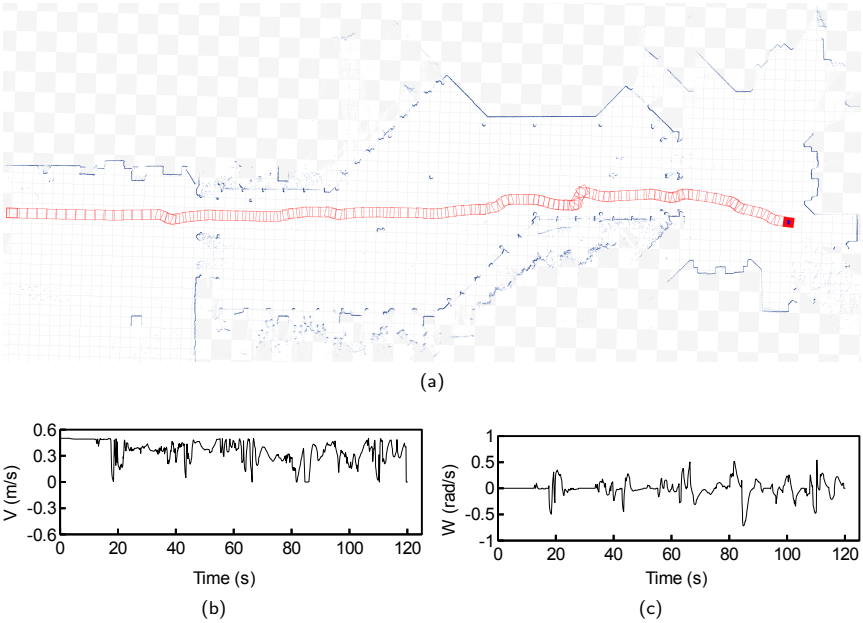


Figure 5.8: Visualization corresponding to Experiment 7. (a) Trajectory generated by the robot applying the TGF method. (c,d) Recorded motion commands plotted against the time elapsed.

Several aspects should be considered while evaluating a motion system, such as the robot's behavior along the trajectory, the time needed to accomplish the task, what kind of risk the robot faces during execution, etc. In the last years, some efforts have been devoted towards finding common performance metrics of a robotic navigation system (e.g. [GW03], [MVL07], [CN09], and [YAT10]). In general, those metrics can be classified into five groups: efficiency, oscillation, smoothness, physics-based, and security.

5.3.1 Efficiency Metrics

The efficiency or effectiveness of the vehicle is one of the most important measures to be considered when evaluating the performance of a motion system. It can be

reflected by the time and space dimensions of the trajectory. Loosely speaking, the least the zig-zag and curved motion, the higher the efficiency of the vehicle. The most commonly used metrics for assessing the efficiency of a navigation system are described in the following.

Total execution time $\mathrm{T_{tot}}\,[s]$: The total amount of time it takes a robot to fulfill a given task [CN09] (time to reach a target). For a better performance, it is desirable to have a low execution time.

Path length $\mathrm{P_{len}}\,[m]$: The total distance traveled by a robot from an initial location to a goal. This metric is useful for tasks in which the power consumption is of great concern. Apparently, a shorter path is preferable for achieving a better performance. Assume that a trajectory is given by $y = f(x)$ in the $X - Y$ plane and $(x_i, f(x_i)), (x_t, f(x_t))$ denote the initial and target locations, the total path length $\mathrm{P_{len}}$ is defined according to [MVL07]:

$$\mathrm{P_{len}} = \int_{x_i}^{x_t} (1 + (f'(x))^2)^{\frac{1}{2}}\, dx \tag{5.1}$$

where $f'(x)$ is the derivative of $f(x)$ with respect to x.

5.3.2 Oscillation Metrics

The measures introduced in this and in the following sections, reflect the robot's behavior during the whole task execution. In particular, this section presents the metrics that are helpful to detect oscillations along the trajectory. As we have pointed out in chapters 3 and 4, oscillations may occur as a result of successive turn changes while passing through narrow passages or due to deflections towards free areas. Notice that these metrics also provide a measure of smoothness, since the less the oscillations the smoother the trajectory.

Curvature change $\mathrm{C_{chg}}\,[rad/m]$: The curvature change is useful for detecting oscillations and unstable motion along the trajectory. Given the linear and angular robot velocities ($v(t)$, $\omega(t)$), the curvature can be defined as [CN09]:

$$k(t) = \left| \frac{\omega(t)}{v(t)} \right| \tag{5.2}$$

The curvature change is then computed as:

$$C_{chg} = \int_0^{T_{tot}} \left| k'(t) \right| \, dt \tag{5.3}$$

The lower the value of C_{chg}, the less oscillations occur.

For a better comparison, an average curvature change is computed by dividing the value of C_{chg} by the total execution time T_{tot}.

Number of zero crossings along the curve of rotational speed Z_ω: This metric, that we proposed in [MFM16], measures the amount of variations in the heading direction (number of times a robot turns from one direction to another). Hence, it gives an impression about the degree of oscillations in a trajectory. Having less Z_ω is desirable and indicates more stable control commands.

5.3.3 Smoothness Metrics

The smoothness of a trajectory reflects the energy and time consumption and can be related to the system integrity [CN09]. It also indicates the consistency in decision-making and shows the ability of the system to anticipate events [Ros97]. Moreover, a robot that navigates in a smooth way is socially more acceptable, particularly if it shares spaces with humans. In order to measure the smoothness of a trajectory, we use the following two measures.

Accumulated linear and rotational jerks, $J_{acc}\,[m^2/s^5]$ and $\zeta_{acc}\,[rad^2/s^5]$: Jerk (defined as the time derivative of acceleration) is correlated with abrupt variations in the actuator forces [Fre12]. Therefore, it is possible to quantify smoothness in both steering and speed as a function of jerk [Ros97].

Given the linear and rotational robot velocities ($v(t)$, $\omega(t)$), we have defined the accumulated jerk costs (linear and rotational) in [MFM16] as:

$$J_{acc} = \frac{1}{T_{tot}} \int_0^{T_{tot}} \left[\ddot{v}(t) \right]^2 dt \tag{5.4}$$

$$\zeta_{acc} = \frac{1}{T_{tot}} \int_0^{T_{tot}} \left[\ddot{\omega}(t) \right]^2 dt \tag{5.5}$$

For a smoother behavior, trajectories that minimize both the linear and rotational jerk costs are desirable.

5.3.4 Physics-based Metrics

In this section, we describe the metrics that are related to the stability of the motion and the dynamics of the vehicle itself. Therefore, they are useful for evaluating the performance of those systems where the dynamics of the robot have to be considered. Basically, These metrics are defined in terms of the inertial forces acting on the robot along the trajectory. However, they consider a unit-mass robot, i.e. the mass of the robot is neglected. In the following, we present the most commonly used physics-based metrics.

Lateral stress $\mathrm{S_{lat}}$ $[N.s]$: With the lateral stress, the centrifugal force acing on the robot is directly measured. The centrifugal force affects the stability of the robot and can lead to lateral wheel skidding (due to a curved motion). Therefore, it is desirable to have a lower $\mathrm{S_{lat}}$ value. In general, the straighter the generated path, the lower the value of $\mathrm{S_{lat}}$.

The lateral stress is computed by integrating the centrifugal force along the trajectory [CN09]:

$$\mathrm{S_{lat}} = \int_0^{\mathrm{T_{tot}}} \frac{v(t)^2}{r(t)}\, dt \tag{5.6}$$

where $r(t)$ is the instantaneous curvature radius (i.e. $r(t) = 1/k(t)$); the reciprocal of the curvature .

Tangential stress $\mathrm{S_{tng}}$ $[N.s]$: Similar to the lateral stress, $\mathrm{S_{tng}}$ $[N.s]$ is directly associated with the robot dynamics. It is useful to detect abrupt speed changes (sudden acceleration and braking) which may cause slipping while turning. The value of $\mathrm{S_{tng}}$ is computed as follows [CN09]:

$$\mathrm{S_{tng}} = \int_0^{\mathrm{T_{tot}}} |a(t)|\, dt \tag{5.7}$$

where $a(t)$ is the instantaneous linear acceleration ($a(t) = \dot{v}(t)$).

Having a lower $\mathrm{S_{tng}}$ is preferable for achieving a more stable robot motion.

5.3.5 Security Metrics

The metrics discussed in this section describe the hazards along the trajectory in terms of the distance between the robot and obstacles. Notice that there is a trade-off between path length and security, hence it is up to the algorithm to decide how far the robot stays away from obstacles

Obstacles risk $\mathrm{R_{obs}}\,[m^{-1}]$: This metric is concerned with measuring the risk of obstacles, reflected by their proximity to the robot across the entire task. Let $r_{\min}(t)$ represents the distance between the robot and the closest obstacle at time t, $\mathrm{R_{obs}}$ can be expressed as [CN09]:

$$\mathrm{R_{obs}} = \int_0^{\mathrm{T_{tot}}} \frac{1}{r_{\min}(t)}\, dt \tag{5.8}$$

Collisions count $\mathrm{N_{col}}$: The number of collisions is counted per task [Min08]. A safe motion requires a collision-free operation along the entire trajectory.

It is worth to mention that while computing $\mathrm{C_{chg}}$, $\mathrm{S_{lat}}$ or $\mathrm{R_{obs}}$, a division by zero may occur. This problem is avoided by adding a small value ϵ to the denominator in Eqs. (5.3), (5.6), and (5.8). In our analysis, ϵ was set to 0.001.

5.4 Evaluation and Discussion

The aforementioned measures were employed to evaluate the performance of the "TGF approach" and to compare its behavior to that of the ND variants. Experiments 1 - 6, presented in section 5.2, yield the results shown in table 5.1. It can be observed that the proposed "TGF approach" presented the best results in terms of all metrics considered. Significant differences in execution can be observed looking at the oscillation and smoothness metrics ($\mathrm{C_{chg}}$, $\mathrm{Z_{\omega}}$, $\mathrm{J_{acc}}$, and ζ_{acc}). However, in the first scenario, the lateral stress ($\mathrm{S_{lat}}$) and risk ($\mathrm{R_{obs}}$) metrics associated with the ND+ method were slightly better than those associated with TGF. This can be explained by the tendency of ND+ to drive the robot across the center of openings, regardless of their width. Such a behavior maximizes the clearance to obstacles, but at the same time increases the distance

Table 5.1: Performance assessment of the proposed TGF approach for experiments 1 - 6, presented in section 5.2, using the metrics defined in sections 5.3.1 - 5.3.5 (results of experiments 1 - 4 are reprinted from [MFM16] with permission from Elsevier).

Exp.	Method	T_{tot}	P_{len}	C_{chg}	Z_{ω}	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}	N_{col}
1	ND+	93	10.27	128.25	40	1.42	18.01	**0.84**	8.16	**515.87**	0
	SND	97	**8.75**	13.53	46	0.59	4.21	0.99	6.73	1002.15	0
	CG	117	10.56	121.83	50	0.61	2.84	1.20	11.02	832.02	0
	TGF	**85**	8.96	**1.60**	**10**	**0.13**	**0.46**	1.05	**3.49**	579.70	0
2	ND+	75	8.19	186.37	52	1.95	15.49	1.11	9.96	669.48	1
	SND	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
	CG	75	8.04	213.26	54	1.43	9.29	1.00	10.14	924.68	1
	TGF	**52**	**7.17**	**2.17**	**8**	**0.18**	**0.86**	**0.75**	**2.53**	**445.84**	0
3	ND+	184	16.39	160.01	126	1.55	105.54	2.06	18.42	2635.02	1
	SND	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
	CG	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
	TGF	**131**	**15.19**	**12.35**	**20**	**0.23**	**1.66**	**1.78**	**6.83**	**1478.25**	0
4	ND+	241	36.80	131.90	121	1.50	8.71	4.91	34.41	1193.48	0
	SND	232	34.10	82.56	143	0.85	6.41	3.97	24.69	2980.42	3
	CG	277	36.04	143.74	215	0.84	9.61	4.49	27.54	41404.00	1
	TGF	**184**	**33.72**	**9.41**	**38**	**0.26**	**1.07**	**2.72**	**11.09**	**1008.60**	0
5	ND+	45	7.66	232.34	20	3.00	10.15	0.98	8.29	170.81	0
	SND	41	**6.95**	55.16	31	1.30	11.13	**0.89**	5.51	199.36	3
	CG	48	7.73	166.53	19	1.75	5.44	1.05	8.14	193.36	1
	TGF	**39**	7.52	**54.41**	**5**	**0.69**	**1.76**	0.94	**3.74**	**158.54**	0
6	ND+	98	16.51	169.79	52	2.99	9.12	2.28	18.16	475.86	0
	SND	85	16.17	71.45	56	2.03	7.75	2.05	11.85	412.28	0
	CG	86	16.24	61.40	36	1.80	7.08	2.12	11.30	450.68	0
	TGF	**72**	**14.16**	**29.28**	**26**	**0.57**	**2.05**	**1.19**	**6.14**	**411.97**	0

traveled (P_{len}). Additionally, this results in a trajectory consisting of successive small segments connected by sharp corners, significantly increasing the curvature change and jerk costs (see figure 5.1b). Another observation from experiments 1 and 5 is that the trajectories created by SND have lower P_{len} and S_{lat} values than those generated by TGF. This is a result of driving the robot close to objects (figures 5.1c and 5.5d). Such a behavior reduces the path length (lower P_{len}) and curvature (lower S_{lat}), but at the same time highly increases the value of R_{obs}.

The presented experiments show that all ND variants have in common the drawback of experiencing oscillations at passages that suddenly get narrower or at sharp-turning maneuvers [MFM16]. As an example, see the location marked as 1 in figures 5.3b - 5.3d and the locations labeled 1 - 8 in figures 5.4b, 5.4c, and

5.4e. The abrupt variation in the width of a passage results in a severe reduction in the robot's velocity (may approach zero) followed by a sharp turn. Due to the speed gained before performing the turn, the robot may experience oscillations and wheel slippage. This is reflected by the values of C_{chg}, Z_{ω}, ζ_{acc}, and S_{tng}. We attribute this behavior to the usage of the Artificial Potential Field concept in performing the avoidance maneuver, regardless of the location and field of view of the traversed opening. In this regard, approaching one side of a tight opening generates a strong repulsion force, causing a sharp turn that takes the robot away from obstacles. Apparently, this behavior is repeated with the other side. Additionally, whenever the robot faces a relatively wide area located between two narrow passages, it tends to deviate towards the free region performing a sharp turn, followed by a another sharp turn towards the opposite side trying to enter the second narrow passage (e.g. see the trajectories near point 6 in figures 5.4b - 5.4e). These sudden turn changes may lead to oscillations and instability, which can be unsafe if the robot is passing through a narrow passage or if it is navigating at a relatively high speed. Furthermore, the time required to reach the goal increases as turn maneuvers are executed at lower speeds [MFM16].

It is worth to compare the discussed ND variants with each other. It can be seen that ND+ is safer and may negotiate more difficult scenarios. It managed to guide the robot through all obstacle courses with lower risk value and fewer collisions. Nevertheless, SND and CG are better in terms of smoothness. See, for instance, the path while navigating through passage P5 in experiment 4. This can also be deduced from the corresponding J_{acc} and ζ_{acc} values. CG and SND seem to be more sensitive to the obstacle distribution [MFM16]. For instance, whenever the SND-controlled robot passes through a tight gap having more threats on one side compared to the other, it usually approaches the side having the least number of threats (e.g. A and C in figure 5.1c, A - C in figure 5.4c), experiences collisions (e.g. D - F in figure 5.4c), or fails to reach the goal (e.g. experiments 2 and 3). By respecting the percentage of threats on each side, CG was able to drive GETbot towards the goal in experiment 2, but with more oscillations (larger C_{chg}). Furthermore, it experienced collisions as in experiment 3.

Another observation is that the motion behavior of TGF appears more human-like compared to the ND variants. In [MFM16], we have explained this claim by

describing the reaction to threats falling in between the robot and the closest gap: "Using the TGF method, each obstacle generates an avoidance angle which points directly towards the closest gap or parallel to the obstacle in the direction closer to the gap. Hence, all these obstacles contribute in heading the robot towards the closest gap. By using the ND variants, on the other hand, each obstacle point falling within D_s (ND+ only considers the closest two, whereas SND and CG consider all) causes an avoidance angle which points directly away from the obstacle regardless of the direction towards the closest gap. This avoidance angle is weighted by the relative proximity of the obstacles in the SND and CG methods (each one uses different weights). The direction of motion is then adjusted by the total averaged (ND+) or weighted (SND, CG) avoidance angle".

In order to demonstrate the capability of TGF to safely drive the robot at higher speeds, experiments 3 and 4 were performed again using higher speed limits $(0.7\,m/s, 1.3\,rad/s)$. Notice that $0.7\,m/s$ is the maximum possible linear velocity in our robot. Both experiments were carried out using the implementation of the TGF and ND+ methods[3]. Among the ND variants, ND+ was selected as it managed to negotiate harder scenarios. Also, the experimental setup of experiments 3 and 4 were selected as they were challenging and had several difficulties as discussed in section 5.2. With the ND+ method, GETbot failed to reach the goal in experiment 3 and the mission was aborted after $85\,s$. This is due to the fact that GETbot collided with the obstacles marked as A - C and finally overturned obstacle D as depicted in figures 5.9a and 5.9e. By employing TGF, the goal was successfully reached in $105\,s$ only (see figures 5.9b and 5.9f). In experiment 4, ND+ was unable to drive GETbot towards the goal as well. This is owing to the fact that it overturned the obstacle labeled A and the mission was aborted after $241\,s$ (see figures 5.9c and 5.9g). Furthermore, ND+ was prone to oscillations and instability as can depicted from the generated trajectory near points 1 - 4 (see figure 5.9c) Using TGF, GETbot successfully passed through the obstacle course and the goal was reached in $140\,s$ (figures 5.9d and 5.9h).

It is worth to differentiate the execution of TGF to that of the SG method. For this purpose, we make use of experiments 1 - 3 from chapter 4, which have been

[3] Videos of these experiments can be found at: "`http://getwww.uni-paderborn.de/research/videos/tgf2`"

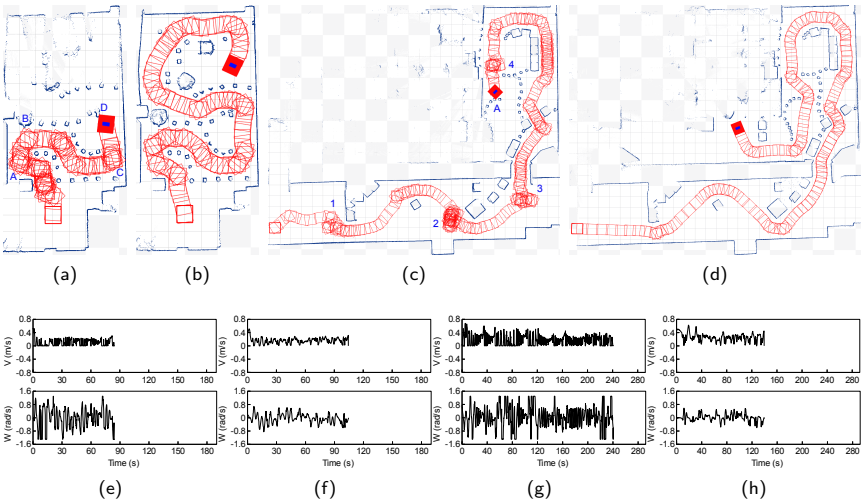


Figure 5.9: Scenario 3 and 4 from section 5.2, but with $(0.7\,m/s, 1.3\,rad/s)$ speed limits (reprinted from [MFM16], with permission from Elsevier). (a, b) Paths generated in experiment 3 using (a) ND+ and (b) TGF. (c, d) Paths generated in experiment 4 using (c) ND+ and (d) TGF. (e-h) Speed profiles corresponding to the paths shown in (a-d), respectively.

conducted using the implementation of ND+, CG, SG, and TGF. Table 5.2 shows the performance of all methods, where it can be deduced that the results of the TGF approach and both ND variants are roughly similar to those obtained from the experiments presented in section 5.1 and discussed above. It is also obvious that the TGF approach outperforms the SG method in all metrics, establishing a much more stable and reliable navigation approach. In particular, significant performance improvements can be seen looking at the oscillation and smoothness metrics. We believe that this improvement is the result of integrating the "tangential" and "gap flow" concepts and determining the avoidance trajectory based on all surrounding obstacles. Additionally, one can notice from figures 4.10 - 4.12 and from table 5.2 that the narrower the passages the bigger the difference in performance between the TGF and SG methods. For example, the biggest difference in performance between both methods occurs in experiment 3 (figure

Table 5.2: Performance evaluation results for the experiments presented in section 4.3 from chapter 4, using the metrics defined in sections 5.3.1 - 5.3.5.

Exp.	Method	T_{tot}	P_{len}	C_{chg}	Z_{ω}	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}	N_{col}
1	ND+	55	6.59	134.65	36	1.41	10.46	0.81	6.39	537.35	0
	CG	56	6.79	34.30	42	0.71	4.76	0.75	5.46	588.85	0
	SG	48	6.57	18.67	22	0.58	4.48	0.54	3.54	647.58	0
	TGF	**45**	**6.48**	**1.66**	**11**	**0.21**	**0.87**	**0.58**	**2.59**	**451.06**	0
2	ND+	90	10.80	208.45	45	1.91	13.07	1.56	14.07	634.37	0
	CG	86	9.88	70.37	83	1.06	10.72	1.26	9.78	1378.00	2
	SG	72	10.10	16.15	21	0.67	9.76	1.16	5.02	**573.06**	0
	TGF	**63**	**9.83**	**3.91**	**7**	**0.15**	**0.59**	**1.06**	**4.98**	594.26	0
3	ND+	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
	CG	fail	fail	fail	fail	fail	fail	fail	fail	fail	fail
	SG	76	8.45	80.28	44	0.94	12.26	0.91	7.00	1029.60	1
	TGF	**66**	**8.26**	**10.07**	**12**	**0.10**	**0.52**	**0.77**	**6.28**	**770.29**	0

4.12), since the environment is composed of very narrow passages. Finally, when comparing the performance of the SG approach with that of the ND+ and CG methods, it becomes obvious that the SG method achieves the best results.

As we have pointed out in sections 4.3 and 5.2, the presented experiments were performed using the ND-controller. It is important to demonstrate the effectiveness of the TGF-controller (proposed in section 4.2) on the performance of the robot. For this purpose, experiments 3 and 4 from section 5.2 and experiments 2 and 3 from section 4.3 were repeated, but using the TGF-controller[4]. These experiments were carried out using two different speed limits: firstly, by setting $v_{\max}$ and $\omega_{\max}$ to 0.5 m/s and 1 rad/s, similar to the original setup, and then by limiting them to $v_{\max} = 0.4\,m/s$ and $\omega_{\max} = 0.8\,rad/s$. Using both speed limits, the time needed to reach the goal was less than that of the ND-controller. Figures 5.10 and 5.11 show the paths generated and the speed profiles. The performance assessment is also shown in tables 5.3 and 5.4. It can be seen that the TGF-controller outperforms the ND-controller in terms of T_{tot}, C_{chg}, J_{acc}, and S_{tng}. As we reported in [MFM16], this improved performance is indeed a result of directly deriving the TGF-controller from the robot's kinematic model in such away that stability of the system is guaranteed in the Lyapunov sense. Comparing the other performance measures does not present significant differences.

[4]Videos of experiments 3 and 4 from section 5.2 using the TGF-controller can be found at: "`http://getwww.uni-paderborn.de/research/videos/tgf3`"

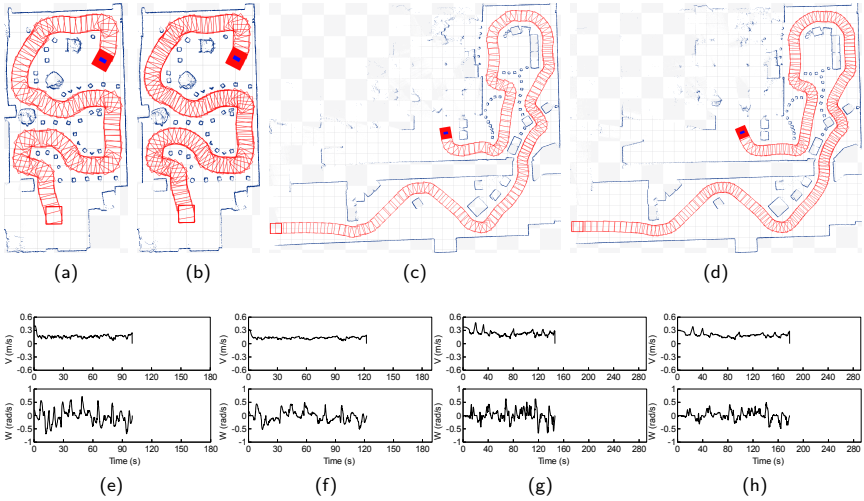


Figure 5.10: Scenarios 3 and 4 from section 5.2 running TGF, but using the TGF-controller (reprinted from [MFM16] with permission from Elsevier). (a, c) Paths generated in (a) Scenario 3 and (c) Scenario 4, using $(0.5\,m/s, 1\,rad/s)$ speed limits . (b, d) Paths generated in (b) Scenario 3 and (d) Scenario 4, using $(0.4\,m/s, 0.8\,rad/s)$ speed limits. (e-h) Speed profiles corresponding to the paths shown in (a-d).

Table 5.3: Performance assessment of the TGF-controller for scenarios 3 and 4 from section 5.2 (reprinted from [MFM16] with permission from Elsevier). As a reference, the results of the ND-controller from table 5.1 are listed, too.

Exp.	Speeds	T_{tot}	P_{len}	C_{chg}	Z_{ω}	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}	N_{col}
3 (ND-controller)	(0.5, 1.0)	131	15.19	12.35	20	0.23	1.66	1.78	6.83	1478.25	0
3 (TGF-controller)	(0.5, 1.0)	100	15.65	1.67	21	0.06	2.47	3.27	3.62	1119.05	0
3 (TGF-controller)	(0.4, 0.8)	122	15.46	1.18	7	0.03	1.18	2.42	2.77	1340.76	0
4 (ND-controller)	(0.5, 1.0)	184	33.72	9.41	38	0.26	1.07	2.72	11.09	1008.60	0
4 (TGF-controller)	(0.5, 1.0)	146	34.05	0.91	30	0.07	2.30	4.59	4.7	803.57	0
4 (TGF-controller)	(0.4, 0.8)	178	33.73	0.62	20	0.02	0.88	3.45	3.42	885.50	0

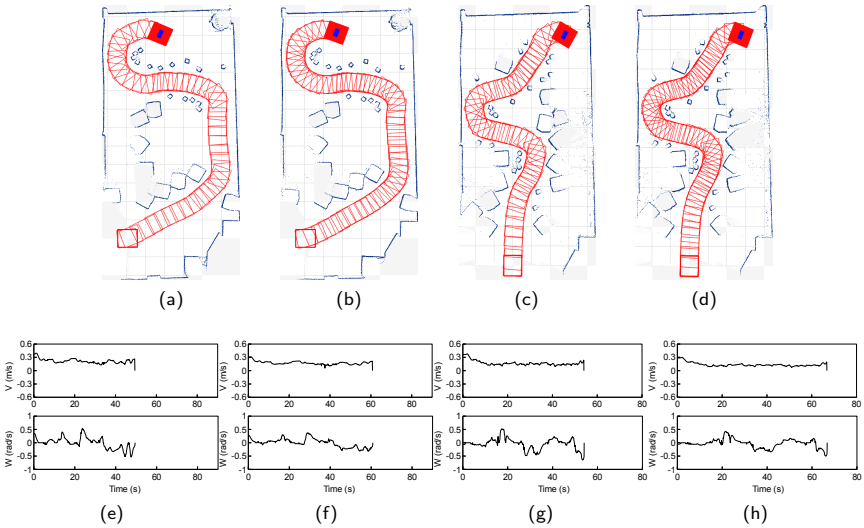


Figure 5.11: Scenarios 2 and 3 from section 4.3 running TGF, but using the TGF-controller. (a, c) Paths generated in (a) Scenario 2 and (c) Scenario 3 using $(0.5\,m/s, 1\,rad/s)$ speed limits. (b, d) Paths generated in (b) Scenario 2 and (d) Scenario 3 using $(0.4\,m/s, 0.8\,rad/s)$ speed limits. (e-h) Speed profiles corresponding to the paths shown in (a-d).

Table 5.4: Performance assessment of the TGF-controller for scenarios 2 and 3 from section 4.3. As a reference, the results of the ND-controller from table 5.2 are listed, too.

Exp.	Speeds	T_{tot}	P_{len}	C_{chg}	Z_{ω}	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}	N_{col}
2 (ND-controller)	(0.5, 1.0)	63	9.83	3.91	7	0.15	0.59	1.06	4.98	594.26	0
2 (TGF-controller)	(0.5, 1.0)	49	9.96	0.78	5	0.06	1.00	1.64	1.77	376.97	0
2 (TGF-controller)	(0.4, 0.8)	60	9.88	0.52	5	0.03	0.44	1.29	1.43	459.19	0
3 (ND-controller)	(0.5, 1.0)	66	8.26	10.07	12	0.10	0.52	0.77	6.28	770.29	0
3 (TGF-controller)	(0.5, 1.0)	54	8.52	1.04	10	0.07	1.09	1.28	2.00	693.57	0
3 (TGF-controller)	(0.4, 0.8)	66	8.47	0.85	12	0.03	0.41	1.02	1.58	819.72	0

6 Under-constrained Reactive Collision Avoidance Navigation

This chapter introduces a novel obstacle avoidance approach for mobile robots that must perform in highly cluttered environments. The previous chapters illustrated the limitations of existing techniques when it comes to guiding robots through narrow gaps in highly cluttered environments. In [MFM13b] [MFM13a] [MJFM13] [MFM15] [MFM16] [MFM17] we have discussed a group of methods that, at first sight, seem well equipped to deal with such difficult scenarios. Their commonality is the employment of some sort of high-level information description of the environmental structure. In particular, they include the SG and TGF approaches, as described in chapters 3 and 4, as well as the Nearness-Diagram (ND) techniques [MM04] [DB08] [MFMJ10]. However, these methods (called here *gap-based* methods) rely on a strong assumption which may not be valid in real-world scenarios; they assume a holonomic disc shaped robot. This ignores the actual robot shape and its kinematics, which may hinder finding feasible motions or lead to collisions. Considering these constraints is especially critical for robots operating in highly cluttered environments [MM17] [MFM18].

The drawbacks mentioned above have been addressed in the literature by making use of the holonomic solution to generate motion commands that respect the robot's shape and kinematics. Bemporad et al. uses a least squares optimization to align the robot's orientation with the holonomic solution [BLOD96], whereas Minguez and Montano decompose the problem into motion, shape, and kinematics which are independently solved [MM02]. Because these solutions are based on discretization of the problem and rely on approximations, such methods present limited capabilities in environments requiring high maneuverability. A more general methodology has been introduced in [MM09] by transforming the workspace

into ARM, an "Arc Reachable Manifold" that implicitly considers the robot's shape and kinematic constraints. Within this method, robots travel along paths made up of circular segments. While this may provide a smoother motion than previously mentioned methods, some problems may arise: not all gaps that are in principle navigable may be reached via such arcs[1] and the involved coordinate transformations make it difficult to search for openings in the first place, which presents a serious challenge for gap-based navigation techniques [MFM18].

In summary, all existing obstacle avoidance methods either ignore the vehicle constraints or have limitations in cluttered environments. To deal with this drawback, in [MM17] [MM16] [MFM18] we have introduced the concept of an "admissible gap" (AG), which will be discussed in the remainder of this chapter.

A gap is called "admissible" if it is traversable by performing a single motion command that respects both the shape and kinematic constraints. By employing this concept, a new collision avoidance method, abbreviated as AG, has been developed and implemented. AG avoids the above mentioned limitations of existing gap-based methods, while still being applicable for highly cluttered environments. This has been possible by directly respecting the vehicle constraints rather than adapting a holonomic-based solution. The overall approach works as follows: similar to the holonomic solutions proposed in chapters 3 and 4, the sensor data is searched for the most promising opening to navigate through. Once determined, an "admissible gap" is constructed in an iterative manner, which serves as a bridge to the specified opening and provides a compromise between safety and efficiency. Our approach is directly applied to the workspace without having to construct an abstraction layer. Another important contribution of AG is the development of a new procedure for finding out gaps. The method can be applied to full or limited field of view sensors. Moreover, it discards useless gaps, hence reducing oscillations. Outstanding results have been achieved in cluttered environments, where the AG approach outperforms existing state-of-the-art methods in terms of smoothness, safety, efficiency, and robustness [MM17] [MFM18].

This chapter is organized as follows. In section 6.1, some preliminary definitions are presented. Section 6.2 describes our strategy of finding out gaps, and sub-

[1] By transforming the workspace into ARM, a gap is navigable only if it coincides with the circular path that goes through the robot's origin and tangential to its heading [MFM18].

sequently, section 6.3 introduces the concept of "admissible gap". In section 6.4, we show how this concept is employed to develop a navigation approach. Section 6.5 discusses the experimental results, while section 6.6 evaluates the execution of AG. Finally, we point out some concluding remarks in section 6.7.

6.1 Preliminary Definitions

The following definitions and assumptions are used to explain the AG approach. Notice that the definitions introduced in section 3.1.1 are also used here.

As mentioned in section 3.1.1, the list of scan points is denoted by $S = \left\{\mathbf{p}_1^{\mathrm{S}}, ..., \mathbf{p}_n^{\mathrm{S}}\right\}$, where $r_{\max}$ denotes the maximum range of the sensor.

The rank of an element in S defines its relative location. For instance, the obstacle point $\mathbf{p}_i^{\mathrm{S}}$ is located to the *right* of $\mathbf{p}_{i+1}^{\mathrm{S}}$ and $\mathbf{p}_i^{\mathrm{S}}$ is located to the *left* of $\mathbf{p}_{i-1}^{\mathrm{S}}$. Additionally, $\left\{\mathbf{p}_{i-1}^{\mathrm{S}}, \mathbf{p}_{i-2}^{\mathrm{S}}, , ..., \mathbf{p}_1^{\mathrm{S}}\right\}$ contains the list of scan points to the right of $\mathbf{p}_i^{\mathrm{S}}$ and is denoted by $\mathbf{p}_i^{\mathrm{S}-}$. In the same way, $\mathbf{p}_i^{\mathrm{S}+} = \left\{\mathbf{p}_{i+1}^{\mathrm{S}}, \mathbf{p}_{i+2}^{\mathrm{S}}, , ..., \mathbf{p}_n^{\mathrm{S}}\right\}$ denotes the list of scan points to the left of $\mathbf{p}_i^{\mathrm{S}}$. Both lists $\mathbf{p}_i^{\mathrm{S}-}$ and $\mathbf{p}_i^{\mathrm{S}+}$ can be extended beyond $\mathbf{p}_1^{\mathrm{S}}$ and $\mathbf{p}_n^{\mathrm{S}}$ if the adopted sensor has a 360° field of view (FFOV). In such a case, both lists are adjusted as follows: $\mathbf{p}_i^{\mathrm{S}-} = \left\{\mathbf{p}_{i-1}^{\mathrm{S}}, ..., \mathbf{p}_1^{\mathrm{S}}, \mathbf{p}_n^{\mathrm{S}}, \mathbf{p}_{n-1}^{\mathrm{S}}..., \mathbf{p}_{i+1}^{\mathrm{S}}\right\}$ and $\mathbf{p}_i^{\mathrm{S}+} = \left\{\mathbf{p}_{i+1}^{\mathrm{S}}, ..., \mathbf{p}_n^{\mathrm{S}}, \mathbf{p}_1^{\mathrm{S}}, \mathbf{p}_2^{\mathrm{S}}, ..., \mathbf{p}_{i-1}^{\mathrm{S}}\right\}$. It is apparent that, in case of a FFOV sensor, $\mathbf{p}_i^{\mathrm{S}-}$ and $\mathbf{p}_i^{\mathrm{S}+}$ consist of the same elements but with a reversed order.

Each point $\mathbf{p}_i^{\mathrm{S}}$ has two neighborhoods; one is located to its left denoted by $\mathbf{p}_{i+}^{\mathrm{S}}$ while the other is located to its right denoted by $\mathbf{p}_{i-}^{\mathrm{S}}$. For example, for a FFOV sensor, the right (resp. left) neighborhood of $\mathbf{p}_1^{\mathrm{S}}$ is $\mathbf{p}_n^{\mathrm{S}}$ (resp. $\mathbf{p}_2^{\mathrm{S}}$).

It is important to note that accessing any element in a list (e.g. $\mathbf{p}_i^{\mathrm{S}+}$, $\mathbf{p}_i^{\mathrm{S}-}$) is performed sequentially (the order is respected). For example, point $\mathbf{p}_i^{\mathrm{S}}$ is accessed before point $\mathbf{p}_{i+1}^{\mathrm{S}}$.

Assume that $\mathcal{F}$ represents a frame centered at point c and rotated by angle $\theta_{\mathcal{F}}$, relative to the robot coordinate system. The position of a point $\mathbf{p}_i$, with respect to frame $\mathcal{F}$ is represented as follows:

$$^{\mathcal{F}}\mathbf{p}_i = R^{-1}\left(\mathbf{p}_i - c\right) \tag{6.1}$$

where R is defined as:

$$R = \begin{bmatrix} \cos(\theta_{\mathcal{F}}) & -\sin(\theta_{\mathcal{F}}) \\ \sin(\theta_{\mathcal{F}}) & \cos(\theta_{\mathcal{F}}) \end{bmatrix} \tag{6.2}$$

For a better visualization, superscripts are dropped in figures (for instance, the depth point $\mathbf{p}_j^{\mathrm{S}}$ becomes $\mathbf{p}_j$).

6.2 Detecting Gaps

This section introduces a new strategy for finding out gaps. Compared to our earlier work developed in [MFMJ10] (presented in section 3.1.3), the new strategy avoids improper gaps as explained in section 6.2.3. The major part in locating a gap is the detection of a spatial discontinuity in the sensor data, as will be described in section 6.2.1. Generally, the algorithm consists of two steps: first, all gaps V that can be seen from the current robot's view are found out, as explained in section 6.2.2. The second step, presented in section 6.2.3, implies identifying and discarding useless gaps, reducing V to G. Notice that this algorithm is based on our paper published in [MFM18]

6.2.1 Spatial Discontinuities

A spatial discontinuity is associated with an area in the workspace which is invisible from $\mathbf{p}_r$ [MFM18]. Assume that $w_{\min}$ represents the width of the narrowest opening through which the vehicle may pass. In principle, specifying $w_{\min}$ depends on the shape of the vehicle; if it is rectangular, $w_{\min}$ is set to its minimum dimension (width). But, for a disc-shaped vehicle, $w_{\min}$ will be its diameter. A spatial discontinuity takes place between two adjacent scan measurements (e.g. $\mathbf{p}_i^{\mathrm{S}}$ and $\mathbf{p}_{i\pm1}^{\mathrm{S}}$) if any of the following is met:

1) One of the two scan measurements is not an obstacle point (i.e. returns the maximum range of the sensor):

$$(r_i^{\mathrm{S}} = r_{\max}) \oplus (r_{i\pm1}^{\mathrm{S}} = r_{\max})$$

2) There is a spatial distance between both scan points more than $w_{\min}$:

$$\left\| \mathbf{p}_i^{\mathrm{S}} - \mathbf{p}_{i\pm1}^{\mathrm{S}} \right\| > w_{\min}$$

It is apparent that the first discontinuity consists of one end point (an obstacle point), while the second has two. The former is called a "unilateral discontinuity", whilst the latter is "bilateral discontinuity". Each discontinuity is characterized by its basis. In case of a "bilateral discontinuity", the basis is the end point closer to the robot. For a unilateral discontinuity, the basis is its unique end point.

We classify each discontinuity to *left* or *right* based on the location of the invisible area, as seen by the robot sensors. A discontinuity of type unilateral is called a "left discontinuity" if its basis is to the left of the non obstacle point. It is called a "right discontinuity" otherwise. For a discontinuity of type bilateral, it is characterized as left if its basis is to the left of the other end point. Otherwise, it is characterized as right.

6.2.2 Gaps Search

The major part in extracting gaps is the detection of spatial discontinuities in the sensor data. Similar to the procedure presented in section 3.1.3, this is carried out in two steps: first, we detect discontinuities of type right, traveling from $\mathbf{p}_1^{\mathrm{S}}$ to $\mathbf{p}_n^{\mathrm{S}}$ (counterclockwise). Second, discontinuities of type left are detected, traveling from $\mathbf{p}_n^{\mathrm{S}}$ to $\mathbf{p}_1^{\mathrm{S}}$ (clockwise). Both searches are described in the following:

"Counterclockwise search": For every two adjacent scan points (e.g. $\mathbf{p}_i^{\mathrm{S}}$ and $\mathbf{p}_{i+}^{\mathrm{S}}$), it is checked whether a right discontinuity $\mathrm{d_r}$ exists or not[2]. If so, the basis $\mathbf{p}_{\mathrm{b}}(\mathrm{d_r})$ of $\mathrm{d_r}$ specifies the "right side" $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ of a gap g. According to the "left side" $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, it is determined as follows:

1) Let O^+ denotes the sequence of *obstacles* located to the left of $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, where the angular difference between $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and any element in O^+ is less than π:

$$\mathrm{O}^+ = \left\{ \mathbf{p}_k^{\mathrm{S}} \in \mathbf{p}_{\mathrm{r}}^{\mathrm{S}+}(\mathrm{g}) \;\middle|\; r_k^{\mathrm{S}} \neq r_{\max},\, \mathrm{proj}(\theta_k^{\mathrm{S}} - \theta_{\mathrm{r}}(\mathrm{g})) > 0 \right\} \tag{6.3}$$

[2]Notice that for a FFOV sensor, the last two scan points are $\mathbf{p}_n^{\mathrm{S}}$ and $\mathbf{p}_1^{\mathrm{S}}$, but for a sensor whose field of view is limited (LFOV), they will be $\mathbf{p}_{n-1}^{\mathrm{S}}$ and $\mathbf{p}_n^{\mathrm{S}}$.

where $\theta_{\mathrm{r}}(\mathrm{g})$ represents the angle towards $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$.

2) Each element $\mathbf{p}_i \in \mathrm{O}^+$ is checked for validity to be a left side as follows[3]. Let g be a virtual gap created by $\mathbf{p}_i$ and $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$. Obstacle $\mathbf{p}_i$ is valid if g is visible from $\mathbf{p}_r$. Otherwise, it is invalid. Visibility here means that the area defined by the line segments $\overline{\mathbf{p}_r\mathbf{p}_i}$, $\overline{\mathbf{p}_r\mathbf{p}_{\mathrm{r}}(\mathrm{g})}$, and $\overline{\mathbf{p}_i\mathbf{p}_{\mathrm{r}}(\mathrm{g})}$ is collision-free. This condition is violated if the line segment that connects $\mathbf{p}_i$ to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ passes through an occupied or an occluded region. To represent this condition mathematically, in [MFM18] we have defined an angle, named the "visibility angle" of $\mathbf{p}_i$ with respect to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$. This angle corresponds to each $\mathbf{p}_i \in \mathrm{O}^+$ and can be expressed as follows:

$$^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_i}(\mathrm{g}) = \arccos\left(\frac{r_{\mathrm{r}}^2(\mathrm{g}) + D_i^2 - r_i^2}{2D_i\, r_{\mathrm{r}}(\mathrm{g})}\right) \tag{6.4}$$

where $r_{\mathrm{r}}(\mathrm{g})$ and r_i are the distances to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and $\mathbf{p}_i$, respectively, and D_i the distance from $\mathbf{p}_i$ to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$.

The visibility condition of $\mathbf{p}_i$ is fulfilled if the following equation holds:

$$^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_i}(\mathrm{g}) < \wedge^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g})$$

where $\wedge^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g})$ is the minimum "visibility angle" among those associated with all obstacles belonging to O^+ and having a rank less than i:

$$\wedge\,^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g}) = \underset{\mathbf{p}_k \in \mathrm{O}^+}{\operatorname{argmin}}\, ^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_k}(\mathrm{g}), \;\; k < i \tag{6.5}$$

3) Let $\mathrm{V}(\mathrm{O}^+)$ be the list of valid obstacle points belonging to O^+, the left side is created by the obstacle point closest to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and contained in $\mathrm{V}(\mathrm{O}^+)$:

$$\mathbf{p}_{\mathrm{l}}(\mathrm{g}) = \underset{\mathbf{p}_k}{\operatorname{argmin}} \left\|\mathbf{p}_k - \mathbf{p}_{\mathrm{r}}(\mathrm{g})\right\|, \;\; \mathbf{p}_k \in \mathrm{V}(\mathrm{O}^+) \tag{6.6}$$

Notice that if $\mathrm{V}(\mathrm{O}^+)$ does not contain any element ($\mathrm{V}(\mathrm{O}^+) = \phi$), the left side $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ is created by a virtual point at an angle of $\theta^{\mathrm{S}}_{\mathrm{r}+}(\mathrm{g})$ and a distance of $R + \mathrm{d}_{\mathrm{safe}}$ from $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, where $\theta^{\mathrm{S}}_{\mathrm{r}+}(\mathrm{g})$ is the angle towards the "left neighborhood" of $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, $\mathbf{p}^{\mathrm{S}}_{\mathrm{r}+}(\mathrm{g})$, and $\mathrm{d}_{\mathrm{safe}}$ a desired clearance to obstacles.

[3]Notice that the subscript of $\mathbf{p}_i$ represents the rank of $\mathbf{p}_i$ with respect to O^+, not to S.

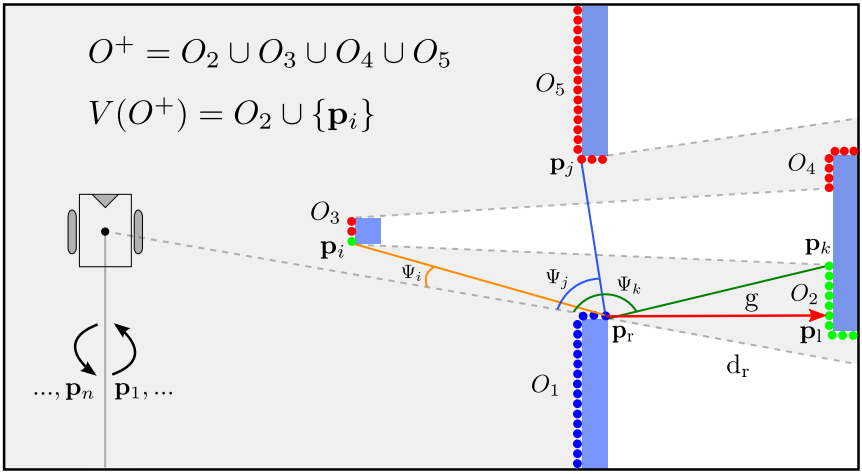


Figure 6.1: Finding out a gap (g) by the "counterclockwise search". The light gray color depicts the area covered by the sensing system. The obstacles are shown by blue regions, where the list of detected depth points S are visualized by small colored circles. See the text for explaining how gap g is detected. For a better visualization, the symbol denoting the gap (g) and the superscript $\mathbf{p}_r$ in the "visibility angle" are eliminated (reprinted from [MFM18], with permission from Elsevier).

Seeking for the remaining gaps is performed starting from point $\mathbf{p}_l(g)$. We denote the sequence of gaps detected by the "counterclockwise search" as V_{cc}.

An example is shown in figure 6.1 where the environmental structure consists of four obstacles, perceived as 5 lists of points, labeled O_1 - O_5. Point $\mathbf{p}_r$ represents the right side of gap g which is created by the basis of d_r. The list of valid and invalid obstacles in O^+ are visualized by green and red colors, respectively, where $O^+ = O_2 \cup O_3 \cup O_4 \cup O_5$. For example, $\mathbf{p}_i$ is valid because it has the minimum visibility angle Ψ_i among those associated with all obstacles belonging to O^+ and having a rank less than i (O_2 here). Point $\mathbf{p}_j$ does not satisfy the validity condition because $\Psi_j \not< \wedge\Psi_{j^-}$ where, in this example, $\wedge\Psi_{j^-} = \Psi_i$. It can be seen that the line segments between the invalid points and $\mathbf{p}_r$ (e.g. $\overline{\mathbf{p}_r\mathbf{p}_j}$) pass through either an obstacle or an occluded region. Among the valid points in O^+, the closest to $\mathbf{p}_r$ defines the left side of g (marked as $\mathbf{p}_l$).

"Clockwise search": The sensor data is searched for gaps similar to the counterclockwise search, but performed in the reverse order: for every two adjacent scan points $(\mathbf{p}_i^{\mathrm{S}}, \mathbf{p}_{i-}^{\mathrm{S}})$, it is checked whether a left discontinuity $\mathrm{d_l}$ exists or not[4]. If yes, the "left side" $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ of a gap g is formed by the basis $\mathbf{p}_{\mathrm{b}}(\mathrm{d_l})$ of $\mathrm{d_l}$. The "right side" $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ is determined as described in the following.

1) The sequence of *obstacles* O^- located to the right of $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ is identified, where the angular difference between $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ and any element in O^- is less than π:

$$\mathrm{O}^- = \left\{ \mathbf{p}_k^{\mathrm{S}} \in \mathbf{p}_{\mathrm{l}}^{\mathrm{S}-}(\mathrm{g}) \;\middle|\; r_k^{\mathrm{S}} \neq r_{\max},\, \mathrm{proj}\left(\theta_k^{\mathrm{S}} - \theta_{\mathrm{l}}(\mathrm{g})\right) < 0 \right\} \tag{6.7}$$

where $\theta_{\mathrm{l}}(\mathrm{g})$ denotes the angle towards the left side.

2) The elements of O^- are checked for validity. The outcome is the sequence of valid obstacles $\mathrm{V}(\mathrm{O}^-)$:

$$\mathrm{V}(\mathrm{O}^-) = \left\{ \mathbf{p}_i \in \mathrm{O}^- \;\middle|\; {}^{\mathbf{p}_{\mathrm{l}}}\Psi_{\mathbf{p}_i}(\mathrm{g}) < \wedge^{\mathbf{p}_{\mathrm{l}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g}) \right\} \tag{6.8}$$

where $\wedge^{\mathbf{p}_{\mathrm{l}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g})$ can be expressed as:

$$\wedge\, {}^{\mathbf{p}_{\mathrm{l}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g}) = \underset{\mathbf{p}_k \in \mathrm{O}^-}{\mathrm{argmin}}\, {}^{\mathbf{p}_{\mathrm{l}}}\Psi_{\mathbf{p}_k}(\mathrm{g}), \;\; k < i \tag{6.9}$$

3) The right side is created by the obstacle point belonging to $\mathrm{V}(\mathrm{O}^-)$ and closest to $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$:

$$\mathbf{p}_{\mathrm{r}}(\mathrm{g}) = \underset{\mathbf{p}_k}{\mathrm{argmin}} \left\| \mathbf{p}_k - \mathbf{p}_{\mathrm{l}}(\mathrm{g}) \right\|, \;\; \mathbf{p}_k \in \mathrm{V}(\mathrm{O}^-) \tag{6.10}$$

If $V(\mathrm{O}^-) = \phi$, the right side is created by a virtual point at an angle of $\theta_{\mathrm{l}-}^{\mathrm{S}}(\mathrm{g})$ and a distance of $R + \mathrm{d_{safe}}$ from $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, where $\theta_{\mathrm{l}-}^{\mathrm{S}}(\mathrm{g})$ is the angle towards the "right neighborhood" of $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, denoted as $\mathbf{p}_{\mathrm{r}-}^{\mathrm{S}}(\mathrm{g})$.

Seeking for the remaining gaps is performed starting from point $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$. The sequence of gaps detected by the "clockwise search" is denoted as V_{c}.

After performing both searches (counterclockwise and clockwise), the output is the set of all gaps $V = V_{\mathrm{cc}} \cup V_{\mathrm{c}}$ that can be seen by the sensing system.

[4]Notice that for a FFOV sensor, the last two scan points are $\mathbf{p}_1^{\mathrm{S}}$ and $\mathbf{p}_n^{\mathrm{S}}$, but for a sensor whose field of view is limited (LFOV), they will be $\mathbf{p}_2^{\mathrm{S}}$ and $\mathbf{p}_1^{\mathrm{S}}$.

6.2.3 Gaps Reduction

As mentioned at the beginning of this chapter, our method discards *useless* gaps, reducing V to G. A gap is considered useless if it can be reached by traversing another gap. This section shows how useless gaps are identified and eliminated.

Every gap $\mathrm{g} \in V$ is classified based on its position with respect to the sensor frame. If $|\theta_{\mathrm{l}}(\mathrm{g}) - \theta_{\mathrm{r}}(\mathrm{g})| \leq \pi$, g is called a "front gap". Otherwise, it is a "rear gap". A gap $\mathrm{g_j}$ is reachable from gap $\mathrm{g_i}$ if two conditions are met: first, both gaps have the same type. Second, $\mathrm{g_i}$ includes $\mathrm{g_j}$. Fulfilling the second condition is based on the gap type: if both $\mathrm{g_i}$ and $\mathrm{g_j}$ are of type front, the following equation must hold:

$$\theta_{\mathrm{r}}(\mathrm{g_j}) \geq \theta_{\mathrm{r}}(\mathrm{g_i}) \wedge \theta_{\mathrm{l}}(\mathrm{g_j}) \leq \theta_{\mathrm{l}}(\mathrm{g_i})$$

However, if they are of type rear, the equation to be checked is:

$$\hat{\theta}_{\mathrm{r}}(\mathrm{g_j}) \geq \hat{\theta}_{\mathrm{r}}(\mathrm{g_i}) \wedge \hat{\theta}_{\mathrm{l}}(\mathrm{g_j}) \leq \hat{\theta}_{\mathrm{l}}(\mathrm{g_i})$$

where $\hat{\theta}_{\mathrm{l}}(\mathrm{g})$ (resp. $\hat{\theta}_{\mathrm{r}}(\mathrm{g})$) is the angle towards the point that makes an angular difference of π with $\theta_{\mathrm{l}}(\mathrm{g})$ (resp. $\theta_{\mathrm{r}}(\mathrm{g})$):

$$\hat{\theta}_{\mathrm{l}}(\mathrm{g}) = \mathrm{proj}\,(\theta_{\mathrm{l}}(\mathrm{g}) - \pi) \tag{6.11}$$

$$\hat{\theta}_{\mathrm{r}}(\mathrm{g}) = \mathrm{proj}\,(\theta_{\mathrm{r}}(\mathrm{g}) - \pi) \tag{6.12}$$

From V, we eliminate those gaps that are reachable by other gaps. The list of remaining gaps is denoted as G:

$$G = V \setminus \{\mathrm{g_j} \in V \mid \mathrm{g_j} \text{ is reachable from } \mathrm{g_i} \in V,\, \mathrm{g_i} \neq \mathrm{g_j}\} \tag{6.13}$$

Figure 6.2 shows an example of the "gaps search" procedure. In the "counter-clockwise search", the bases of the discontinuities (of type right) labeled AB, HI, NO, RS, and WX create the "right sides" of the gaps marked as 1, 2, 3, 4, and 5. Their corresponding "left sides" are, respectively, marked as C, L, Q, V, and Z. Performing the "clockwise search" results in gaps $6 - 10$, whose "left sides" are formed by the bases of UT, QP, LK, FE, and ZY, and whose "right sides" are

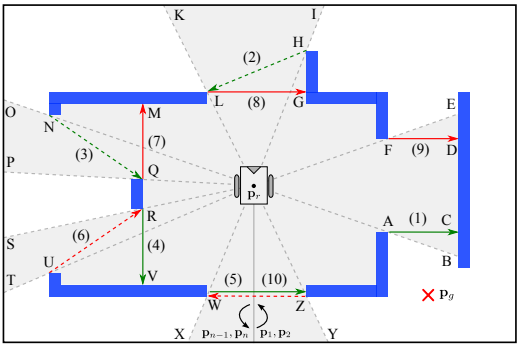


Figure 6.2: Finding out gaps by the AG method. The green and red arrows visualize the gaps that are found by the "counterclockwise" and "clockwise" searches, respectively. In the "gaps reduction" step, each gap depicted by a dashed arrow is eliminated (reprinted from [MFM18] , with permission from Elsevier).

created by points R, M, G, D, and W, respectively. It is obvious that gaps 2, 3, 6, and 10 can be discarded as they are reachable through gaps 8, 7, 4, and 5.

Remark 1 (Comparison to Other Strategies) *The AG approach as well as the CG technique [MFMJ10] have in common the advantage of reducing the number of detected gaps compared to those found by the ND methods (e.g. [MM04], [MOM04], [Min05], and [DB08]). An example is shown in figure 6.3 where the total number of gaps returned by both AG and CG are 5 and 4 marked as* 1 - 5 *and i - iv, respectively. For the ND methods, 14 gaps are detected labeled* A - N. *When compared to CG, the AG method avoids improper gaps such as gap ii. It can be seen that gap ii is reachable through either gap* 2 *or gap* 3 *which are even more convenient to be detected here*[5]*. Furthermore, gap* 5 *replaces gaps iii and iv since the AG method, unlike CG, deals with FFOV sensors. Additionally, gaps* 1 *and* 4 *have been returned rather than gap i, since the angular width of each gap in AG is limited to a maximum absolute value of* π*. This is important to cope with the collision avoidance procedure presented in section 6.4.*

[5]Despite the fact that the robot may pass through gap *ii*, the navigability check algorithm proposed in [MOM04] considers this gap as non navigable because the line segment between the center of *ii* and the robot's origin intersects an obstacle. AG avoids detecting such a gap by examining the validity of obstacles in the "gaps search" process.

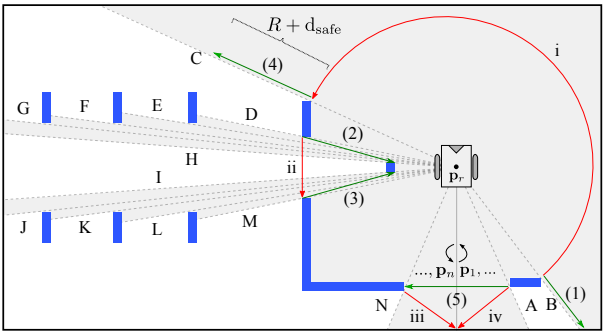


Figure 6.3: Extracting gaps by different methodologies, including the proposed AG approach. The total number of gaps returned by AG and CG [MFMJ10] are 5 (marked as 1 - 5) and 4 (labeled i - iv), depicted by green and red arrows, respectively. For the ND methods (e.g [MM04] and [DB08]), 14 gaps are detected, marked as A - N (reprinted from [MFM18], with permission from Elsevier).

6.3 Admissible Gap

This section introduces the "Admissible gap" (AG) concept, providing a basis for developing our collision avoidance approach. Generally speaking, a gap is admissible if it is traversable by performing a motion command that respects the vehicle constraints. Section 6.3.1 reviews the kinematic constraints and describes important characteristics of the vehicle's path, which will be used to explain the AG concept. In section 6.3.2, a new methodology for traversing gaps is presented. Section 6.3.3 shows how this methodology, which respects the kinematic constraints, is utilized to identify the admissibility status of a specific gap. For a better visualization the symbol denoting the gap (g) is omitted in this section. For example, $\mathbf{p}_r(g)$ is replaced by $\mathbf{p}_r$.

6.3.1 Kinematic Constraints

The motion of any robotic system undergoes some constraints imposed by physical limitations. Here, we consider a differential-drive robot that navigates on an

even surface and subject to "rolling without slipping" velocity constraints. As discussed in chapter 2, the motion of this robot can be described by [LSL98]:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{6.14}$$

where (x, y, θ) denotes the position and orientation of the robot with respect to the global coordinate system.

This non-holonomic equality constraint reduces by one the dimension of the velocity space. Therefore, at a specific configuration, the robot's motion is described by only two parameters (linear and angular velocity).

As pointed out in chapter 2, the kinematic model of a differential-drive mobile robot is given as follows:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix} v + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} w \tag{6.15}$$

Fox et al. [FBT97] showed that any mobile robot whose model is described by Eq. (6.15) travels along trajectories made up of arc segments. In [MMS06], it has been shown that, at each control cycle, the robot's path is characterized by a circle whose center lies at the robot's y-axis. Assume that $\mathbf{p}_i$ represents any point in the workspace. The circular path followed by the robot to reach $\mathbf{p}_i$ is referred to as $\mathcal{T}_i$. The radius of this path is denoted by r_i and expressed as:

$$r_i = \frac{x_i^2 + y_i^2}{2y_i},\ r_i \in]-\infty, \infty[\tag{6.16}$$

where $(x_i,\ y_i)$ denotes the location of $\mathbf{p}_i$ relative to the robot's reference frame. The instantaneous center of curvature c_i of the circular path $\mathcal{T}_i$ is given by $(0, r_i)$. Notice that r_i can be positive or negative based on the sign of y_i.

Whenever the robot reaches $\mathbf{p}_i$, it will be tangential to $\mathcal{T}_i$, i.e. its heading is:

$$\theta_i = \begin{cases} \arccos\left(\frac{r_i - y_i}{r_i}\right), & \text{if } \operatorname{sgn}(x_i) = \operatorname{sgn}(y_i) \\ -\arccos\left(\frac{r_i - y_i}{r_i}\right), & \text{otherwise} \end{cases} \tag{6.17}$$

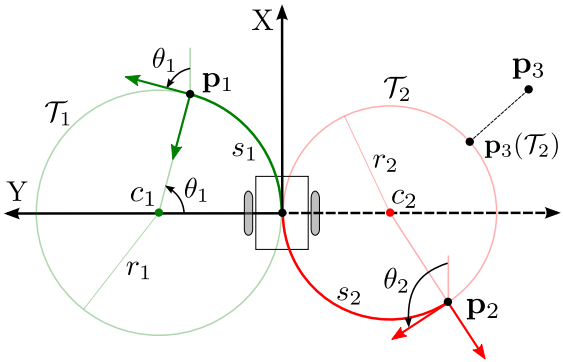


Figure 6.4: Visualization of the paths followed by a kinematically constrained robot. Circles $\mathcal{T}_1$ and $\mathcal{T}_2$ describe the paths followed by the robot to reach points $\mathbf{p}_1$ and $\mathbf{p}_2$. Whenever $\mathbf{p}_1$ (resp. $\mathbf{p}_2$) is reached, the robot's orientation is θ_1 (resp. θ_2) and the distance traversed is s_1 (resp. s_2). The point closest to $\mathbf{p}_3$ and falling on $\mathcal{T}_2$ is denoted by $\mathbf{p}_3(\mathcal{T}_2)$ (reprinted from [MM17], with permission from IEEE).

An example is shown in figure 6.4 which visualizes two points in the workspace marked as $\mathbf{p}_1$ and $\mathbf{p}_2$. These points are reached by traveling along circles $\mathcal{T}_1$ and $\mathcal{T}_2$. The radii and centers of $\mathcal{T}_1$ and $\mathcal{T}_2$ are $(r1, c_1)$ and $(r2, c_2)$, respectively. Angles θ_1 and θ_2 denote the robot's orientation when $\mathbf{p}_1$ and $\mathbf{p}_2$ are reached.

The distance traversed by the robot to reach $\mathbf{p}_i$ is the length s_i of the arc along $\mathcal{T}_i$ which connects the robot's origin to $\mathbf{p}_i$ (in figure 6.4, s_1 and s_2 represent the distances traveled by the robot to reach $\mathbf{p}_1$ and $\mathbf{p}_2$):

$$s_i = \begin{cases} |x_i|\,, & \text{if } y_i = 0 \\ |\theta_i \cdot r_i|\,, & \text{otherwise} \end{cases} \tag{6.18}$$

Let $\mathbf{p}_k$ be a point whose Cartesian coordinates are $(x_k,\, y_k)$, the point closest to $\mathbf{p}_k$ and falling on $\mathcal{T}_i$ is given by (in figure 6.4, $\mathbf{p}_3(\mathcal{T}_2)$ is the closest to $\mathbf{p}_3$):

$$\mathbf{p}_k(\mathcal{T}_i) = \langle 0, r_i \rangle + \vec{\mathbf{u}}_b \cdot |r_i| \tag{6.19}$$

where $\vec{\mathbf{u}}_b = \vec{\mathbf{b}} / \left|\vec{\mathbf{b}}\right|$ represents the unit direction vector of $\vec{\mathbf{b}} = \langle x_k, y_k - r_i \rangle$.

6.3.2 Traversing Gaps

There are several ways in which a robot navigates though a gap $\mathrm{g} \in G$, for instance, it may pass through the gap center or it may circumnavigate its side closer to $\mathbf{p}_g$. Our methodology is to achieve a compromise between safety and efficiency while driving the robot through g. The key idea is to assign a subgoal to g, referred to as $\mathbf{p}_\mathrm{s}$. The location of $\mathbf{p}_\mathrm{s}$ is determined in such a way that the robot circumnavigates one side of g, denoted as $\mathbf{p}_\mathrm{nav}$. While circumnavigating $\mathbf{p}_\mathrm{nav}$, progress towards g is made and a proper distance $\mathrm{d_s}$ is maintained to the obstacle point creating it. The value of $\mathrm{d_s}$ depends on the width of g and has been previously defined in section 3.1.1 (see Eq. (3.5)).

Determining $\mathbf{p}_\mathrm{nav}$ depends on the position of the gap with respect to the robot and goal locations. Let $\mathbf{p}_\mathrm{m}$ be the gap center (i.e. the point equidistant from $\mathbf{p}_\mathrm{r}$ and $\mathbf{p}_\mathrm{l}$). Since the main objective is to drive the robot towards the goal $\mathbf{p}_g$, we set $\mathbf{p}_\mathrm{nav}$ to the side $\mathbf{p}_\mathrm{cg}$ of gap g closer to $\mathbf{p}_g$. However, if this makes the distance to either side of g gets less than $\mathrm{d_s}$, $\mathbf{p}_\mathrm{nav}$ is set to the closer side of g along $\mathcal{T}_m$, denoted as $\mathbf{p}_\mathrm{cr}$ ($\mathcal{T}_m$ is the circular path that the robot follows to reach $\mathbf{p}_\mathrm{m}$):

$$\mathbf{p}_\mathrm{nav} = \begin{cases} \mathbf{p}_\mathrm{cg}, & \text{if } \|\mathbf{p}_\mathrm{l}(\mathcal{T}_\mathrm{m}) - \mathbf{p}_\mathrm{l}\| > \mathrm{d_s} \wedge \|\mathbf{p}_\mathrm{r}(\mathcal{T}_\mathrm{m}) - \mathbf{p}_\mathrm{r}\| > \mathrm{d_s} \\ \mathbf{p}_\mathrm{cr}, & \text{otherwise} \end{cases} \tag{6.20}$$

where $\mathbf{p}_\mathrm{l}(\mathcal{T}_\mathrm{m})$ (resp. $\mathbf{p}_\mathrm{r}(\mathcal{T}_\mathrm{m})$) is the point closest to $\mathbf{p}_\mathrm{l}$ (resp. $\mathbf{p}_\mathrm{r}$) and falling on $\mathcal{T}_\mathrm{m}$ (defined in Eq. (6.19)). Point $\mathbf{p}_\mathrm{cr}$ is expressed by the following equation:

$$\mathbf{p}_\mathrm{cr} = \begin{cases} \mathbf{p}_\mathrm{l}, & \text{if } s_\mathrm{l}(\mathcal{T}_\mathrm{m}) \leq s_\mathrm{r}(\mathcal{T}_\mathrm{m}) \\ \mathbf{p}_\mathrm{r}, & \text{otherwise} \end{cases} \tag{6.21}$$

where $s_\mathrm{l}(\mathcal{T}_\mathrm{m})$ and $s_\mathrm{r}(\mathcal{T}_\mathrm{m})$ are the distances traversed along $\mathcal{T}_\mathrm{m}$ to reach $\mathbf{p}_\mathrm{l}(\mathcal{T}_\mathrm{m})$ and $\mathbf{p}_\mathrm{r}(\mathcal{T}_\mathrm{m})$, respectively (defined in Eq. (6.18)).

Assume that $\mathcal{S}$ represents a circle whose center is $\mathbf{p}_\mathrm{nav}$ and whose radius is $\mathrm{d_s}$. The subgoal $\mathbf{p}_\mathrm{s}$ is placed at the "tangent point" $\mathbf{p}_t$ between $\mathcal{S}$ and $\mathcal{T}_t$ (the circular path that leads to $\mathbf{p}_t$). In principle, $\mathcal{S}$ and $\mathcal{T}_t$ are "mutually tangent" if:

$$x_\mathrm{nav}^2 + (y_\mathrm{nav} - r_t)^2 = (|r_t| \pm \mathrm{d_s})^2 \tag{6.22}$$

where r_t is the radius of $\mathcal{T}_t$ and $(x_{\text{nav}},\, y_{\text{nav}})$ the Cartesian coordinates of $\mathbf{p}_{\text{nav}}$ with respect to the robot coordinate system .

It can be deduced from Eq. (6.22) that two circles are tangential to $\mathcal{S}$. The radii of both circles are:

$$r_t = \frac{x_{\text{nav}}^2 + y_{\text{nav}}^2 - \mathrm{d_s}^2}{2(y_{\text{nav}} \pm \mathrm{d_s})} \tag{6.23}$$

With this, we can define $\mathbf{p}_t$ in terms of r_t as follows:

$$\mathbf{p}_t = \begin{cases} \langle x_{\text{nav}}, 0\rangle, & \text{if } r_t = \infty \\ \langle 0, r_t\rangle + \vec{\mathbf{u}}_v \cdot |r_t|\,, & \text{otherwise} \end{cases} \tag{6.24}$$

where $\vec{\mathbf{u}}_v = \vec{\mathbf{v}}/\,|\vec{\mathbf{v}}|$, $\vec{\mathbf{v}} = \langle x_{\text{nav}}, y_{\text{nav}} - r_t\rangle$. Notice that if $y_{\text{nav}} = -\mathrm{d_s}$ in Eq. (6.23), the value of r_t is ∞. In this case, $\mathcal{T}_t$ represents a line rather than a circle.

Solving Eqs. (6.23) and (6.24) results in two tangent points, $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$; only one of them is located in the direction leading to g, and it is apparently necessary to characterize the direction of travel to determine which one it is. Let $\mathbf{p}_i$ be a point in the workspace, $\mathcal{T}_i$ can be uniquely described by a *tangent direction* [MM09]:

$$\chi_i = \begin{cases} \arctan\left(\frac{1}{r_i}\right), & \text{if } x_i \geq 0 \\ \operatorname{sgn}(y_i) \cdot \pi - \arctan\left(\frac{1}{r_i}\right), & \text{otherwise} \end{cases} \tag{6.25}$$

This definition represents the direction towards $\mathbf{p}_i$ for a kinematically constrained mobile robot (the direction along $\mathcal{T}_i$). Notice that a positive x_i indicates a forward motion, while a negative x_i indicates a backward motion.

The subgoal $\mathbf{p}_s$ is placed at the "tangent point" $\mathbf{p}_t$ that lies in the direction of g, which is apparently the point that satisfies the following condition:

$$\operatorname{proj}(\chi_{\text{t}} - \chi_{\text{nav}}) \cdot \Upsilon < 0 \tag{6.26}$$

where χ_{nav} and χ_t denote the tangent directions corresponding to $\mathcal{T}_{\text{nav}}$ (the circular path that lead to $\mathbf{p}_{\text{nav}}$) and $\mathcal{T}_t$. The value of Υ is given by:

$$\Upsilon = \begin{cases} +1, & \text{if } \mathbf{p}_{\text{nav}} \text{ is a left side} \\ -1, & \text{if } \mathbf{p}_{\text{nav}} \text{ is a right side} \end{cases} \tag{6.27}$$

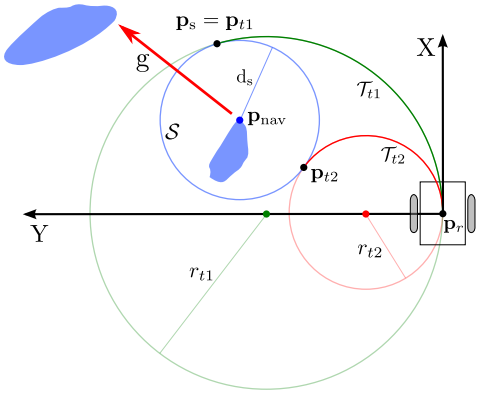


Figure 6.5: Assigning a subgoal $\mathbf{p}_s$ to a gap g in such a way that the robot circumnavigates one of its sides $\mathbf{p}_{\text{nav}}$ while obeying the kinematic constraints. First, we identify circle S whose center is $\mathbf{p}_{\text{nav}}$ and whose radius is d_s. This circle is mutually tangent to two circular paths ($\mathcal{T}_{t1}$ and $\mathcal{T}_{t2}$), each of which lies on the y-axis. The tangent points are labeled $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$. We locate the subgoal at $\mathbf{p}_{t1}$ as it leads to g (reprinted from [MM17], with permission from IEEE).

Figure 6.5 visualizes how a subgoal $\mathbf{p}_\text{s}$ associated with a gap g is located. It can be seen that S is mutually tangent to two circular paths; $\mathcal{T}_{t1}$ at $\mathbf{p}_{t1}$ and $\mathcal{T}_{t2}$ at $\mathbf{p}_{t2}$. Notice that $\mathbf{p}_\text{s}$ is located at $\mathbf{p}_{t1}$ as it lies in the direction that leads to g.

Remark 2 (Oscillation Removal) *The procedure presented above locates* $\mathbf{p}_\text{s}$ *at the tangent point on circle* $\mathcal{S}$*. By this means, a proper distance* d_s *is preserved to* $\mathbf{p}_{\text{nav}}$*, thus increasing the safety of navigation. However, if the origin of the robot's reference frame is touching circle* $\mathcal{S}$ *(i.e.* $\|\mathbf{p}_r - \mathbf{p}_{\text{nav}}\| = \text{d}_\text{s}$*), the coordinates of* $\mathbf{p}_\text{s}$ *will be* $(0,\ 0)$*. In this regard, the robot experiences oscillations and may turn on spot. Additionally, while moving along (circumnavigating)* $\mathbf{p}_{\text{nav}}$*, the robot may get inside circle* $\mathcal{S}$ *(i.e.* $\|\mathbf{p}_r - \mathbf{p}_{\text{nav}}\| < \text{d}_\text{s}$*). In this case, the new* $\mathbf{p}_\text{s}$ *takes the robot outside* $\mathcal{S}$*. Apparently, this behavior is repeated if the robot comes inside* $\mathcal{S}$ *once again. Next, we propose a solution that has shown the ability to avoid this drawback. Let* $\mathcal{N}$ *be a circle whose radius is* $r_{\text{nav}} = \|\mathbf{p}_r - \mathbf{p}_{\text{nav}}\|$ *and whose center is* $\mathbf{p}_{\text{nav}}$*. As soon as the distance between* $\mathbf{p}_r$ *and* $\mathbf{p}_{\text{nav}}$ *gets less than or equal to* d_s*, we locate* $\mathbf{p}_\text{s}$ *on* $\mathcal{N}$ *rather than on* $\mathcal{S}$*. Accordingly, the current*

distance between $\mathbf{p}_r$ *and* $\mathbf{p}_{\text{nav}}$ *is preserved, until having traversed the gap. Notice that* $\mathbf{p}_{\text{s}}$ *can be located at any distance* (d_{rs}) *from* $\mathbf{p}_r$ (d_{rs} *is the arc length between* $\mathbf{p}_r$ *and* $\mathbf{p}_{\text{s}}$*, not the Euclidean distance between them). In our experiments,* d_{rs} *was set to* ($r_{\text{nav}} \cdot \frac{\pi}{4}$). *It is clear that two points can be located on* $\mathcal{N}$*, such that the distance between either of them and* $\mathbf{p}_r$ *equals to* d_{rs}*. Obviously, we select the point that lies in the direction of the gap, following Eq. (6.26).*

6.3.3 Checking Admissibility

Reactive navigation consists of computing one action at each control cycle, such that collisions are avoided and progress towards the goal is guaranteed. This results in a series of collision-free motion controls, that guides a mobile robot from its starting position towards a given goal. Within this context, it is of interest to study the path that results from performing a single motion control.

It has been shown in section 6.3.2 that the execution of a single motion command $u = (v, w)$ causes the robot to travel along a circular path. Assume that this path goes through $\mathbf{p}_{\text{s}}$; it is characterized by circle $\mathcal{T}_{\text{s}}$ in section 6.3.2. Assume also that $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$ represents the arc segment which connects $(0, 0)$ to $\mathbf{p}_{\text{s}}$ along $\mathcal{T}_{\text{s}}$. A gap g is "admissible" (denoted as $\text{AG}\left[\mathbf{p}_r \rightarrow \text{g}\right]$) iff the robot does not collide with obstacles while driving it along $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$, i.e. $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$ is collision-free:

$$\text{AG}\left[\mathbf{p}_r \rightarrow \text{g}\right] \iff \mathcal{T}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right] \cap \bigcup_{i=1}^{n} \mathbf{p}_i^{\text{S}} = \phi$$

where $\mathcal{T}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$ represents the area that the robot occupies while traveling from $(0, 0)$ to $\mathbf{p}_s$ (i.e. traveling along $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$).

In the following, we propose an algorithm for checking the admissibility state of gaps in such a way that the shape of the robot is taken into account. The output is the list of all obstacles that may cause collision while driving the robot from $\mathbf{p}_r$ towards $\mathbf{p}_{\text{s}}$ (i.e. along $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$), referred to as $\mathcal{O}_{\text{collision}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\text{s}}\right]$.

Assume that the robot's footprint can be represented by a polygon whose edges are $\mathcal{P}_e, e = 1, ..., m$. For each $\mathbf{p}_i^{\text{S}} \in \text{S}$, it is checked if any of the robot edges intersects circle $\mathcal{C}(c_s, \mathbf{p}_i^{\text{S}})$ that goes through $\mathbf{p}_i^{\text{S}}$ and centered at $c_s = (0, r_{\text{s}})$; the

instantaneous center of curvature leading to the subgoal[6]. If no intersection occurs, $\mathbf{p}_i^{\mathrm{S}}$ is collision-free, and therefore it is discarded. Otherwise, we do the following: assume that $\mathbf{p}_e$ represents the point at which circle $\mathcal{C}(c_s, \mathbf{p}_i^{\mathrm{S}})$ intersects the robot, say at edge $\mathcal{P}_e$. Whenever $\mathbf{p}_{\mathrm{s}}$ is reached (moving a long $\tau\,[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}]$), the coordinates of $\mathbf{p}_e$ with respect to the current robot's frame is given by:

$$\mathbf{p}_e^* = \begin{cases} \mathbf{p}_e + \mathbf{p}_{\mathrm{s}}, & \text{if } y_{\mathrm{s}} = 0 \\ \mathcal{R}\,\mathbf{p}_e + t, & \text{otherwise} \end{cases} \tag{6.28}$$

where $\mathcal{R}$ and t are given by:

$$\mathcal{R} = \begin{bmatrix} K & -L \\ L & K \end{bmatrix}, t = \begin{bmatrix} x_{\mathrm{s}} \\ y_{\mathrm{s}} \end{bmatrix} \tag{6.29}$$

where x_{s} and y_{s} are the x and y locations of $\mathbf{p}_{\mathrm{s}}$, and K and L are given by:

$$K = \frac{x_{\mathrm{s}}^2 - y_{\mathrm{s}}^2}{x_{\mathrm{s}}^2 + y_{\mathrm{s}}^2} \tag{6.30}$$

$$L = \frac{2x_{\mathrm{s}}y_{\mathrm{s}}}{x_{\mathrm{s}}^2 + y_{\mathrm{s}}^2} \tag{6.31}$$

Given $\mathbf{p}_i^{\mathrm{S}}$, $\mathbf{p}_e$, and $\mathbf{p}_e^*$ all located on circle $\mathcal{C}(c_s, \mathbf{p}_i^{\mathrm{S}})$, we check whether $\mathbf{p}_i^{\mathrm{S}}$ falls on the arc that connects $\mathbf{p}_e$ with $\mathbf{p}_e^*$, traveling along $\tau\,[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}]$. If yes, $\mathbf{p}_i^{\mathrm{S}}$ causes collision. Otherwise, it is collision-free. This condition is checked as follows: let $\mathcal{F}$ be a frame (shown in figure 6.6 by dark blue) centered at c_s and heads towards $\mathbf{p}_e$, i.e. rotated by $\theta_{\mathcal{F}}$:

$$\theta_{\mathcal{F}} = \operatorname{atan2}\left(y_e - r_{\mathrm{s}}, x_e\right) \tag{6.32}$$

where $(x_e,\, y_e)$ represents the location of $\mathbf{p}_e$, relative to the robot's frame.

Let ${}^{\mathcal{F}}\theta_e^*$ and ${}^{\mathcal{F}}\theta_i^{\mathrm{S}}$ be the angles towards $\mathbf{p}_e^*$ and $\mathbf{p}_i^{\mathrm{S}}$ with respect to frame $\mathcal{F}$, respectively, $\mathbf{p}_i^{\mathrm{S}}$ falls on the arc that connects $\mathbf{p}_e$ with $\mathbf{p}_e^*$, and hence added to $\mathcal{O}_{\text{collision}}\,[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}]$, if the following condition is fulfilled:

$$(\Delta \cdot {}^{\mathcal{F}}\theta_i^{\mathrm{S}}) \bmod 2\pi \leq (\Delta \cdot {}^{\mathcal{F}}\theta_e^*) \bmod 2\pi$$

[6]Notice that if $\mathcal{T}_{\mathrm{s}}$ is a line rather than a circle ($\mathbf{p}_{\mathrm{s}}$ lies on the x axis of the robot's reference frame), it is checked if any of the robot edges intersects the line that is parallel to $\overrightarrow{\mathbf{p}_r\mathbf{p}_s}$ and goes through the obstacle point $\mathbf{p}_i^{\mathrm{S}}$.

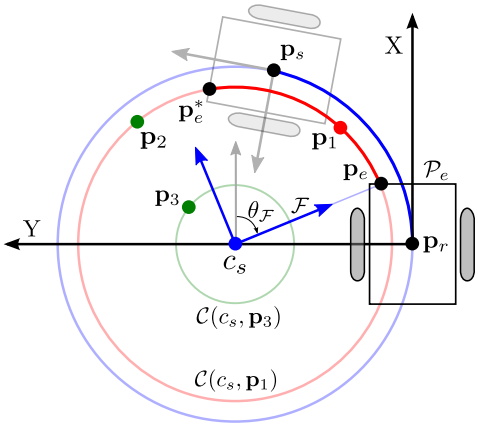


Figure 6.6: Collision checking along $\tau[\mathbf{p}_r \rightarrow \mathbf{p}_s]$; the path followed to reach $\mathbf{p}_s$. This example consists of three obstacles $\mathbf{p}_1$ - $\mathbf{p}_3$. The circles that pass through them and centered at c_s are $\mathcal{C}(c_s, \mathbf{p}_1)$ and $\mathcal{C}(c_s, \mathbf{p}_3)$. Circle $\mathcal{C}(c_s, \mathbf{p}_1)$ hits $\mathcal{P}_e$ in $\mathbf{p}_e$, where $\mathbf{p}_1$ lies on the arc between $\mathbf{p}_e$ and $\mathbf{p}_e^*$, but $\mathbf{p}_2$ doesn't. Therefore, $\mathbf{p}_1$ is in collision, but $\mathbf{p}_2$ is not. None of the robot edges intersects $\mathcal{C}(c_s, \mathbf{p}_3)$, thus $\mathbf{p}_3$ is collision-free (reprinted from [MM17], with permission from IEEE).

where Δ is defined as follows:

$$\Delta = \begin{cases} +1, & \text{if } \operatorname{sgn}(x_s) = \operatorname{sgn}(y_s) \\ -1, & \text{otherwise} \end{cases} \tag{6.33}$$

Notice that if $y_s = 0$ (i.e. $\tau[\mathbf{p}_r \rightarrow \mathbf{p}_s]$ represents a line segment rather than a circle) the condition to be checked (Eq.) is replaced by:

$$\operatorname{sgn}(x_s) x_e \leq \operatorname{sgn}(x_s) x_i^S \leq \operatorname{sgn}(x_s)(x_e^*) \tag{6.34}$$

where x_e^* represents the x location of $\mathbf{p}_e^*$, with respect to the robot's frame.

Figure 6.6 shows how to check whether $\tau[\mathbf{p}_r \rightarrow \mathbf{p}_s]$ (depicted by a dark blue arc) is collision free or not, given 3 obstacles labeled $\mathbf{p}_1$ - $\mathbf{p}_3$. It is apparent that circle $\mathcal{C}(c_s, \mathbf{p}_1)$, which goes through both $\mathbf{p}_1$ and $\mathbf{p}_2$, hits edge $\mathcal{P}_e$ in $\mathbf{p}_e$. The arc that connects $\mathbf{p}_e$ to $\mathbf{p}_e^*$, while moving along $\tau[\mathbf{p}_r \rightarrow \mathbf{p}_s]$, is visualized by dark red. It

can be seen that $\mathbf{p}_1$ lies on this arc, but $\mathbf{p}_2$ doesn't. Therefore, the former is in collision with the robot, but the latter is not. For obstacle $\mathbf{p}_3$, none of the robot edges intersects circle $\mathcal{C}(c_s, \mathbf{p}_3)$, and thus it is collision-free.

Notice that the above mentioned steps are applied for each edge of the polygon representing the robot $\mathcal{P}_e$. Apparently, the collision check can be reduced to one step if the robot's boundary can be represented by one equation (e.g. an eclipse) rather than approximating it by a polygonal shape.

6.4 AG Obstacle Avoidance Method

After having explained the concept of "admissible gap", we show how it can be used to drive a mobile robot in unknown cluttered scenarios, such that the shape and kinematic constraints are respected. The overall idea is described in the following. At each time step, the data coming from sensors is analyzed to find out if there is a navigable path towards the goal (section 6.4.1). If no such path exists, the robot will be driven towards a gap instead (similar to the holonomic solutions proposed in chapters 3 and 4). For determining the set of surrounding gaps, our strategy presented in section 6.2 is followed. Among the assembled list of gaps G, the closest to the goal (called "closest gap" in chapter 3) is selected. This gap must satisfy two conditions: first, the Euclidean distance between one of its sides and the goal must be the minimum compared to that of the other gaps. Second, it has to be navigable as described in section 6.4.2. All gaps are checked for both conditions until the closest gap is detected, or it is decided that no closest gap exists. Section 6.4.3 describes how the AG method computes the motion control which drives a robot towards the goal (resp. gap).

6.4.1 Goal Navigability Check

There are two cases in which the goal $\mathbf{p}_g$ is navigable from the current robot's location, either *directly* or using a virtual gap which we refer to as a *goal bridge*. Nevertheless, In both cases the goal should be visible from $\mathbf{p}_r$ (a direct line of sight to the goal exists). Let $\tau\,[\mathbf{p}_r \rightarrow \mathbf{p}_g]$ be the arc the robot follows to

reach $\mathbf{p}_g$, the goal is directly navigable if $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_g\right]$ is a collision-free path (i.e. $\mathcal{O}_{\text{collision}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_g\right] = \phi$). However, if there are obstacles causing collision, a goal bridge $\mathcal{B}$ is constructed between the robot and the goal. The idea behind the goal bridge is to maintain a safe trajectory while approaching the goal.

The first side of the goal bridge $\mathcal{B}$, denoted as $\mathbf{p}_{\text{f}}(\mathcal{B})$, is created by the obstacle point falling in $\mathcal{O}_{\text{collision}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_g\right]$ and closest to the circular path the robot follows to reach $\mathbf{p}_g$, $\mathcal{T}_g$:

$$\mathbf{p}_{\text{f}}(\mathcal{B}) = \underset{\mathbf{p}_i^{\text{S}}}{\operatorname{argmin}} \|\mathbf{p}_i^{\text{S}} - \mathbf{p}_i^{\text{S}}(\mathcal{T}_g)\| \tag{6.35}$$

where $\mathbf{p}_i^{\text{S}} \in \mathcal{O}_{\text{collision}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_g\right]$ and $\mathbf{p}_i^{\text{S}}(\mathcal{T}_g)$ denotes the point on $\mathcal{T}_g$ closest to $\mathbf{p}_i^{\text{S}}$ (see Eq. (6.19)).

Assume that the vector connecting the robot's origin to the goal $\overrightarrow{\mathbf{p}_r\mathbf{p}_g}$ divides the workspace into two regions; one to its left and the other to its right ($\mathcal{R}^+$ and $\mathcal{R}^-$). Denote the region that does not include $\mathbf{p}_{\text{f}}(\mathcal{B})$ by $\mathcal{R}^*$. The other side, denoted as $\mathbf{p}_{\text{o}}(\mathcal{B})$, is created by the *obstacle* point contained in $\mathcal{R}^*$ and closest to $\mathbf{p}_{\text{f}}(\mathcal{B})$. The angular distance between both sides, in the direction of the goal, must not exceed π. This is necessary to guarantee that traversing $\mathcal{B}$ directs the robot towards the direction leading to the goal. i.e:

$$\mathbf{p}_{\text{o}}(\mathcal{B}) = \underset{\mathbf{p}_i^{\text{S}} \in \mathcal{R}^*}{\operatorname{argmin}} \|\mathbf{p}_i^{\text{S}} - \mathbf{p}_{\text{f}}(\mathcal{B})\|, \ \alpha < \pi \tag{6.36}$$

where α is defined by:

$$\alpha = \begin{cases} \text{proj}(\theta_{\text{f}}(\mathcal{B}) - \theta_g) - \text{proj}(\theta_i^{\text{S}} - \theta_g), & \text{if } \mathbf{p}_{\text{f}}(\mathcal{B}) \in \mathcal{R}^+ \\ \text{proj}(\theta_i^{\text{S}} - \theta_g) - \text{proj}(\theta_{\text{f}}(\mathcal{B}) - \theta_g), & \text{otherwise} \end{cases} \tag{6.37}$$

where $\theta_{\text{f}}(\mathcal{B})$ and θ_g are the angles toward $\mathbf{p}_{\text{f}}(\mathcal{B})$ and $\mathbf{p}_g$, respectively.

Notice that if $\mathcal{R}^*$ does not contain any obstacle, $\mathbf{p}_{\text{o}}(\mathcal{B})$ is set to a virtual point in such a way that $\mathbf{p}_g$ is the center point between $\mathbf{p}_{\text{f}}(\mathcal{B})$ and $\mathbf{p}_{\text{o}}(\mathcal{B})$:

$$\mathbf{p}_{\text{o}}(\mathcal{B}) = (2x_g - x_{\text{f}}(\mathcal{B}), 2y_g - y_{\text{f}}(\mathcal{B})) \tag{6.38}$$

where (x_g, y_g) and $(x_{\text{f}}(\mathcal{B}), y_{\text{f}}(\mathcal{B}))$ are the Cartesian coordinates of $\mathbf{p}_g$ and $\mathbf{p}_{\text{f}}(\mathcal{B})$.

Once gap $\mathcal{B}$ is constructed, it is checked for navigability using the algorithm explained hereinafter in section 6.4.2. If it is navigable, the robot is directed towards it. Otherwise, we search about a navigable gap in G as explained above.

6.4.2 Gap Navigability Check

In general, a gap g is considered "navigable" if it can be traversed by performing a sequence of collision-free motion controls. This condition is fulfilled if there exists a list of "virtual gaps" $\mathrm{g}_k^*, k = 1, ..., L$, where g_1^* is "admissible" from $\mathbf{p}_r$, g_2^* is "admissible" from g_1^*, ..., g_k^* is "admissible" from g_{k-1}^*, ..., g is "admissible" from g_L^*. Nevertheless, in terms of reactive navigation, it is enough to find one gap g^* that satisfies two conditions: first, g^* is admissible from $\mathbf{p}_r$. Second, g can be reached by traversing g^*. By transforming the workspace into ARM in [MM09], only "admissible" gaps are considered "navigable". In the following, an algorithm is proposed that determines the position of g^*, if it exists. The inputs are the scan points list S and the gap g to be checked. The output of the algorithm g^* is initially set to g and computed by performing the following 2 steps iteratively:

Step 1: We split the set of scan points S into two subsets; one is denoted by $\mathrm{S_{in}}$ and referred to as the "interior" of g^*. It consists of those points that lie between both sides of g^*, with respect to the current robot's view (i.e. $\mathrm{S_{in}} = \{\mathbf{p}_\mathrm{r}(\mathrm{g}^*), \mathbf{p}_{\mathrm{r}+}^\mathrm{S}(\mathrm{g}^*), ..., \mathbf{p}_{\mathrm{l}-}^\mathrm{S}(\mathrm{g}^*), \mathbf{p}_\mathrm{l}(\mathrm{g}^*)\}$). Apparently, the second subset is the set difference of S and $\mathrm{S_{in}}$. It is denoted by $\mathrm{S_{ex}}$ and called the "exterior" of g^*. Let $\mathrm{O_{in}}$ and $\mathrm{O_{ex}}$ be the set of *obstacle* points contained in $\mathrm{S_{in}}$ and $\mathrm{S_{ex}}$, respectively (i.e. non obstacle points are discarded). We exclude from $\mathrm{O_{ex}}$ all obstacles making an angular difference (traveling in the direction of g^*) greater than π with either side of g^*. The resultant set of obstacles is referred to as $\mathrm{O'_{ex}}$:

$$\mathrm{O'_{ex}} = \mathrm{O_{ex}^+} \cup \mathrm{O_{ex}^-} \tag{6.39}$$

where $\mathrm{O_{ex}^-}$ and $\mathrm{O_{ex}^+}$ are defined by:

$$\mathrm{O_{ex}^+} = \left\{ \mathbf{p}_i^\mathrm{S} \in \mathrm{O_{ex}} \;\middle|\; \mathrm{proj}\,(\theta_i^\mathrm{S} - \theta_\mathrm{r}(\mathrm{g}^*)) > 0 \right\} \tag{6.40}$$

$$\mathrm{O_{ex}^-} = \left\{ \mathbf{p}_i^\mathrm{S} \in \mathrm{O_{ex}} \;\middle|\; \mathrm{proj}\,(\theta_i^\mathrm{S} - \theta_\mathrm{l}(\mathrm{g}^*)) < 0 \right\} \tag{6.41}$$

where $\theta_{\mathrm{l}}(\mathrm{g}^*)$ and $\theta_{\mathrm{r}}(\mathrm{g}^*)$ denote the angles towards the left $\mathbf{p}_{\mathrm{l}}(\mathrm{g}^*)$ and right $\mathbf{p}_{\mathrm{r}}(\mathrm{g}^*)$ sides of gap g^*, respectively.

All obstacles belonging to $\mathrm{O}'_{\mathrm{ex}}$ are checked for collision with $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$; the path followed to reach the subgoal $\mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)$ corresponding to g^*. The collision check is preformed using the algorithm presented in section 6.3.3, where the outcome is denoted by $\mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$. Apparently, g^* violates the admissibility condition if $\mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right] \neq \phi$. Nevertheless, a new gap g^{**} can be created in this case, such that traversing it leads to g^*. The location of g^{**} is determined in step 2 of the algorithm. If the collision set $\mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$ is empty, on the other hand, O_{in} is checked for collision similar to $\mathrm{O}'_{\mathrm{ex}}$ and the algorithm is terminated. Obviously, if the outcome of this collision check is empty, g^* is admissible and hence the algorithm returns the current g^*. Otherwise, the output will be NULL, which implies that g^* is not navigable. It is worth to mention that the obstacles eliminated from O_{ex} lie behind the robot while driving it towards $\mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)$. That is why we exclude them from the collision check.

Step 2: This step consists of creating a new "virtual gap" g^{**}, given the current one g^* and the collision set $\mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$. Similar to the goal bridge gap, the "first side" $\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})$ of g^{**} is set to the obstacle belonging to $\mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$ and closest to the circular path $\mathcal{T}_{\mathrm{s}}(\mathrm{g}^*)$ the robot follows to reach $\mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)$:

$$\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**}) = \underset{\mathbf{p}_i^{\mathrm{S}}}{\operatorname{argmin}} \|\mathbf{p}_i^{\mathrm{S}} - \mathbf{p}_i^{\mathrm{S}}(\mathcal{T}_{\mathrm{s}}(\mathrm{g}^*))\| \tag{6.42}$$

where $\mathbf{p}_i^{\mathrm{S}} \in \mathcal{O}'_{\mathrm{ex}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}^*)\right]$ and $\mathbf{p}_i^{\mathrm{S}}(\mathcal{T}_{\mathrm{s}}(\mathrm{g}^*))$ denote the point closest to $\mathbf{p}_i^{\mathrm{S}}$ and falling on $\mathcal{T}_{\mathrm{s}}(\mathrm{g}^*)$ (defined in Eq. (6.19)).

Assume that $\mathbf{p}_{\mathrm{m}}(\mathrm{g}^*)$ represents the center of g^* (i.e. the point equidistant from $\mathbf{p}_{\mathrm{r}}(\mathrm{g}^*)$ and $\mathbf{p}_{\mathrm{l}}(\mathrm{g}^*)$). Assume also that $\mathcal{M}$ denotes a frame centered at $\mathbf{p}_r$ and heads towards $\mathbf{p}_{\mathrm{m}}(\mathrm{g}^*)$. The workspace is divided into two regions; one to the right and the other to the left of the x-axis. They are denoted by $\mathcal{W}^-$ and $\mathcal{W}^+$, respectively. Let $O_{\mathrm{g}}^- = \{\mathbf{p}_{\mathrm{r}}(\mathrm{g}^*), \mathbf{p}_{\mathrm{r}-1}(\mathrm{g}^*), ...\}$ denotes the set of scan points falling to the right of g^*. Similarly, $O_{\mathrm{g}}^+ = \{\mathbf{p}_{\mathrm{l}}(\mathrm{g}^*), \mathbf{p}_{\mathrm{l}+1}(\mathrm{g}^*), ...\}$ denotes the set of scan points falling to the left of g^*. We search about the other side $\mathbf{p}_{\mathrm{o}}(\mathrm{g}^{**})$ in O_{g}^- if $\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})$ lies in $\mathcal{W}^+$ or in O_{g}^+ if $\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})$ lies in $\mathcal{W}^-$, where the obstacle point

closest to the first side is chosen. Notice that the search process is performed sequentially until the angular distance between the accessed element and $\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})$ gets more than π. This is to guarantee that g^{*} is reachable from g^{**}:

$$\mathbf{p}_{\mathrm{o}}(\mathrm{g}^{**}) = \underset{\mathbf{p}_i^{\mathrm{S}} \in \tilde{\mathrm{O}}_{\mathrm{ex}}}{\operatorname{argmin}} \|\mathbf{p}_i^{\mathrm{S}} - \mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})\|, \ \gamma \leq \beta < \pi \tag{6.43}$$

where $\tilde{\mathrm{O}}_{\mathrm{ex}} = \mathrm{O}_{\mathrm{ex}} \cup \mathbf{p}_{\mathrm{r}}(\mathrm{g}^{*}) \cup \mathbf{p}_{\mathrm{l}}(\mathrm{g}^{*})$, and β and γ are given by:

$$\beta = \begin{cases} {}^{\mathcal{M}}\theta_{\mathrm{f}}(\mathrm{g}^{**}) - {}^{\mathcal{M}}\theta_i^{\mathrm{S}}, & \text{if } \mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**}) \in \mathcal{W}^{+} \\ {}^{\mathcal{M}}\theta_i^{\mathrm{S}} - {}^{\mathcal{M}}\theta_{\mathrm{f}}(\mathrm{g}^{**}), & \text{otherwise} \end{cases} \tag{6.44}$$

$$\gamma = \begin{cases} {}^{\mathcal{M}}\theta_{\mathrm{f}}(\mathrm{g}^{**}) - {}^{\mathcal{M}}\theta_{\mathrm{r}}(\mathrm{g}^{*}), & \text{if } \mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**}) \in \mathcal{W}^{+} \\ {}^{\mathcal{M}}\theta_{\mathrm{l}}(\mathrm{g}^{*}) - {}^{\mathcal{M}}\theta_{\mathrm{f}}(\mathrm{g}^{**}), & \text{otherwise} \end{cases} \tag{6.45}$$

where ${}^{\mathcal{M}}\theta_{\mathrm{l}}(\mathrm{g}^{*})$, ${}^{\mathcal{M}}\theta_{\mathrm{r}}(\mathrm{g}^{*})$, ${}^{\mathcal{M}}\theta_i^{\mathrm{S}}$, and ${}^{\mathcal{M}}\theta_{\mathrm{f}}(\mathrm{g}^{**})$ are the angles towards $\mathbf{p}_{\mathrm{l}}(\mathrm{g}^{*})$, $\mathbf{p}_{\mathrm{r}}(\mathrm{g}^{*})$, $\mathbf{p}_i^{\mathrm{S}}$, and $\mathbf{p}_{\mathrm{f}}(\mathrm{g}^{**})$ with respect to frame $\mathcal{M}$, respectively.

After having determined g^{**}, it is assigned to g^{*} and step 1 is once again performed. Apparently, the procedure terminates because at each iteration the total number of elements contained in the exterior of g^{*} is decreased by at least one. If the algorithm returns a valid virtual gap g^{*}, then g satisfies the navigability condition. But if it returns NULL, g is considered as non-navigable.

An example of the algorithm is shown in figure 6.7, in which a robot navigates towards a given goal $\mathbf{p}_g$. Obviously, only two gaps (g_1, g_2) can be detected by the robot, where g_1 is the closest to the goal. The "interior" (O_{in}) of g_1 consists of the dark blue obstacle points in addition to the left $\mathbf{p}_{\mathrm{l}}(\mathrm{g}_1)$ and right $\mathbf{p}_{\mathrm{r}}(\mathrm{g}_1)$ sides of g_1 (visualized by red and violate). Apparently, the "exterior" (O_{ex}) of g_1 consists of the remaining obstacle points. Notice that, in this example, $\mathrm{O}'_{\mathrm{ex}} = \mathrm{O}_{\mathrm{ex}}$ since non of the elements contained in O_{ex} makes an angular difference more than π with either side of g_1. It can be seen that the path towards $\mathbf{p}_{\mathrm{s}}(\mathrm{g}_1)$ is in collision with the obstacle point shown by orange in addition to the dark green points (denoted by $\mathcal{O}'_{\mathrm{ex}}[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}_1)]$). Hence, a virtual gap is constructed whose first side $\mathbf{p}_{\mathrm{f}}(\mathrm{g}_1^{*})$ is created by the point contained in $\mathcal{O}'_{\mathrm{ex}}[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}_1)]$ and closest to $\mathcal{T}_{\mathrm{s}}(\mathrm{g}_1)$, which is obviously the orange point. Searching about the other side starts from $\mathbf{p}_{\mathrm{l}}(\mathrm{g}_1)$ and proceeds counterclockwise until the angular distance measured

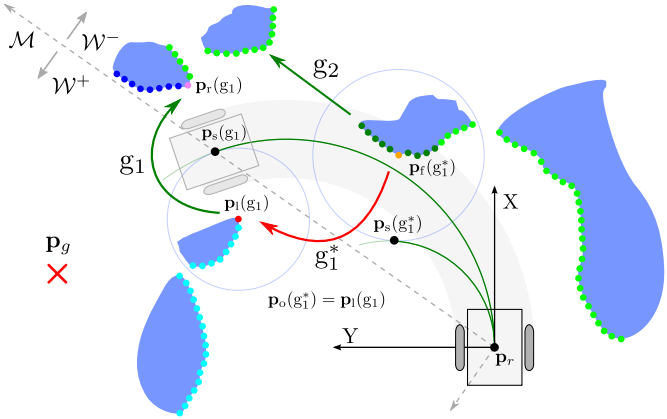


Figure 6.7: Checking navigability. The robot detects two gaps; g_1 and g_2. It is obvious that the closest gap (g_1) is non-admissible, since $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_s(\mathrm{g}_1)\right]$ is in collision with the obstacle points shown by orange and dark green dots. But, g_1 is navigable because it is possible to create an *admissible* gap (g_1^*) that leads to g_1. See the text for more information on constructing g_1^* (reprinted from [MM17], with permission from IEEE).

from $\mathbf{p}_f(\mathrm{g}_1^*)$ gets more than π. This includes the obstacles visualized by light blue in addition to $\mathbf{p}_l(\mathrm{g}_1)$. Among these obstacles, $\mathbf{p}_o(\mathrm{g}_1^*)$ is set to the closest to $\mathbf{p}_f(\mathrm{g}_1^*)$, which is clearly $\mathbf{p}_l(\mathrm{g}_1)$. It can be observed that the path towards $\mathbf{p}_s(\mathrm{g}_1^*)$ is collision free. Hence, the iterations are terminated and g_1^* is returned.

Remark 3 (Safety Improvement) *The collision check algorithm considers the exact region that the robot occupies while traveling along* $\tau\left[\mathbf{p}_r \rightarrow \mathbf{p}_s(\mathrm{g}^*)\right]$. *By this means, the distance to obstacles may get very small, causing the robot to be stuck somewhere. A straightforward solution to this issue is to perform the collision check after enlarging the robot's boundary by some threshold. But, this solution is a trade-off between completeness and safety (e.g. a gap may appear non navigable although it is not). Next, we introduce a better methodology to cope with this drawback. In a first step, the robot's boundary is enlarged by a proper clearance (a value of* $\mathrm{d}_s(\mathrm{g}^*) - w_{\min}$ *was used in our experiments). Then, the algorithm is applied and at each iteration the obstacle point belonging to* $\mathcal{O}_{\text{collision}}\left[\mathbf{p}_r \rightarrow \mathbf{p}_s(\mathrm{g}^*)\right]$ *and closest to* $\mathcal{T}\left[\mathbf{p}_r \rightarrow \mathbf{p}_s(\mathrm{g}^*)\right]$ *is registered. After having terminated the algo-*

rithm, the "virtual gap" $\mathrm{g}^*_{\mathrm{opt}}$ *that yields the maximum clearance is identified. In a second step, the algorithm is applied again, but starting from* $\mathrm{g}^*_{\mathrm{opt}}$ *(instead of* g*) and considering the exact robot size (i.e. without enlargement).*

6.4.3 Setting Motion Commands

In sections 6.4.1 and 6.4.2, we have seen how to identify the instantaneous target $\hat{\mathbf{p}}_g$, which can be the goal itself $\mathbf{p}_g$ or a subgoal corresponding to a navigable gap (either a goal bridge $\mathcal{B}$ or a virtual gap g^*). Next, we explain how the AG approach computes the motion control that drives a robot towards $\hat{\mathbf{p}}_g$.

In section 6.3, it has been shown that, at each control cycle, the robot's path can be described by a circular arc. Hence, the motion control is determined in such a way that the radius of curvature is maintained. Assume that the path followed by the robot to reach $\hat{\mathbf{p}}_g$ has a radius of $\hat{r}_g$ (defined in Eq. (6.16)). Apparently, a valid motion control must satisfy $v = w\hat{r}_g$. This condition can be seen as a line L_v in the control space whose slope is $\hat{r}_g^{-1}$ (see figure 6.8) [MMS06]. So, any point on L_v can be a valid motion control. Here, we consider controlling the velocity based on the clearance to obstacles, and hence S_{limit} is defined as:

$$S_{\mathrm{limit}} = \left(\sqrt{1 - \mathrm{sat}_{[0,1]} \left(\frac{D_{vs} - r_{\mathrm{min}}}{D_{vs}} \right)} \right) . S_{\mathrm{max}} \tag{6.46}$$

where r_{min} is the distance to the closest obstacle, D_{vs} a parameter that defines the size of a zone around the robot in which the velocity is limited (see chapter 4 for determining the value of D_{vs} based on the physical and dynamic properties of the robot), and S_{max} the distance from (0, 0) to the point at which the boundary of the maximum speeds intersects line L_v (coordinates are relative to the control space). The "sat" function caps S_{limit} at 0 if an obstacle is touched by the robot and at S_{max} if no obstacle lies within D_{vs}.

The motion control can now be defined as (originally from [MMS06]):

$$v = \mathrm{sgn}(\hat{x}_g) \cdot S_{\mathrm{limit}} \cdot \cos(\hat{\zeta}_g) \tag{6.47}$$

$$\omega = \mathrm{sgn}(\hat{x}_g) \cdot S_{\mathrm{limit}} \cdot \sin(\hat{\zeta}_g) \tag{6.48}$$

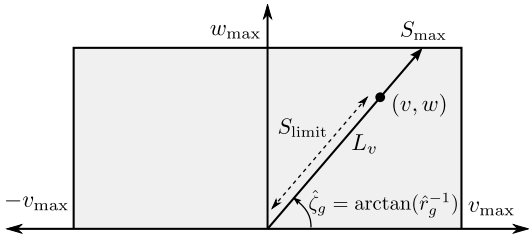


Figure 6.8: Determining the motion control in such a way that the radius of curvature $\hat{r}_g$ is maintained and the maximum possible velocity is respected. Any point on L_v can be a valid motion control as it satisfies $v = w\hat{r}_g$. Here, we specify S_{limit} so that the velocity of the robot is controlled based on the clearance to obstacles (originally from [MMS06]).

where $\hat{\zeta}_g = \arctan(\hat{r}_g^{-1})$ and $\hat{x}_g$ denotes the x-coordinate of $\hat{\mathbf{p}}_g$. A positive $\hat{x}_g$ indicates a forward motion, while a negative $\hat{x}_g$ indicates a backward motion.

6.5 Experimental Results

The reliability of the proposed method has been validated using our robotic platform GETbot, adopting the same setup presented in section 5.1. The following subsections provide the outcome of seven experiments executed in different scenarios[7] (Experiments 1 - 6 have been previously presented in [MFM18]). Notice that the only prior information available were the sensor data and the goal. The results are presented and compared with two state of the art methods which are ARM-ND+ [MM09] and the open source "robot navigation stack" DWA-A* [MBF$^+$10]. Moreover, the results are checked against the TGF approach [MFM16], presented in chapter 4. ARM-ND+ is an extension of ND+ [MOM04] that considers the robot shape and kinematics by transforming the workspace into ARM [MM09]. Since ND+ only computes forward motions, a change of coordinates was necessary [MMS06]. DWA-A* uses a costmap to look for a path, utilizing A* search, and employs the DWA method [FBT97] to track the generated path. For the sake of accuracy, only the front laserscanner was used to build

[7] Videos of these experiments can be found at: "`http://getwww.uni-paderborn.de/research/videos/ag`"

the costmap. In order to ensure safe navigation through tight passages, the maximum robot speeds were restricted to ($v = 0.5\,m/s,\ \omega = 1.0\,rad/s$). The robot operating system (ROS) has been employed to implement and test the aforementioned approaches. Maps of the environment were created by employing an available open source SLAM system [Gc10] implemented in ROS.

6.5.1 Experiment 1

This experiment consisted of a relatively simple scenario where GETbot was supposed to traverse an arena in which boxes were distributed as shown in figure 6.9a. The robot managed to reach the given goal using all of the aforementioned approaches. The generated trajectories are visualized in figures 6.9b, 6.9c, 6.9d, and 6.9e. Using ARM-ND+, oscillations occurred along the traversed route, see for example the trajectory near the locations labeled 1 - 3 in figure 6.9b. This is attributed to the limitations inherited from ND+ [MM04] as discussed in chapter 4. By running DWA-A*, GETbot got close to the A and B obstacles as shown in figure 6.9c. This is due to the fact that DWA-A* follows a path generated by A* planner. It can be seen that the trajectories followed by both TGF and AG are quite similar. They were able to drive GETbot with improved safety and smoothness compared to ARM-ND+ and DWA-A*. The robot speeds were recorded and plotted against the elapsed time as shown in figures 6.9f - 6.9i.

6.5.2 Experiment 2

In this experiment, the difficulty of navigation was increased by placing obstacles in such a way that a large U-shape is formed as shown in figure 6.10a. Additionally, GETbot had to negotiate a narrow curved passage (P) made up of obstacles with different shapes and sizes. It can be seen from figure 6.10b that ARM-ND+ was prone to oscillations (e.g. see the path next to points 1 - 4). Moreover, GETbot navigated close to obstacles (e.g. A and D), causing a considerable reduction in speed. By applying DWA-A+, GETbot was able to move at a relatively high speed, but ended up getting very close to obstacles, such as those labeled B, C, and E in figure 6.10c. Furthermore, once GETbot reached the starting area of

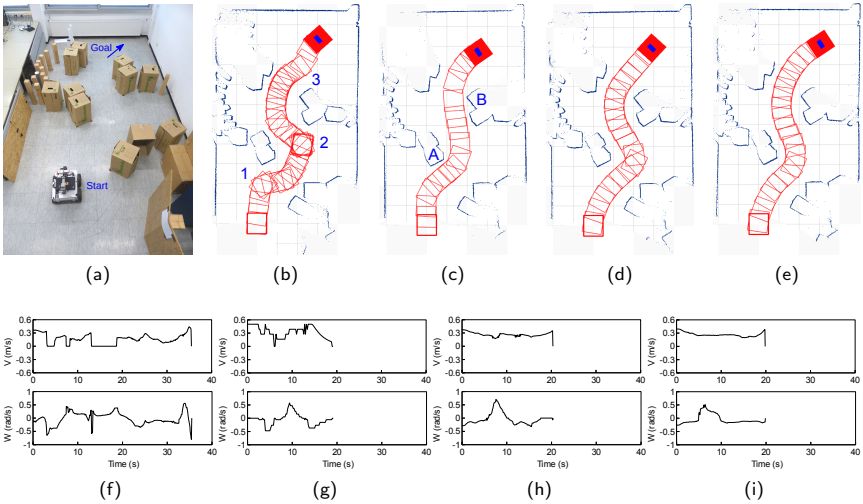


Figure 6.9: Scenario 1 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

passage P, it turned in-place for a while before it could move forward again. The in-place rotation was due to the execution of a recovery behavior [Epp16] which was invoked since DWA was unable to find a feasible control at that time. Using this behavior, the robot eventually got unstuck after clearing out space. The performance of TGF was relatively good (see figure 6.10d), however, in terms of smoothness, the generated trajectory could still be enhanced, particularly near points 1 - 3. This was possible by employing AG, as shown in figure 6.10e. We attribute this improvement in behavior to the consideration of the robot shape and kinematics. Figures 6.10f - 6.10i show the speed profiles for all methods.

6.5.3 Experiment 3

This experiment had three different complexities; first, several consecutive tight and curvy routes had to be negotiated to reach the goal, see figure 6.11a. Sec-

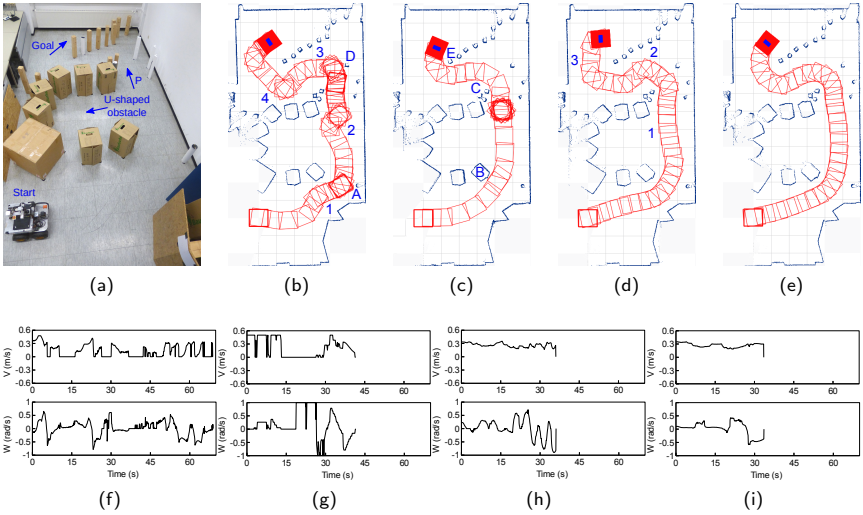


Figure 6.10: Scenario 2 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

ond, the obstacle course included small plastic/wooden poles with narrow gaps between them, through which GETbot could not fit. Finally, GETbot had to avoid obstacles forming a U-like shape during its course of navigation. By running ARM-ND+, the mission was aborted after reaching a certain point at which GETbot just kept on rotating right and left, as shown in figure 6.11b. It was discovered that these turn changes was a result of transforming the workspace into ARM, in which the *navigable* gap G1 vanished and the *non-navigable* gap G2 appeared. At that particular situation, GETbot rotated towards G2 until it was realized that G1 was free and G2 was blocked. This caused the robot to rotate back towards G1 but only to get caught again in this repetitive loop of failure. Using DWA-A*, GETbot navigated smoothly until it reached the starting area of passage P2, after which it rotated and navigated back across passage P1. This behavior was due to unexpected change in the costmap, which blocked the path planned towards P2, and thus, a new path was generated backwards.

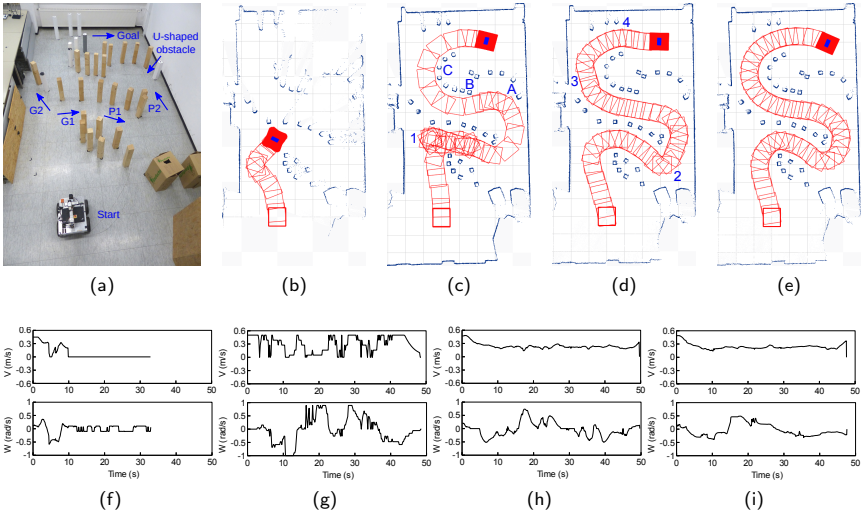


Figure 6.11: Scenario 3 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

The costmap is re-computed once GETbot reached point 1 (see figure 6.11c). Accordingly, the obstructed region appeared free again, which caused GETbot to turn and proceed towards the target. Similar to the previous scenarios, GETbot came close to obstacles at different points (e.g. A - C). Figures 6.11d and 6.11e show that TGF and AG managed to safely and smoothly drive GETbot towards the goal with roughly similar performance. However, a closer look at the paths next to points 2 - 4 verifies that, in terms of smoothness, AG was better than TGF. The speed profiles of all approaches are shown in figures 6.11f - 6.11i.

6.5.4 Experiment 4

For this scenario, we created the environmental structure shown in figure 6.12a, which resembles a Robocup rescue arena [PJK+14]. Notice that an oscillatory

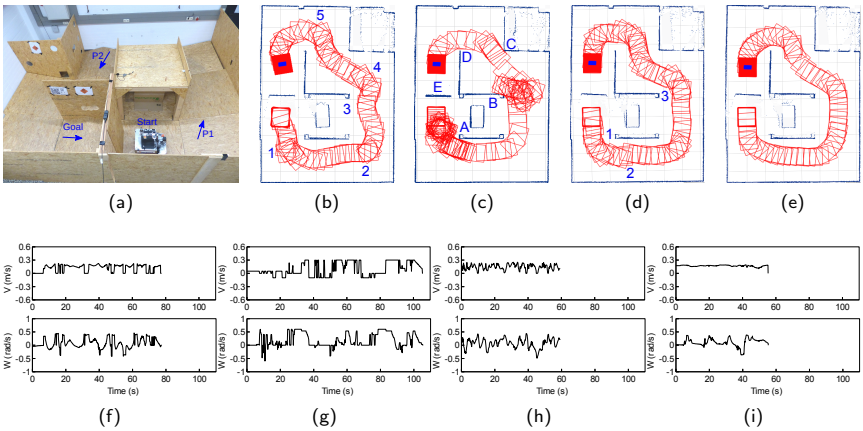


Figure 6.12: Scenario 4 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

motion over ramps may lead to wheel skidding, and hence it was important to achieve a high degree of smoothness in this experiment. Additionally, fast motion was hard to accomplish over ramps, especially employing DWA-A*. Hence, the maximum permitted speeds for this scenario were ($v = 0.3\,m/s,\ \omega = 0.6\,rad/s$). The goal was reached by employing all techniques, as can be seen from figures 6.12b - 6.12e. However, with ARM-ND+ the robot's motion was oscillatory, see for instance the generated trajectory near locations 1 - 5 in figure 6.12b. By using the DWA-A* algorithm, GETbot almost touched walls A, C and D, and collided with wall B, as can be seen from figure 6.12c. Furthermore, while entering the passages labeled P1 and P2, GETbot performed to-and-fro motion momentarily before it could move forward again. The root cause of this behavior was the wheel skidding on ramps, which made GETbot move closer to walls instead of following the planned path. This situation demanded a new path to be generated so that GETbot moves away from walls. The process of re-planning continued, until GETbot successfully escaped. It is important to mention that the front laserscanner was unable to detect wall E at the starting point. Therefore, the

robot turned and started to move towards the wall before it could discover that this region was obstructed. According to TGF, it showed smoother motion when compared to the ARM-ND+ method, but oscillations were not completely gone. See, for instance, the path next to the locations marked as 1 - 3 in figure 6.12d. The AG method was able to drive GETbot with improved smoothness and safety as compared to the other employed approaches. This visualization has been supported by plotting the speed profiles of all techniques in figures 6.12f - 6.12i.

6.5.5 Experiment 5

This scenario aimed to test the performance of AG in a cluttered environment occupied by a dynamic obstacle (a pedestrian). At the beginning of the mission, GETbot smoothly traversed the expected route through passages P1 and P2 until the line labeled L3 in figure 6.13a was crossed. At that moment, a pedestrian suddenly stepped in front of GETbot and blocked passage P2, as shown in figure 6.13b. GETbot reacted to this situation by navigating backwards towards P1, avoiding collision with the pedestrian. Once line L1 was crossed by the robot, the pedestrian stepped out of the track, leaving passage P2 free as depicted in figure 6.13c. This new situation was detected by GETbot and the decision was to move forward again, i.e. towards gap G1. As soon as GETbot reached the line labeled L2, box B was taken out of the arena, creating a new *navigable* gap labeled G2 in figure 6.13d. It was recognized that the new created gap was the closest to the goal, and therefore GETbot decided to pass through it and proceed towards the goal. The generated trajectory is shown in figure 6.13h. By running ARM-ND+ (figure 6.13e), GETbot came to a halt and was unable to complete the mission after the pedestrian stepped in front of it. The root cause was the inability of ARM-ND+ to detect the rear gap leading to P1. By applying DWA-A* (figure 6.13f), the feet of the pedestrian was touched by GETbot and it took quite some time to realize that a new gap G2 was created by removing the B box. This is attributed to the slow reaction of DWA-A* to the dynamic changes. Moreover, after having blocked passage P2, GETbot turned and navigated towards P1 without recognizing that passage P2 got free again. Moving back towards P2 was resumed only after traversing the entire P1 passage. Furthermore, it was

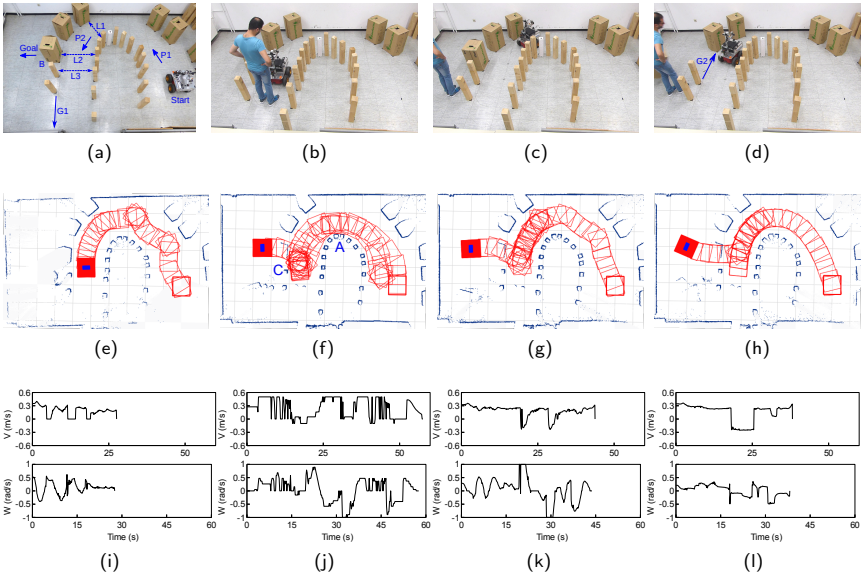


Figure 6.13: Scenario 5 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b) A pedestrian stepped in front of the robot. (c) The pedestrian stepped out of the track. (d) Box B was taken out of the arena, creating gap G2. (e-h) Paths generated by (e) ARM-ND+, (f) DWA-A*, (g) TGF, and (h) AG. (i-l) Speed profiles for (i) ARM-ND+, (j) DWA-A*, (k) TGF, and (l) AG.

observed that GETbot came close to obstacles at some points like A and C. By using TGF, GETbot turned on spot to move forwards instead of backwards, when the pedestrian stepped in and out of the track, see figure 6.13g. Besides the improved reaction to the pedestrian, AG achieved the best performance in terms of smoothness, as can be seen from figures 6.13e - 6.13h and 6.13i - 6.13l.

6.5.6 Experiment 6

For this scenario, we created a very complex and challenging arena. Figures 6.14a and 6.14b visualize two images of the arena which were captured from two

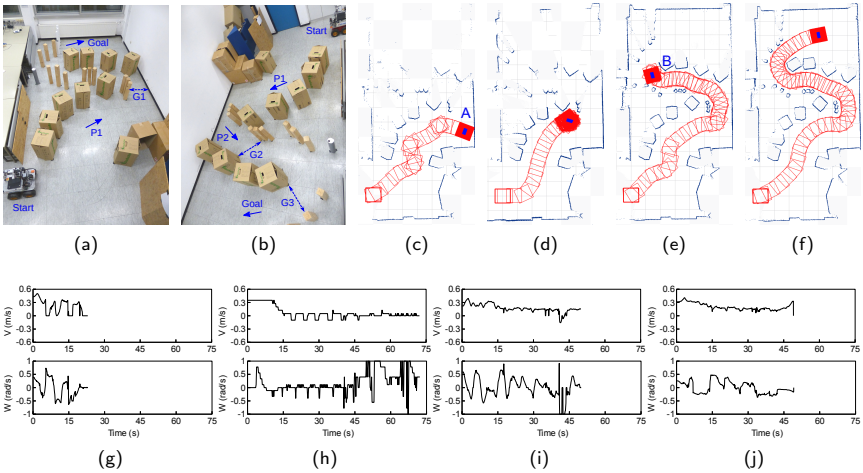


Figure 6.14: Scenario 6 (reprinted from [MFM18], with permission from Elsevier). (a, b) Environmental setup. (c-f) Paths generated by (c) ARM-ND+, (d) DWA-A*, (e) TGF, and (f) AG. (g-j) Speed profiles for (g) ARM-ND+, (h) DWA-A*, (i) TGF, and (j) AG.

different points of view. GETbot had to negotiate very tight and curvy openings, where at some points the maneuvering space was very limited. For example, gap G3 in figure 6.14b has a width of $0.63\,m$, which is less than the robot's diameter (evaluates to $0.71\,m$ approximately). Additionally, the environmental structure was made up of obstacles with different shapes and sizes. An additional challenge was the large difference in width between passages P1 and P2 (at gap G1). The goal was successfully reached by implementing the AG approach only. Although the environment was very complex and contained very narrow passages, AG managed to drive GETbot with a high degree of smoothness, as shown in figure 6.14f. By applying ARM-ND+, the mission was aborted after GETbot collided with the wall labeled A, coming to a full stop, as shown in figure 6.14c. The root cause was the inability of ARM-ND+ to find a navigable gap once GETbot approached the very narrow opening leading to passage P2 (at gap G1). Because of the speed gained before this particular situation, GETbot kept on moving until it crashed into the wall. By running DWA-A*, GETbot navigated smoothly until

it reached the starting area of passage P2, after which it kept on rotating and performing to-and-fro motion (see figure 6.14d). It was recognized that a path was properly planned across P2, but the DWA approach failed to track it. The TGF-controlled GETbot managed to traverse most of the route, but ended up colliding with obstacle B after rotating in-place (see figure 6.14e). This rotation aimed to drive GEbot back towards passage P1, since at that time TGF was unable to detect gap G3 whose width was less than the robot's diameter (TGF ignores the robot shape and its kinematics). Notice that we limited the value of R to 0.33 m to get this performance. By using the actual value ($R = 0.355$ m), the robot could not even pass through gap G2 since its width was $0.68\,m < 2R$. It is worth mentioning that, by further decreasing the radius ($R < 0.33\,m$), the robot pushed over the wooden pole creating gap G2 and didn't complete the mission. The speed profiles of all approaches are shown in figures 6.14g - 6.14j.

6.5.7 Experiment 7

In this experiment, we tested the capability of the proposed AG approach to drive a mobile robot in unknown environments occupied by a crowd of moving persons. The experiment was conducted in building P1 at the university of Paderborn, with the help of several students working in our lab[8]. The students were informed to randomly move in the working area of the robot, trying to close its way sometimes. The objective was to resemble real world scenarios where the robot may share the same environment with humans. The mission was started at the main entrance of the P1 building and the robot had to pass through the connection between buildings P1 and P7, see figure 6.15a. The goal was given at the end of the connection near the entrance to building P7, as shown in figure 6.15b. We observed that the robot was able to react on time avoiding the students who stepped in to close its way. For example, in the first part of the experiment the path to the goal was free, and hence the robot moved directly towards it, see figure 6.15c. Once the student marked as S1 moved across the corridor, the robot escaped by moving towards the free gap marked as G1 between students S1 and S2, see figure 6.15d. Student S1 kept moving and climbed the

[8] A video of this experiment is available at: `https://getwww.uni-paderborn.de/research/videos/dynamic-obstacles`

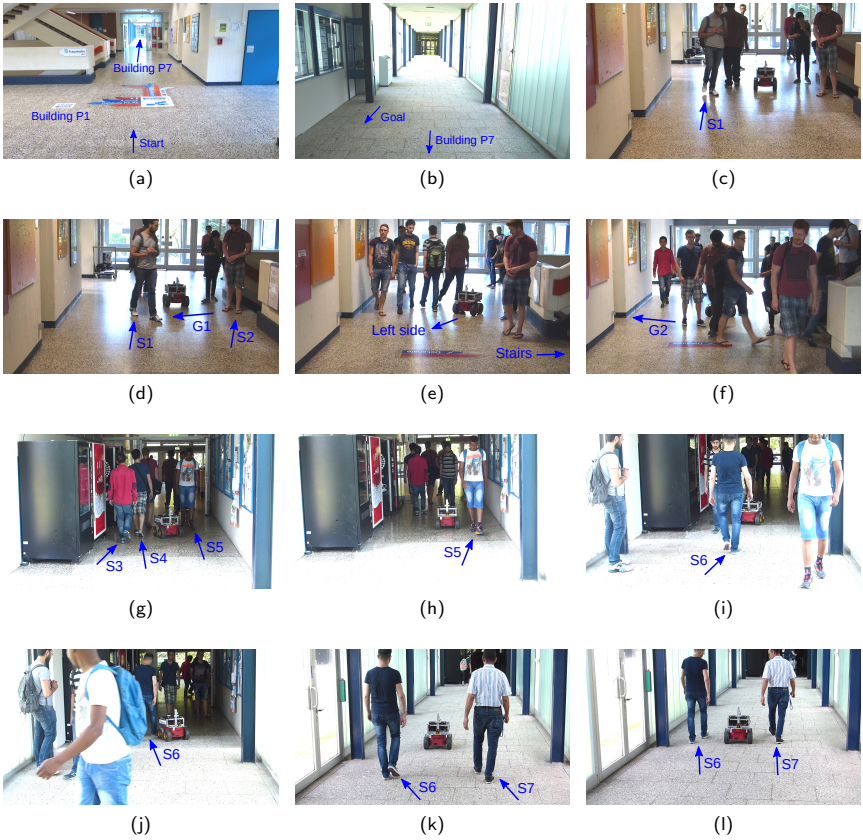


Figure 6.15: Experiment 7. (a, b) Environmental setup, the robot had to move through the connection between buildings P1 and P7. (c-l) Snapshots of the experiment taken at different locations, showing that the robot was able to react on time avoiding the students who stepped in to close its way to the goal.

stairs, while student S2 started to move across the corridor as shown in figure 6.15e. Consequently, the robot modified its orientation and started to move towards the free area to the left-hand side as shown in figure 6.15e. At a later time, most of the corridor was occupied by students, and thus the robot escaped

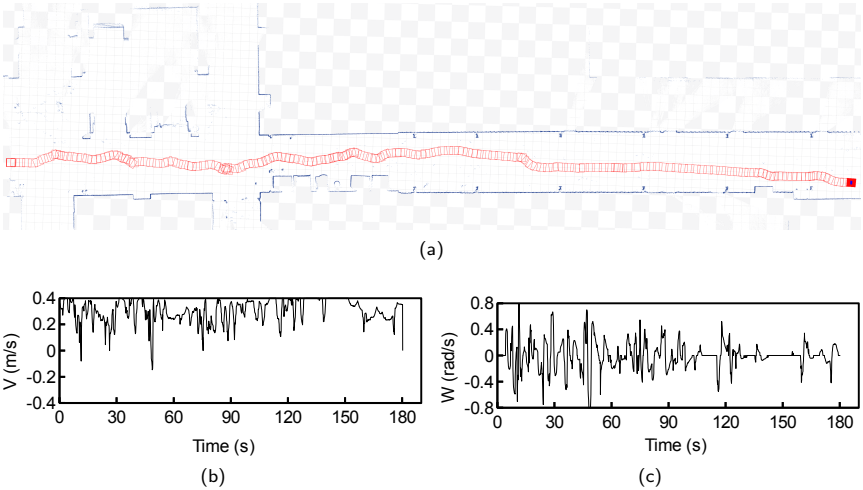


Figure 6.16: Trajectory followed by the robot (a) and the recorded motion commands against the time (b, c) corresponding to experiment 7.

towards gap G2, the only free gap at that moment (see figure 6.15f). Figure 6.15g shows another situation where the robot was moving towards the right-hand side, avoiding students S3 and S4. At that time, student S5 approached the robot from behind. The situation was detected and the orientation of the robot was modified accordingly, see figure 6.15h. A similar behavior can also be seen in figures 6.15i and 6.15j, where student S6 approached the robot from front this time. Figures 6.15k and 6.15l show a situation where students S6 and S7 were moving to both sides of the robot, but not closing its way to the goal. Hence, the robot kept moving directly towards its goal. Figure 6.16 shows the trajectory followed by the robot and the motion commands against the time.

6.6 Evaluation and Discussion

The proposed "AG approach" is evaluated on the basis of the metrics presented in chapter 5. Moreover, its performance is compared to that of the techniques discussed in section 6.5. The results obtained from experiments 1 - 6 (presented

Table 6.1: Performance assessment results of the proposed "AG approach" for experiments 1 - 6 from section 6.5 (reprinted from [MFM18], with permission from Elsevier).

Exp.	Method	T_{tot}	P_{len}	C_{chg}	Z_{ω}	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}
1	ARM-ND+	35.4	5.89	118.62	10	2.28	17.52	0.95	2.90	252.28
	DWA-A*	**19.0**	**4.91**	0.53	**2**	4.87	6.00	**0.56**	2.23	169.34
	TGF	20.3	5.26	0.67	**2**	0.22	0.98	0.91	0.85	134.44
	AG	19.8	5.24	**0.28**	**2**	**0.02**	**0.89**	0.95	**0.55**	**128.29**
2	ARM-ND+	68.8	9.81	181.15	18	6.82	16.44	1.92	9.34	543.02
	DWA-A*	41.3	**8.03**	161.22	**3**	11.05	55.77	**1.16**	4.91	375.81
	TGF	36.0	9.14	1.29	9	0.10	2.57	2.16	1.65	195.97
	AG	**33.5**	8.71	**0.38**	**3**	**0.04**	**1.77**	1.71	**0.76**	**194.19**
3	ARM-ND+	fail	fail	fail	fail	fail	fail	fail	fail	fail
	DWA-A*	48.4	12.95	51.52	5	14.42	32.78	2.40	8.96	510.24
	TGF	49.6	11.45	1.97	10	0.09	**2.09**	2.25	1.77	**273.15**
	AG	**47.5**	**11.10**	**0.39**	**3**	**0.04**	2.20	**1.84**	**1.32**	289.68
4	ARM-ND+	77.3	10.40	119.60	32	2.13	12.96	1.42	6.00	340.61
	DWA-A*	105.4	13.67	103.18	26	7.52	20.69	1.55	11.05	17415.50
	TGF	67.3	10.01	31.65	26	1.02	2.40	1.12	4.16	260.22
	AG	**55.3**	**9.29**	**0.67**	**8**	**0.03**	**1.53**	**1.31**	**0.86**	**253.06**
5	ARM-ND+	NC	NC	NC	NC	NC	NC	NC	NC	NC
	DWA-A*	57.4	13.92	61.17	6	24.04	39.87	2.60	11.92	940.37
	TGF	43.7	8.85	12.39	12	3.55	23.34	2.53	4.49	399.19
	AG	**37.6**	**8.24**	**0.64**	**5**	**3.48**	**4.38**	**1.16**	**2.38**	**209.72**
6	ARM-ND+	fail	fail	fail	fail	fail	fail	fail	fail	fail
	DWA-A*	fail	fail	fail	fail	fail	fail	fail	fail	fail
	TGF	fail	fail	fail	fail	fail	fail	fail	fail	fail
	AG	**49.1**	**9.74**	**0.76**	**7**	**0.28**	**3.50**	**1.84**	**2.23**	**681.60**

in section 6.5) are shown in table 6.1. It is interesting to notice that the ARM-ND+ method yields the worst results in all experiments except experiment 4, in which the DWA-A* approach was the worst. The poor performance of ARM-ND+ is attributed to two causes: first, transforming the workspace into the "Arc Reachable Manifold" (ARM) may hinder finding navigable gaps like the gap labeled G1 in experiments 3 and 6 (only "admissible" gaps are considered navigable in ARM). Second, ARM-ND+ builds upon the ND+ method which tends to generate oscillations and instability, as discussed in chapter 5.

In fact, the DWA-A* approach is a hybrid system which employs both local (DWA) and global (A* algorithm) planners, whereas ARM-ND+, TGF, and AG are pure reactive methods. Therefore, a direct comparison between DWA-A* and

the other approaches could be unfair. Despite this fact, however, the presented results show that TGF and AG outperform DWA-A*, especially in terms of the C_{chg}, R_{obs}, J_{acc}, S_{tng}, and ζ_{acc} metrics. This is attributed to the tendency of the A* algorithm to generate a path near obstacles. Apparently, following this path results in a higher R_{obs} value and makes the robot get close to corners, which in turn may hide some parts of the environment from the local planner (DWA). At that particular situation, the robot may face unforeseen obstacles, requiring a rapid change in speed (reflected by high J_{acc} and S_{tng} values) and heading (reflected by high C_{chg} and ζ_{acc} values). On the other hand, AG and TGF both tend to maintain a safe distance between obstacles and the robot's footprint (see section 6.3.2). Furthermore, the smoothness of the generated trajectories is increased by performing collision avoidance based on all obstacles falling between the robot and the closest gap, achieving a kind of look-ahead. An additional point which needs to be mentioned here is the fact that DWA-A* is in principle able to drive the robot at higher speeds with shorter distances. This is attributed to the usage of A* to look for a path that is often short compared to that of the other methods. This path is then followed by DWA, which accounts for the dynamics of the vehicle. However, invoking the recovery behavior and re-planning the path in experiments 2 and 3 canceled this benefit. The situation was even more difficult in experiment 4 when the robot slipped towards walls while entering passages P1 and P2 and took quite some time before it could move forward again.

When comparing the performance of AG with that of TGF, it can be observed that AG achieves better performance, especially in terms of J_{acc}, C_{chg}, and Z_{ω}. More important, AG successfully drove the robot through the obstacle structure of experiment 6, but TGF did not. The improved performance of the AG approach is attributed to the consideration of the robot shape and kinematic constraints. The TGF method, on the other hand, ignores these constrains, which may hinder finding feasible motions or reduce performance. In fact, the output of TGF is a direction solution, with which the robot's heading must be aligned. A closer look at figures 6.9 - 6.14 shows that the TGF-controlled robot often had to reduce speed (approached zero sometimes) and perform a sharp turn (rotated in-place sometimes) to face the direction it follows. Apparently, this explains the higher values of J_{acc}, C_{chg}, and Z_{ω} compared to those of the AG method. It is

significant to note that GETbot is roughly square and works in a skid-steering mode. If it had a complicated shape or if it was a car-like robot, TGF could have failed in more experiments or it would have had less performance, particularly in scenarios demanding high maneuverability. It is worth to mention that implementing the TGF method took a lot of time to achieve this relatively good performance. This is due to that fact that aligning the robot's heading with the holonomic output is not straightforward and based on experimentation.

In addition to the results presented above, we highlight other advantages and features of the AG approach. First, similar to the TGF method, AG has only one parameter $\mathrm{d_{safe}}$ that is easy to determine. It specifies how far the robot stays away from the obstacles' boundary (as long as there is a free space), providing a trade-off between efficiency and safety. Hence, it should be seen as a feature rather than a limitation. In principle, we may totally discard $\mathrm{d_{safe}}$ and set $\mathrm{d_s(g)}$ in Eq. (3.5) to $\frac{1}{2}w(\mathrm{g})$ always, but this may lead to a longer path if the gap is too wide. In all experiments presented in section 6.5, the value of $\mathrm{d_{safe}}$ was set to $2R$. ARM-ND+ has another parameter D_s, which defines a region around the robot, once occupied, the robot's trajectory is adjusted building upon the APF concept. Determining D_s is not straightforward and has a significant influence on the performance of the algorithm. According to the DWA-A* approach, it has many parameters to tune, where finding a good parameter setting is environment dependent and consumes much time and effort [MFM18].

An additional advantage of the AG approach is its robustness against the environmental changes. It managed to successfully drive GETbot in all conducted tests, either those discussed in section 6.5 or others. Furthermore, each time a test was repeated, the obtained results were almost the same. The other tested methods presented different degrees of sensitivity to the environmental structure. For example, ARM-ND+ managed to drive GETbot in simple scenarios only. The DWA-A*-controlled GETbot generated a different path each time a test was repeated. TGF showed better robustness than DWA-A* and ARM-ND+ as it successfully drove GETbot in most of the scenarios. However, each time an experiment was repeated, the differences in results were higher than those corresponding to the AG method. Perhaps, finding scenarios where these three methods succeed was the most time consuming and frustrating part of the exper-

iments. Notice that the complexity of the environment in experiments 1, 2, and 5 was reduced, so that the expected route could be traversed by all techniques.

Another yet important feature of the "AG approach" is the simplicity of the problem formulation and implementation. For example, unlike TGF, it is easy to replicate the presented experiments by directly implementing the algorithm without the need to adapt the output to cope with the vehicle constraints. Additionally, unlike ARM-ND+, it is not necessary to map the workspace into a new manifold, which is not trivial to model and can be computationally expensive. Last but not least, unlike DWA, AG is able to guide a robot through complex and tight passages without integrating it with a path planner. A higher-level planner is only necessary to avoid cyclic loops or global trap situations [MFM18].

6.7 Conclusions

We have presented a new concept, the "admissible gap" (AG), which addresses the question of whether a given gap is traversable by performing a collision-free motion control, that respects the shape and vehicle constraints. By employing this concept, a new collision avoidance method has been developed and implemented. Compared to existing techniques, the new approach achieves an outstanding performance in cluttered scenarios. This has been possible by directly obeying the vehicle constraints rather than adapting a holonomic-based solution. A key idea of the AG approach is the creation of an "admissible gap", which serves as a bridge to the opening closest to the goal (the closest gap). To this end, a new methodology for traversing gaps has been proposed in such a way that the vehicle constraints are respected. This methodology provides a compromise between safety and efficiency. Our approach is directly applied to the workspace without having to construct an abstraction layer. Additionally, this chapter introduces a new procedure for finding out gaps. The method can be applied to full or limited field of view sensors. Moreover, it discards useless gaps, reducing the possibility of oscillation and improving the stability of navigation. Experimental results along with performance assessment in highly cluttered scenarios demonstrate that the proposed AG approach outperforms existing state-of-the-art methods in terms of smoothness, safety, efficiency, and robustness.

7 Conclusions and Future Work

In this chapter, the contributions of the research work presented in this thesis are summarized, pointing out some concluding remarks, and subsequently, several issues that could be investigated in future research are discussed.

7.1 Conclusions

The work presented in this thesis contributes to the field of autonomous mobile robot navigation. In particular, it addresses the problem of reactive collision avoidance in very dense, complex, and cluttered environments which is one of the most significant and challenging problems in mobile robotics. Among the wide variety of reactive collision avoidance techniques, the Nearness-Diagram (ND) Navigation is a well-known and effective approach that deals with this problem. However, experiments demonstrated that ND-based navigation is prone to several problems, namely oscillatory motion, risk of collision in narrow spaces, unreasonable deviations towards free areas, and the tendency to generate slower trajectories and longer paths. The first objective of this work was to develop a new reactive collision avoidance approach that avoids the above mentioned limitations. To this end, two methods have been proposed, the "Safe Gap" (SG) and the "Tangential Gap Flow" (TGF) navigation. The second objective of this work was to account for the exact shape and kinematic constraints, improving the navigation performance. For this purpose, another approach has been proposed building upon a new concept, called the "admissible gap" (AG). In the following, we discuss the main features and contributions of these methods.

The SG method improves the robot's behavior in dense and cluttered environments by generating smoother, faster, and safer avoidance maneuvers and by

avoiding irrational deflections towards free spaces. The key idea behind this improvement is the incorporation of an additional step in analyzing the sensory data, locating a virtual gap in a collision free area, called a "safe gap" [MFM13b]. The location of this gap is determined based on its opening angle and the configuration of the goal, providing a smoother and safer bridge between obstacle avoidance and goal approach. Unlike the ND-based methods, SG does not require the safe distance parameter, and thus saves the parameter tuning overhead.

The TGF approach generates smoother and much more stable trajectories compared to those generated by the SG method, especially in unstructured narrow spaces. Moreover, TGF helps in reducing turn changes occurring in tight gaps, where the robot may switch between avoiding obstacles located to the right or left of the heading direction. This has been possible by considering the clearance to obstacles on both sides of the heading direction and by computing the steering angle in such a way that all surrounding threats are taken into account, not simply the nearest one. Obstacle avoidance is, in general, based on two concepts; the "tangential" and "gap flow" navigation [MJFM13] [MFM15] [MFM16] [MFM17]. The key idea behind both concepts is the usage of the data acquired from the environmental structure in computing the avoidance trajectory. Using the "tangential navigation", the robot moves tangential to the obstacles boundary. The "gap flow navigation" smoothly points the robot towards the free area between obstacles. In both concepts, avoiding collisions and approaching the target are simultaneously performed. An additional contribution of TGF is the development of motion commands that drive a mobile robot towards a given target in such away that the stability of the system is guaranteed in the Lyapunov sense.

The AG concept addresses the question of whether a given gap is traversable by performing a collision-free motion control that respects the shape and vehicle constraints [MFM18]. This concept has been successfully employed to develop a collision avoidance approach, that achieves an outstanding performance in cluttered scenarios. Unlike existing methods, AG is directly applied to the workspace and considers the exact shape and kinematics, achieving a safer and more accurate solution. A key idea is the creation of an "admissible gap", which serves as a bridge obeying the vehicle constraints, once traversed, the robot makes progress towards the target [MM17]. To this end, a new methodology for traversing gaps

has been proposed in such a way that the vehicle constraints are respected. This methodology provides a compromise between safety and efficiency. Additionally, AG proposes a new strategy for extracting gaps which works for both full and limited FOV sensors. Within this strategy, useless gaps are discarded, reducing the possibility of oscillation and improving the stability of navigation.

In order to verify that the proposed approaches comply with the goals of this work, several simulations and experiments in very dense and complex environments were conducted. Moreover, the performance of the proposed approaches was evaluated and compared to that of existing state-of-the-art methods based on the aforementioned drawbacks. In addition to the experiments presented here, many tests were carried out while preparing for and participating in several RoboCup Rescue Robot League competitions (2012- 2016). Up to our knowledge, there is no other method in the literature that provides experimental results in very hard scenarios similar to those presented in this thesis.

Our experience showed that the performance of the proposed approaches is stable against changes of the environmental structure. The ND-based methods, on the other hand, seem to be more sensitive to the obstacle distribution. For instance, whenever a robot navigates through a relatively wide area that is located between two narrow passages, it tends to deviate towards the free space performing a sharp turn, followed by another turn towards the opposite side trying to enter the second narrow passage. These sudden turn changes may lead to oscillations and instability, which can be unsafe if the robot is passing through a narrow passage or if it is navigating at a relatively high speed. Furthermore, this behavior makes the robot follow longer paths owing to the unnecessary deflection towards the free space. The main reason behind this drawback is the computation of the avoidance trajectory regardless of the gap selected for navigation. Another example could be observed in narrow corridors, where the robot usually performs successive turn changes. This problem is basically inherited from the APF concept; within this concept, approaching one side of a tight opening generates a strong repulsion force, causing a sharp turn that takes the robot away from obstacles. Apparently, this behavior is repeated with the other side. It is worth to mention that this problem was clearly visible during our participation in the RoboCup Rescue Robot League 2012. At that time, we were using an ND variant [MFMJ10] to

control our robot GETbot, where the motion was unstable due to turn changes, leading to wheel slippage over ramps and eventually a collision.

It is worth mentioning that the work presented in this thesis does not address the commonality in features between the proposed approaches and the ND-based navigation. These commonalities include the computational efficiency, avoiding the tedious parameter tuning, and the ability to drive a robot towards obstacles when necessary. For more information about these features and their effect on the performance of the system, the reader is directed to [MM04] and [MLL08].

It can be noticed that, in this thesis, the navigation problem has been addressed based on the information obtained from the current sensor readings only. This may arise some problems due to locality (e.g. a cyclic motion). To deal with this issue, the proposed solutions are incorporated into a hybrid system including a planner, as has been addressed in [MMSA01] and [SB02]. By this means, locality problems are avoided while still being able to achieve real-time performance.

7.2 Future Work

The work carried out in this thesis could be further extended in different directions, some of which are presented next.

The proposed navigation methods have been developed assuming a flat arena. It is of interest to consider the geometric properties of the terrain. One possibility could be to improve the gap analysis so that the gap that ensures safer navigation is selected rather than the one closest to the goal. In addition, the motion commands should be adapted by considering the slope and roughness of the terrain. To this end, a 3D map is necessary to model the environment, which can be obtained by a 3D SLAM module.

Another extension could be to consider the future trajectory of obstacles in performing the avoidance maneuver. For this purpose, a module detecting and tracking all visible objects in the scene is required (scene understanding). In the context of reactive navigation, one possibility could be to incorporate the concept of Inevitable Collision States [FA04] in the admissible gap concept.

Bibliography

[ACBB95] Aicardi, Michele; Casalino, Giuseppe; Bicchi, Antonio; Balestrino, Aldo: Closed Loop Steering of Unicycle-like Vehicles via Lyapunov Techniques. In: *IEEE Robotics and Automation Magazine*, vol. 2, 1995, pp. 27–35

[AI08] Andoni, Alexandr; Indyk, Piotr: Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In: *Commununication of the ACM*, vol. 51, 2008, no. 1, pp. 117–122

[And08] Andreasson, Henrik: *Local Visual Feature based Localisation and Mapping by Mobile Robots*, Örebro University, Diss., 2008

[Ark98] Arkin, Ronald C.: *Behavior-based Robotics*. Cambridge, London : The MIT Press, 1998 (Intelligent Robots and Autonomous Agents)

[AW04] An, Dong; Wang, Hong: VPH: A New Laser Radar Based Obstacle Avoidance Method for Intelligent Mobile Robots. In: *Proceedings of the 5th World Congress on Intelligent Control and Automation*. Hangzhou, China, 2004, pp. 4681–4685

[BB15] Bareiss, Daman; Berg, Jur: Generalized Reciprocal Collision Avoidance. In: *The International Journal of Robotics Research*, vol. 34, 2015, April, no. 12, pp. 1501 – 1514

[BDD+14] Babinec, Andrej; Duchon, Frantisek; Dekan, Martin; Paszto, Peter; Kelemen, Michal: VFH*TDT (VFH* with Time Dependent Tree): A New Laser Rangefinder Based Obstacle Avoidance Method Designed for Environment with Non-static Obstacles. In: *Robotics and Autonomous Systems*, vol. 62, 2014, no. 8, pp. 1098–1115

[BGPG17] Bounini, Farid; Gingras, Denis; Pollart, Herve; Gruyer, Dominique: Modified artificial potential field method for online path planning applications. In: *IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA, 2017, pp. 180–185

[BK89] Borenstein, Johann; Koren, Yorem: Real-Time Obstacle Avoidance for Fast Mobile Robots. In: *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, 1989, September, no. 5, pp. 1179–1187

[BK91] Borenstein, Johann; Koren, Yoram: The Vector Field Histogram - Fast Obstacle Avoidance For Mobile Robots. In: *IEEE Transactions on Robotics and Automation*, vol. 7, 1991, June, no. 3, pp. 278–288

[BK99] Brock, Oliver; Khatib, Oussama: High-Speed Navigation using the Global Dynamic Window Approach. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Detroit, MI, USA, May 1999, pp. 341–346

[BL91] Barraquand, Jerime; Latombe, Jean C.: Robot Motion Planning: A Distributed Representation Approach. In: *The International Journal of Robotics Research*, vol. 10, 1991, no. 6, pp. 628–649

[BLM08] Berg, Jur; Lin, Ming C.; Manocha, Dinesh: Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Pasadena, CA, May 2008, pp. 1928 – 1935

[BLOD96] Bemporad, Alberto; Luca, Alessandro D.; Oriolo, Giuseppe; De, Alessandro: Local Incremental Planning for a Car-Like Robot Navigating Among Obstacles. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Minneapolis, Minnesota, April 1996, pp. 1205–1211

[BSAP18] Baldi, Tommaso L.; Scheggi, Stefano; Aggravi, Marco; Prattichizzo, Domenico: Haptic Guidance in Dynamic Environments Using Optimal Reciprocal Collision Avoidance. In: *IEEE Robotics and Automation Letters*, vol. 3, 2018, no. 1, pp. 265–272

[BUVRJ17] Ballesteros, Joaquin; Urdiales, Cristina; Velasco, Antonio B. M.; Ramos-Jimenez, Gonzalo: A Biomimetical Dynamic Window Approach to Navigation for Collaborative Control. In: *IEEE Transactions on Human-Machine Systems*, vol. 47, 2017, no. 6, pp. 1123–1133

[Cao04] Cao, Peter M.: *Autonomous Runway Soil Survey System with the Fusion Of Global and Local Navigation Mechanism*, University of Cincinnati, Diss., March 2004

[CLH+05] Choset, Howie; Lynch, Kevin M.; Hutchinson, Seth; Kantor, George A.; Burgard, Wolfram; Kavraki, Lydia E.; Thrun, Sebastian: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA : MIT Press, 2005

[CML+15] Chiang, Hao; Malone, Nick; Lesser, Kendra; Oishi, Meeko; Tapia, Lydia: Path-Guided Artificial Potential Fields with Stochastic Reachable Sets for Motion Planning in Highly Dynamic Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, USA, May 2015, pp. 2347 – 2354

[CN09] Calisi, Daniele; Nardi, Daniele: Performance Evaluation of Pure-motion Tasks for Mobile Robots with Respect to World Models. In: *Autonomous Robots*, vol. 27, 2009, September, no. 4, pp. 465–481

[DB08] Durham, Joseph W.; Bullo, Francesco: Smooth Nearness-Diagram Navigation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. France, September 2008, pp. 690–695

[Dij59] Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. In: *Numerische Mathematik*, vol. 1, 1959, no. 1, pp. 269–271

[DK02] DeSouza, Guilherme N.; Kak, Avinash C.: Vision for Mobile Robot Navigation: A Survey. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, 2002, no. 2, pp. 237–267

[DS17] Demir, Mustafa; Sezer, Volkan: Improved Follow the Gap Method for obstacle avoidance. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. Munich, Germany, July 2017, pp. 1435–1440

[DSS+13] Dakulovic, Marija; Sprunk, Christoph; Spinello, Luciano; Petrovic, Ivan; Burgard, Wolfram: Efficient Navigation for Anyshape Holonomic Mobile Robots in Dynamic Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, November 2013, pp. 2644–2649

[DTMD10] Dolgov, Dmitri; Thrun, Sebastian; Montemerlo, Michael; Diebel, James: Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments. In: *The International Journal of Robotics Research*, vol. 29, 2010, no. 5, pp. 485–501

[Elf86] Elfes, Alberto: A Sonar-Based Mapping and Navigation System. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA USA, April 1986, pp. 1151–1156

[Epp16] Eppstein, Eitan: *Rotate Recovery*. http://wiki.ros.org/rotate_recovery/ (accessed: 28.05.2019), January 2016

[FA04] Fraichard, Thierry; Asama, Hajime: Inevitable Collision States - a Step Towards Safer Robots? In: *Advanced Robotics*, vol. 18, 2004, no. 10, pp. 1001–1024

[FBL94] Feiten, Wendelin; Bauer, Rudolf; Lawitzky, Gisbert: Robust Obstacle Avoidance in Unknown and Cramped Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Diego, CA, USA, May 1994, pp. 2412–2417

[FBT97] Fox, Dieter; Burgard, Wolfram; Thrun, Sebastian: The Dynamic Window Approach to Collision Avoidance. In: *IEEE Robotics and Automation Magazine*, vol. 4, 1997, no. 1, pp. 23–33

[FNA09] Fahimi, Farbod; Nataraj, C.; Ashrafiuon, Hashem: Real-time Obstacle Avoidance for Multiple Mobile Robots. In: *Robotica*, vol. 27, 2009, no. 2, pp. 189–198

[FPV+08] Ferreira, Andre; Pereira, Flavio G.; Vassallo, Raquel F.; Filho, Teodiano Freire B.; Filho, Mario S.: An Approach to Avoid Obstacles in Mobile Robot Navigation: The Tangential Escape. In: *SBA. Sociedade Brasileira de Automatica*, vol. 19, 2008, no. 4, pp. 395–405

[Fra07] Fraichard, Thierry: A Short Paper About Motion Safety. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Roma, Italy, May 2007, pp. 1140 – 1145

[Fre12] Freeman, Philip: *Minimum Jerk Trajectory Planning for Trajectory Constrained Redundant Robots*, Washington University in St. Louis, Diss., May 2012

[FS98] Fiorini, Paolo; Shiller, Zvi: Motion Planning in Dynamic Environments using Velocity Obstacles. In: *The International Journal of Robotics Research*, vol. 17, 1998, July, no. 7, pp. 760–772

[FS06] Ferguson, Dave; Stentz, Anthony: Using Interpolation to Improve Path Planning: The Field D* Algorithm. In: *Journal of Field Robotics*, vol. 23, 2006, February, no. 2, pp. 79–101

[FSBD04] Fernandez, J. L.; Sanz, R.; Benayas, J. A.; Diaguez, A. R.: Improving Collision Avoidance for Mobile Robots in Partially Known Environments: the Beam Curvature Method. In: *Robotics and Autonomous Systems*, vol. 46, 2004, no. 4, pp. 205–219

[GC02] Ge, S. S.; Cui, Y. J.: Dynamic Motion Planning for Mobile Robots using Potential Field Method. In: *Autonomous Robots*, vol. 13, 2002, pp. 207–222

[Gc10] Gerkey, Brian; contributors: *Karto Mapping Library*. http://wiki.ros.org/karto/ (accessed: 28.05.2019), April 2010

[GPPK13] Goretkin, Gustavo; Perez, Alejandro; Platt, Robert; Konidaris, George: Optimal Sampling-based Planning for Linear-quadratic Kinodynamic Systems. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Germany, May 2013, pp. 2429–2436

[GSR09] Gal, Oren; Shiller, Zvi; Rimon, Elon: Efficient and Safe On-line Motion Planning in Dynamic Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 88–93

[GW03] Guo, Zhihua Qu Y.; Wang, Jing: A New Performance-Based Motion Planner for Nonholonomic Mobile Robots. In: *Proc. of International Workshop on Performance Metrics for Intelligent Systems Workshop (PerMIS)*. Gaithersburg, MD, September 2003, pp. 1–8

[Hau12] Hauser, Kris: On Responsiveness, Safety, and Completeness in Real-time Motion Planning. In: *Autonomous Robots*, vol. 32, 2012, no. 1, pp. 35–48

[Hau15] Hauser, Kris: Lazy Collision Checking in Asymptotically-optimal Motion Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, USA, May 2015, pp. 2951 – 2957

[HE02] Hussein, Abdulla M.; Elnagar, Ashraf: Motion Planning using Maxwell's Equations. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Lausanne, Switzerland, October 2002, pp. 2347–2352

[HNR68] Hart, Peter; Nilsson, Nils; Raphael, Bertram: A formal Basis for the Heuristic Determination of Minimum Cost Paths. In: *IEEE Transactions on Systems Science and Cybernetics*, vol. SSC-4, 1968, no. 2, pp. 100–107

[Hoy14] Hoy, Michael C.: *Methods for Collision-Free Navigation of Multiple Mobile Robots in Unknown Cluttered Environments*, Cornell University, Diss., January 2014

[HP16] Hegde, Rashmi; Panagou, Dimitra: Multi-Agent Motion Planning and Coordination in Polygonal Environments using Vector Fields and Model Predictive Control. In: *European Control Conference (ECC)*. Aalborg, Denmark, June 2016, pp. 1–6

[IO02] Im, Kwang-Young; Oh, Se-Young: An Extended Virtual Force Field Based Behavioral Fusion with Neural Networks and Evolutionary Programming for Mobile Robot Navigation. In: *IEEE Transactions on Evolutionary Computation*, vol. 6, 2002, no. 4, pp. 413–419

[JKWG15] Jin, Jingfu; Kim, Yoon; Wee, Sung; Gans, Nicholas: Decentralized Cooperative Mean Approach to Collision Avoidance for Nonholonomic Mobile Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*. USA, May 2015, pp. 35 – 41

[JSCP15] Janson, Lucas; Schmerling, Edward; Clark3, Ashley; Pavone, Marco: Fast marching tree: A Fast Marching Sampling-based Method for Optimal Motion Planning in Many Dimensions. In: *The International Journal of Robotics Research*, vol. 34, 2015, no. 7, pp. 883–921

[JXK10] Jie, Dong; Xueming, Ma; Kaixiang, Peng: IVFH*: Real-time Dynamic Obstacle Avoidance for Mobile Robots. In: *International Conference on Control, Automation, Robotics and Vision*. Singapore, December 2010, pp. 844–847

[KB91] Koren, Yoram; Borenstein, Johann: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Sacramento, CA, April 1991, pp. 1398–1404

[KF10] Karaman, Sertac; Frazzoli, Emilio: Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods. In: *49th IEEE Conference on Decision and Control*. Atlanta, GA, USA, December 2010, pp. 7681–7687

[KF11] Karaman, Sertac; Frazzoli, Emilio: Sampling-based Algorithms for Optimal Motion Planning. In: *The International Journal of Robotics Research*, vol. 30, 2011, June, no. 7, pp. 846–894

[KF13] Karaman, Sertac; Frazzoli, Emilio: Sampling-based Optimal Motion Planning for Non-holonomic Dynamical Systems. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany, May 2013, pp. 5041–5047

[KFT+08] Kuwata, Yoshiaki; Fiore, Gaston A.; Teo, Justin; Frazzoli, Emilio; How, Jonathan P.: Motion Planning for Urban Driving using RRT. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nice, France, September 2008, pp. 1681–1686

[Kha86] Khatib, Oussama: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In: *International Journal of Robotics Research*, vol. 5, 1986, April, no. 1, pp. 90–98

[Kha14] Khalil, Hassan K.: *Nonlinear Control*. Pearson Education, 2014

[KJIF06] Kunchev, Voemir; Jain, Lakhmi C.; Ivancevic, Vladimir; Finn, Anthony: Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. In: *Proceedings of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems* Bd. 4252, Springer, 2006 (Lecture Notes in Computer Science), pp. 537–544

[KL00] Kuffner, James J.; Lavalle, Steven M.: RRT-Connect: An Efficient Approach to Single-query Path Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA, USA, April 2000, pp. 995–1001

[KL02] Koenig, S.; Likhachev, M.: Incremental A*. In: *Proceedings of the Neural Information Processing Systems*, MIT Press, 2002, pp. 1539–1546

[KL05] Koenig, Sven; Likhachev, Maxim: Fast Replanning for Navigation in Unknown Terrain. In: *IEEE Transactions on Robotics*, vol. 21, 2005, June, no. 3, pp. 354–363

[KO16] Kim, Mingeuk; Oh, Jun-Ho: Study on Optimal Velocity Selection using Velocity Obstacle (OVVO) in Dynamic and Crowded Environment. In: *Autonomous Robots*, vol. 40, 2016, no. 8, pp. 1459–1470

[KR97] Kamon, Ishay; Rivlin, Ehud: Sensory-based Motion Planning with Global Proofs. In: *IEEE Transactions on Robotics and Automation*, vol. 13, 1997, no. 6, pp. 814–822

[KRR98] Kamon, Ishay; Rimon, Elon; Rivlin, Ehud: TangentBug: A Range-Sensor-Based Navigation Algorithm. In: *International Journal of Robotics Research*, vol. 17, 1998, no. 9, pp. 934–953

[KRSV10] Kunz, Tobias; Reiser, Ulrich; Stilman, Mike; Verl, Alexander: Real-time Path Planning for a Robot Arm in Changing Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, October 2010, pp. 5906–5911

[KSLO96] Kavraki, Lydia; Svestka, Petr; Latombe, Jean claude; Overmars, Mark: Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Minneapolis, MN, USA, April 1996, pp. 566–580

[KSRS98] Ko, Nak Y.; Simmons, Reid; Reid, Ko; Simmons, G.: The Lane-Curvature Method for Local Obstacle Avoidance. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Victoria, Canada, October 1998, pp. 1615–1621

[KST+16] Kovacs, Bence; Szayer, Geza; Tajti, Ferenc; Burdelis, Mauricio; Korondi, Peter: A Novel Potential Field Method for Path Planning of Mobile Robots by Adapting Animal Motion Attributes. In: *Robotics and Autonomous Systems*, vol. 82, 2016, pp. 24–34

[KT12] Kiss, Domokos; Tevesz, Gabor: Advanced Dynamic Window Based Navigation Approach using Model Predictive Control. In: *Proceedings of 17th International Conference on Methods and Models in Automation and Robotics (MMAR)*. Miedzyzdrojie, Poland, 2012, pp. 148–153

[Kuc06] Kucsera, Peter: Sensors For Mobile Robot Systems. In: *Academic and Applied Research in Military Science*, vol. 5, 2006, pp. 645–658

[Kum15] Kumar, Vikrant: *Dynamic Path Planning*. http://www.slideshare.net/dare2kreate/dynamic-path-planning (accessed: 28.05.2019), 2015

[Lat91] Latombe, Jean-Claude: *Robot Motion Planning*. Kluwer Academic Publishers, 1991

[LaV98] LaValle, Steven M.: Rapidly-exploring Random Trees: A new Tool for Path Planning. Report No. 98-11 / Computer Science Department, Iowa State University. 1998. – Technical Report

[LaV06] LaValle, Steven M.: *Planning Algorithms*. Cambridge University Press, 2006

[LB99] Laubach, Sharon L.; Burdick, Joel W.: An Autonomous Sensor-Based Path-Planner for Planetary Microrovers. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Detroit, Michigan, May 1999, pp. 347–354

[LDW91] Leonard, John J.; Durrant-Whyte, Hugh F.: Mobile Robot Localization by Tracking Geometric Beacons. In: *IEEE Transactions on Robotics and Automation*, vol. 7, 1991, no. 3, pp. 376–382

[LFG+08] Likhachev, Maxim; Ferguson, Dave; Gordon, Geoff; Stentz, Anthony; Thrun, Sebastian: Anytime Search in Dynamic Graphs. In: *Artificial Intelligence*, vol. 172, 2008, May, no. 14, pp. 1613–1643

[LGT03] Likhachev, Maxim; Gordon, Geoff; Thrun, Sebastian: ARA*: Anytime A* Search with Provable Bounds on Sub-Optimality. In:

Advances in Neural Information Processing Systems (NIPS), MIT Press, 2003, pp. 767–774

[LJO17] Lee, Beom H.; Jeon, Jae D.; Oh, Jung H.: Velocity obstacle based local collision avoidance for a holonomic elliptic robot. In: *Autonomous Robots*, vol. 41, 2017, August, no. 6, pp. 1347–1363

[LK01] LaValle, Steven M.; Kuffner, James J.: Randomized Kinodynamic Planning. In: *The International Journal of Robotics Research*, vol. 20, 2001, no. 5, pp. 378–400

[LLLH08] Liddy, Tommie; Lu, Tien-Fu; Lozo, Peter; Harvey, David: Obstacle Avoidance using Complex Vector Fields. In: *Australasian Conference on Robotics and Automation*, 2008, pp. 1–7

[LLS05] Large, Frederic; Laugier, Christian; Shiller, Zvi: Navigation Among Moving Obstacles using the NLVO: Principles and Applications to Intelligent Vehicles. In: *Autonomous Robots*, vol. 19, 2005, no. 2, pp. 159–171

[LMD+98] Lobo, Jorge; Marques, Lino; Dias, Jorge; Nunes, Urbano; Almeida, Anibal: Sensors for Mobile Robot Navigation. In: *Autonomous Robotic Systems*, vol. 236, 1998, pp. 50–81

[LNWB14] Lawitzky, Andreas; Nicklas, Anselm; Wollherr, Dirk; Buss, Martin: Determining States of Inevitable Collision using Reachability Analysis. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, USA, September 2014, pp. 4142 – 4147

[LS86] Lumelsky, Vladimir J.; Stepanov, Alexander A.: Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment. In: *IEEE Transactions on Automatic Control*, vol. 31, 1986, no. 5, pp. 1058–1069

[LSL98] Laumond, Jean paul; Sekhavat, S.; Lamiraux, F.: Guidelines in Nonholonomic Motion Planning for Mobile Robots. In: *Robot Motion Planning and Control*, vol. 229, 1998, pp. 1–53

[LV14] Lima, Danilo A.; Victorino, Alessandro C.: An Image Based Dynamic Window Approach for Local Navigation of an Autonomous Vehicle in Urban Environments. In: *IEEE ICRA Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots (WMEPC)*. Hong Kong, China, May 2014, pp. 1–6

[LV16] Lima, Danilo A.; Victorino, Alessandro C.: A Hybrid Controller for Vision-based Navigation of Autonomous Vehicles in Urban Environments. In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, 2016, no. 8, pp. 2310–2323

[MA97] Montano, Luis; Asensio, Jose R.: Real-time Robot Navigation in Unstructured Environments using a 3D Laser Rangefinder. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Grenoble, France, September 1997, pp. 526 – 532

[Mas12] Masoud, Ahmad A.: A Harmonic Potential Approach for Simultaneous Planning and Control of a Generic UAV Platform. In: *Journal of Intelligent and Robotic Systems*, vol. 65, 2012, no. 1-4, pp. 153–173

[MBF+10] Marder-Eppstein, Eitan; Berger, Eric; Foote, Tully; Gerkey, Brian P.; Konolige, Kurt: The Office Marathon: Robust Navigation in an Indoor Office Environment. In: *IEEE International Conference on Robotics and Automation ICRA*. Alaska, USA, May 2010, pp. 300–307

[MCL+17] Malone, Nick; Chiang, Hao-Tien; Lesser, Kendra; Oishi, Meeko; Tapia, Lydia: Hybrid Dynamic Moving Obstacle Avoidance Using a Stochastic Reachable Set-Based Potential Field. In: *IEEE Transactions on Robotics*, vol. 33, 2017, no. 5, pp. 1124–1138

[ME85] Moravec, Hans P.; Elfes, Alberto: High Resolution Maps from Wide Angle Sonar. In: *IEEE International Conference on Robotics and Automation (ICRA)*. MO USA, March 1985, pp. 116–121

[MFM13a] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Robust Navigation in Complex Environments. In: *European Navigation Conference (ENC)*. Vienna, Austria, April 2013, pp. 1 – 7

[MFM13b] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Safe Gap Based (SG) Reactive Navigation for Mobile Robots. In: *European Conference on Mobile Robots (ECMR)*. Barcelona, Spain, June 2013, pp. 325 – 330

[MFM15] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Smooth Reactive Collision Avoidance in Difficult Environments. In: *IEEE Conference on Robotics and Biomimetics (ROBIO)*. Zhuhai, China, December 2015, pp. 1471 – 1476

[MFM16] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Tangential Gap Flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments. In: *Journal of Robotics and Autonomous Systems*, vol. 84, 2016, pp. 15–30

[MFM17] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Robust Collision Avoidance for Autonomous Mobile Robots in Unknown Environments. In: *RoboCup 2016: Robot World Cup XX* Bd. 9776, Springer Lecture Notes in Computer Science, November 2017, pp. 327–338

[MFM18] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel: Admissible Gap Navigation: A New Collision Avoidance Approach. In: *Journal of Robotics and Autonomous Systems*, vol. 103, 2018, pp. 93–110

[MFMJ10] Mujahed, Muhannad; Fischer, Dirk; Mertsching, Bärbel; Jaddu, Hussein: Closest Gap Based (CG) Reactive Obstacle Avoidance Navigation for Highly Cluttered Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, October 2010, pp. 1805 – 1812

[MHS13] Matveev, A. S.; Hoy, M. C.; Savkin, A. V.: A Method for Reactive Navigation of Nonholonomic Under-actuated Robots in Maze-like Environments. In: *Automatica*, vol. 49, 2013, May, no. 5, pp. 1268–1274

[Min02] Minguez, Javier: *Robot Shape, Kinematics, and Dynamics in Sensor-Based Motion Planning*, Universidad de Zaragoza, Espana, Diss., July 2002

[Min05] Minguez, Javier: The Obstacle-Restriction Method for Robot Obstacle Avoidance in Difficult Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, Canada, August 2005, pp. 2284–2290

[Min08] Minguez, Javier: Robot Obstacle Avoidance Papers using Experiments / Robotics European Robotics Research Network (EURON). 2008. – Technical Report

[MJFM13] Mujahed, Muhannad; Jaddu, Hussein; Fischer, Dirk; Mertsching, Bärbel: Tangential Closest Gap Based (TCG) Reactive Obstacle Avoidance Navigation for Cluttered Environments. In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Linköping, Sweden - best paper award, October 2013, pp. 1 – 6

[ML09] Muja, Marius; Lowe, David G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: *International Conference on Computer Vision Theory and Applications*, 2009, pp. 331–340

[MLB15] Maravall, Darío; Lope, Javier de; Brea, Juan Pablo F.: Visual Bug Algorithm for Simultaneous Robot Homing and Obstacle Avoidance using Visual Topological Maps in an Unmanned Ground Vehicle. In: *Bioinspired Computation in Artificial Systems - IWINAC, Part II, LNCS 9108*. Switzerland, 2015, pp. 301–310

[MLL08] Minguez, Javier; Lamiraux, Florent; Laumond, Jean-Paul: Motion Planning and Obstacle Avoidance. In: *Springer Handbook of Robotics*. 2008, pp. 827–852

[MLL16] Minguez, Javier; Lamiraux, Florent; Laumond, Jean-Paul: Motion Planning and Obstacle Avoidance. In: *Springer Handbook of Robotics*. 2016, pp. 1177–1202

[MLO+98] Matthies, Larry; Litwin, Todd; Owens, Ken; Rankin, Art; Murphy, Karl; Coombs, David; Gilsinn, Jim; Hong, Tsai; Legowik, Steven; Nashman, Marilyn; Yoshimi, Billibon: Performance Evaluation of UGV Obstacle Detection with CCD/FLIR Stereo Vision and LADAR. In: *Intelligent Control (ISIC), held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS)*. Gaithersburg, MD, September 1998, pp. 658–670

[MM02] Minguez, Javier; Montano, Luis: Robot Navigation in very Complex, Dense, and Cluttered Indoor/Outdoor Evironments. In: *15th IFAC World Congress*. Barcelona, Spain, 2002, pp. 1–6

[MM04] Minguez, Javier; Montano, Luis: Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. In: *IEEE Trans. Rob. Autom*, vol. 20, 2004, no. 1, pp. 45–59

[MM09] Minguez, Javier; Montano, Luis: Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot. In: *IEEE Transactions on Robotics*, vol. 25, 2009, no. 2, pp. 367–381

[MM16] Mujahed, Muhannad; Mertsching, Bärbel: A New Gap-based Collision Avoidance Method for Mobile Robots. In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Lausanne, Switzerland, October 2016, pp. 1 – 7

[MM17] Mujahed, Muhannad; Mertsching, Bärbel: The Admissible Gap (AG) Method for Reactive Collision Avoidance. In: *IEEE Interna-*

tional Conference on Robotics and Automation (ICRA). Singapore, May 2017, pp. 1916 – 1921

[MMS06] Minguez, Javier; Montano, Luis; Santos-Victor, Jose: Abstracting Vehicle Shape and Kinematic Constraints from Obstacle Avoidance Methods. In: *Auton. Robots*, vol. 20, 2006, no. 1, pp. 43–59

[MMSA01] Minguez, Javier; Montano, Luis; Simeon, Thierry; Alami, Rachid: Global Nearness Diagram Navigation (GND). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seoul, Korea, May 2001, pp. 33–39

[MOM04] Minguez, Javier; Osuna, Javier; Montano, Luis: A "Divide and Conquer" Strategy based on Situations to Achieve Reactive Collision Avoidance in Troublesome Scenarios. In: *IEEE International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA, April 2004, pp. 3855–3862

[MSKT13] Maroti, Arpad; Szaloki, David; Kiss, Domokos; Tevesz, Gabor: Investigation of Dynamic Window Based Navigation Algorithms on a Real Robot. In: *Proceedings of IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2013, pp. 95–100

[MSZ09] Mastrogiovanni, Fulvio; Sgorbissa, Antonio; Zaccaria, Renato: Robust Navigation in an Unknown Environment with Minimal Sensing and Representation. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, 2009, no. 1, pp. 212–229

[Muj10] Mujahed, Muhannad: *A Reactive Obstacle Avoidance Method for Autonomous Mobile Robots*, AL-Quds University, Jerusalem, Palestine, Diplomarbeit, June 2010

[MVL07] Munoz, Nelson; Valencia, Jaime; Londono, N.: Evaluation of Navigation of an Autonomous Mobile Robot. In: *International Workshop on Performance Metrics for Intelligent Systems Workshop (PerMIS)*, 2007, pp. 15–21

[NB07] Ng, James; Bräunl, Thomas: Performance Comparison of Bug Navigation Algorithms. In: *Journal of Intelligent and Robotic Systems*, vol. 50, 2007, no. 1, pp. 73–84

[NPL12] Narayanan, Venkatraman; Phillips, Mike; ; Likhachev, Maxim: Anytime Safe Interval Path Planning for Dynamic Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Algarve, Portugal, October 2012, pp. 4708–4715

[NTK+11] Nia, D N.; Tang, H S.; Karasfi, B; Motlagh, O R E.; Kit, A C.: Virtual Force Field Algorithm for a Behaviour-based Autonomous Robot in Unknown Environments. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, 2011, no. 1, pp. 51–62

[Oeg03] Oegren, Petter: *Formations and Obstacle Avoidance in Mobile Robot Control*, Royal Institute of Technology, Stockholm, Sweden, Diss., June 2003

[OL05] Ogren, Petter; Leonard, Naomi E.: A convergent Dynamic Window Approach to Obstacle Avoidance. In: *IEEE Transactions on Robotics*, vol. 21, 2005, no. 2, pp. 188–195

[OM05] Owen, Eduardo; Montano, Luis: Motion Planning in Dynamic Environments using the Velocity Space. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, Canada, August 2005, pp. 2833–2838

[OM06] Owen, Eduardo; Montano, Luis: A Robocentric Motion Planner for Dynamic Environments using the Velocity Space. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Beijing, China, October 2006, pp. 4368–4374

[OS17] Oezdemir, Aykut; Sezer, Volkan: A Hybrid Obstacle Avoidance Method: Follow the Gap with Dynamic Window Approach. In: *IEEE International Conference on Robotic Computing (IRC)*. Taichung, Taiwan, April 2017, pp. 257–262

[Ota09] Ota, Jun: Rearrangement Planning of Multiple Movable Objects by a Mobile Robot. In: *Advanced Robotics*, vol. 23, 2009, no. 1-2, pp. 1–18

[Pan14] Panagou, Dimitra: Motion Planning and Collision Avoidance using Navigation Vector Fields. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, May 2014, pp. 2513–2518

[PCY+16] Paden, Brian; Cáp, Michal; Yong, Sze Z.; Yershov, Dmitry S.; Frazzoli, Emilio: A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles. In: *IEEE Transactions on Intelligent Vehicles*, vol. 1, 2016, no. 1, pp. 33–55

[Pet08] Petti, Stephane R.: *Safe Navigation within Dynamic Environments: a Partial Motion Planning Approach*, Ecole des Mines de Paris, Diss., April 2008

[PF05] Petti, Stephane; Fraichard, Thierry: Safe Motion Planning in Dynamic Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, Canada, August 2005, pp. 2210–2215

[PJK12] Park, Jong J.; Johnson, Collin; Kuipers, Benjamin: Robot Navigation with Model Predictive Equilibrium Point Control. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura, Algarve, Portugal, October 2012, pp. 4945–4952

[PJK+14] Pellenz, Johannes; Jacoff, Adam; Kimura, Tetsuya; Mihankhah, Ehsan; Sheh, Raymond; Suthakorn, Jackrit: RoboCup Rescue Robot League. In: *RoboCup 2014: Robot World Cup XVIII*. Brazil, July 2014, pp. 673–685

[PL11] Phillips, Mike; Likhachev, Maxim: SIPP: Safe Interval Path Planning for Dynamic Environments. In: *IEEE International Conference in Robotics and Automation (ICRA)*. Shanghai, China, May 2011, pp. 5628–5635

[Pla10] Plaku, Erion: *Algorithms for Sensor-based Robotics, Artificial Intelligence*. http://courses.csail.mit.edu/6.034s/ (accessed: 28.05.2019), 2010

[PPK+12] Perez, Alejandro; Platt, Robert; Konidaris, George; Kaelbling, Leslie P.; Lozano-Perez, Tomas: LQR-RRT*: Optimal Sampling-based Motion Planning with Automatically Derived Extension Heuristics. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Minneapolis, MN USA, May 2012, pp. 2537–2542

[PPM13] Park, Chonhyon; Pan, Jia; Manocha, Dinesh: Real-time Optimization-based Planning in Dynamic Environments using GPUs. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany, May 2013, pp. 4090–4097

[PSV11] Pereira, Flavio G.; Santos, Milton Cesar P.; Vassallo, Raquel F.: A Nonlinear Controller for People Guidance Based on Omnidirectional Vision. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. San Francisco, USA, September 2011, pp. 3620–3625

[QCG+09] Quigley, Morgan; Conley, Ken; Gerkey, Brian P.; Faust, Josh; Foote, Tully; Leibs, Jeremy; Wheeler, Rob; Ng, Andrew Y.: ROS: An Open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*. Kobe, Japan, May 2009, pp. 1–6

[RA12] Rezaee, Hamed; Abdollahi, Farzaneh: Adaptive Artificial Potential Field Approach for Obstacle Avoidance of Unmanned Aircrafts. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. Kachsiung, Taiwan, July 2012, pp. 1–6

[RBAM18] Rodrigues, Romulo T.; Basiri, Meysam; Aguiar, A. P.; Miraldo, Pedro: Feature Based Potential Field for Low-Level Active Visual Navigation. In: *ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing, vol 693*, Springer, Cham, 2018, pp. 791–800

[RFS09] Rufli, Martin; Ferguson, Dave; Siegwart, Roland: Smooth Path Planning in Constrained Environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Kobe, Japan, May 2009, pp. 3780–3785

[RGM17] Roelofsen, Steven; Gillet, Denis; Martinoli, Alcherio: Collision avoidance with limited field of view sensing: A velocity obstacle approach. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, June 2017, pp. 1922–1927

[Rib05] Ribeiro, Maria I.: Obstacle avoidance / Institute for Systems and Robotics, ISR-Lisboa. 2005. – Technical Report

[RJ15] Rantanen, Mika T.; Juhola, Martti: Speeding up Probabilistic Roadmap Planners with Locality-sensitive Hashing. In: *Robotica*, vol. 33, 2015, no. 7, pp. 1491–1506

[RK92] Rimon, Elon; Koditschek, Daniel E.: Exact Robot Navigation using Artificial Potential Functions. In: *IEEE Transactions on Robotics and Automation*, vol. 8, 1992, October, no. 5, pp. 501 – 518

[RMP06] Ren, Jing; McIsaac, Kenneth A.; Patel, Rajni V.: Modified Newton's Method Applied to Potential Field-Based Navigation for Mobile Robots. In: *IEEE Transactions on Robotics*, vol. 22, 2006, no. 2, pp. 384–391

[RMP08] Ren, Jing; McIsaac, Kenneth A.; Patel, Rajni V.: Modified Newtons Method Applied to Potential Field Based Navigation for Nonholonomic Robots in Dynamic Environments. In: *Robotica*, vol. 26, 2008, pp. 117–127

[rob19] *RoboCup Federation*. available online: http://www.robocup.org/ (accessed: 28.05.2019), 2019

[Ros97] Rosenblatt, Julio: *DAMN: A Distributed Architecture for Mobile Navigation*, Robotics Institute, Carnegie Mellon University, Diss., January 1997

[ros19] *Robot Operating system (ROS).* available online: http://www.ros.org/ (accessed: 28.05.2019), 2019

[SB02] Stachniss, Cyrill; Burgard, Wolfram: An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Switzerland, October 2002, pp. 508 – 513

[Sch98] Schlegel, Christian: Fast Local Obstacle Avoidance under Kinematic and Dynamic Constraints for a Mobile Robot. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Victoria, Canada, October 1998, pp. 594–599

[Seb14] Sebbane, Yasmina B.: *Planning and Decision Making for Aerial Robots.* Springer International Publishing Switzerland, 2014

[SG12] Sezer, Volkan; Gokasan, Metin: A Novel Obstacle Avoidance Algorithm: "Follow the Gap Method". In: *Robotics and Autonomous Systems*, vol. 60, 2012, no. 9, pp. 1123–1134

[SH13] Savkin, Andrey V.; Hoy, Michael: Reactive and the Shortest Path Navigation of a Wheeled Mobile Robot in Cluttered Environments. In: *Robotica*, vol. 31, 2013, no. 2, pp. 323–330

[Sil05] Silver, David: Cooperative Pathfinding. In: *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE).* California, USA, June 2005, pp. 117–122

[Sim96] Simmons, Reid: The Curvature-Velocity Method for Local Obstacle Avoidance. In: *IEEE International Conference on Robotics and Automation (ICRA).* Minnosota, USA, April 1996, pp. 3375–3382

[Sin97] Singhal, Amit: Issues in Autonomous Mobile Robot Navigation / Computer Science Department University of Rochester. 1997. – Technical Report

[SJP15] Schmerling, Edward; Janson, Lucas; Pavone, Marco: Optimal Sampling-based Motion Planning under Differential Constraints:

The Driftless Case. In: *IEEE International Conference on Robotics and Automation (ICRA)*. WA, USA, May 2015, pp. 2368–2375

[SL91] Slotine, Jean-Jacques; Li, Weiping: *Applied Nonlinear Control*. Upper Saddle River, NJ : Pearson, 1991

[SLS01] Shiller, Zvi; Large, Frederic; Sekhavat, Sepanta: Motion Planning in Dynamic Environments: Obstacles Moving along Arbitrary Trajectories. In: *IEEE International Conference on Robotics and Automation (ICRA)* Bd. 4. Seoul, Korea, May 2001, pp. 3716–3721

[SMP05] Seder, Marija; Macek, Kristijan; ; Petrovic, Ivan: An Integrated Approach to Real-time Mobile Robot Control in Partially Known Indoor Environments. In: *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society (IECON)*, 2005, pp. 1785–1790

[SN04] Siegwart, Roland; Nourbakhsh, Illah R.: *Introduction to Autonomous Mobile Robots*. Cambridge (Mass.) : MIT Press, 2004

[SO97] Svestka, Petr; Overmars, Mark H.: Motion Planning for Car-like Robots using a Probabilistic Learning Approach. In: *The International Journal of Robotics Research*, vol. 16, 1997, no. 2, pp. 119–143

[SP07] Seder, Marija; Petrovic, Ivan: Dynamic Window Based Approach to Mobile Robot Motion Control in the Presence of Moving Obstacles. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Roma, Italy, April 2007, pp. 1986–1991

[SRD+17] Schwarz, Max; Rodehutskors, Tobias; Droeschel, David; Beul, Marius; Schreiber, Michael; Araslanov, Nikita; Ivanov, Ivan; Lenz, Christian; Razlaw, Jan; Schüller, Sebastian; Schwarz, David; Topalidou-Kyniazopoulou, Angeliki; Behnke, Sven: NimbRo Rescue: Solving Disaster-response Tasks with the Mobile Manipulation Robot Momaro. In: *Journal of Field Robotics*, vol. 34, 2017, no. 2, pp. 400–425

[SS07] Shim, David H.; Sastry, Shankar: An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations. In: *American Control Conference*. New York, NY, July 2007, pp. 3886–3891

[SS12] Shiller, Zvi; Sharma, Sanjeev: High Speed on-line Motion Planning in Cluttered Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal, October 2012, pp. 596–601

[SSV09] Siciliano, Bruno; Sciavicco, Lorenzo; Villani, Luigi: *Robotics : Modelling, Planning and Control*. London : Springer, 2009 (Advanced Textbooks in Control and Signal Processing)

[Ste95] Stentz, Anthony: The Focussed D* Algorithm for Real-Time Replanning. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* Bd. 2. San Francisco, CA, USA, August 1995, pp. 1652–1659

[SWY10] Shi, Chaoxia; Wang, Yanqing; Yang, Jingyu: A Local Obstacle Avoidance Method for Mobile Robots in Partially Known Environment. In: *Robotics and Autonomous Systems*, vol. 58, 2010, may, no. 5, pp. 425–434

[UB98] Ulrich, Iwan; Borenstein, Johann: VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Leuven, Belgium, May 1998, pp. 1572–1577

[UB00] Ulrich, Iwan; Borenstein, Johann: VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In: *IEEE International Conference on Robotics and Automation (ICRA)*. San Francisco, CA, USA, April 2000, pp. 2505–2511

[VT12] Valbuena, Luis; Tanner, Herbert G.: Hybrid Potential Field Based Control of Differential Drive Mobile Robots. In: *Journal of Intelligent and Robotic Systems*, vol. 68, 2012, no. 3-4, pp. 307–322

[VTML00] Vadakkepat, Prahlad; Tan, Kay; Ming-Liang, Wang: Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning. In: *Proceedings of the 2000 Congress on Evolutionary Computation.* La Jolla, CA, USA, July 2000, pp. 1–8

[Wan14] Wang, Chao: *Collision Free Autonomous Navigation and Formation Building for Non-holonomic Ground Robots*, Cornell University, Diss., February 2014

[WC00] Wang, Yunfeng; Chirikjian, Gregory S.: A new potential field method for robot path planning. In: *IEEE International Conference on Robotics and Automation (ICRA).* San Francisco, CA, USA, September 2000, pp. 977–982

[WH12] Wu, Albert; How, Jonathan P.: Guaranteed Infinite Horizon Avoidance of Unpredictable, Dynamically Constrained Obstacles. In: *Autonomous Robots*, vol. 32, 2012, no. 3, pp. 227–242

[WIN15] Weerakoon, Tharindu; Ishii, Kazuo; Nassiraei, Amir Ali F.: An Artificial Potential Field Based Mobile Robot Navigation Method To Prevent From Deadlock. In: *Journal of Artificial Intelligence and Soft Computing Researc (JAISCR)*, vol. 5, 2015, no. 3, pp. 189–203

[YAT10] Yuste, Hector; Armesto, Leopoldo; Tornero, Josep: Benchmark Tools for Evaluating AGVs at Industrial Environments. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* Taipei, Taiwan, October 2010, pp. 2657–2662

[Ye07] Ye, Cang: Navigating a Mobile Robot by a Traversability Field Histogram. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, 2007, no. 2, pp. 361–372

[YP09] Yufka, Alpaslan; Parlaktuna, Osman: Performance Comparison of Bug Algorithms for Mobile Robots. In: *International Advanced Technologies Symposium.* Karabuk, Turkey, May 2009, pp. 1–5

List of Notations

Notation	Explanation
$\mathcal{A}(q)$	Region of the workspace $\mathcal{W}$ occupied by the robot at configuration q
$\angle(\phi_a \rightarrow \phi_b)$	The angular distance between ϕ_a and ϕ_b traveling from ϕ_a to ϕ_b
$\angle(\phi_a, \phi_b)$	The minimum angular distance from ϕ_a to ϕ_b
$\mathcal{B}$	Goal bridge $\mathcal{B}$ constructed between the robot and the goal
$\mathcal{C}$	Closest gap
$\mathrm{C_{chg}}$	Curvature Change
$\mathcal{C}_{\text{free}}$	Free configuration space
$\mathcal{C}_{\text{obstacles}}$	Configuration space obstacles
$\mathcal{C}_{\text{space}}$	Configuration space
$\mathcal{CC}_i$	Set of colliding relative velocities between the robot and obstacle $\mathcal{O}_i$, *collision cone*
χ_i	Tangent direction associated with $\mathcal{T}_i$
$\mathcal{CO}_i$	Configuration space obstacle
$\mathrm{d}(\mathbf{p}_a, \mathbf{p}_b)$	Euclidean distance between points $\mathbf{p}_a$ and $\mathbf{p}_b$
d_b	Buffer distance
D_ϵ	An open and connected subset of $\mathbb{R}^n$ centered at the origin
d_l	Lookahead distance
D_s	Safe distance around the robot
$\mathrm{d_s(g)}$	Suitable distance to the sides creating gap g
$\mathrm{d_{safe}}$	Desired clearance to obstacles
D_{vs}	A parameter used to limit the speed
δ	Any value greater than 0
$\Delta(\mathbf{p}_\mathrm{i})$	Gap flow angle corresponding to $\mathbf{p}_\mathrm{i}$

Notation	Explanation
$\Delta(\mathbf{p}_i^{\mathrm{H}})$	Gap flow angle corresponding to $\mathbf{p}_i^{\mathrm{H}}$
$\mathrm{F}_{\mathrm{att}}(\mathbf{p}_r)$	Attractive force resulting from $\mathrm{U}_{\mathrm{att}}(\mathbf{p}_r)$
$\mathrm{F}_{\mathrm{rep}}^{i}(\mathbf{p}_r)$	Repulsive force resulting from $\mathrm{U}_{\mathrm{rep}}^{i}(\mathbf{p}_r)$
G	List of assembled gaps
g	Represents a gap: a potentially open area among obstacles through which the vehicle may pass
$\hat{\mathrm{g}}$	Gravitational acceleration
γ	Angle used to compute Ψ_{vg} and equals $\gamma = \theta_t - \theta_{\mathrm{c}}$
$\Gamma(\mathbf{p}_i)$	Function used to head the robot to $\mathbf{p}_t$. It is set to 1 or -1 based on location of $\mathbf{p}_i$ relative to $\mathbf{p}_t$
H	Subset (left or right) containing threat $\mathbf{p}_i^{\mathrm{H}}$
H^*	Subset (left or right) that does not contain $\mathbf{p}_i^{\mathrm{H}}$
$\mathrm{J}_{\mathrm{acc}}$	Linear Jerk Cost
$k(t)$	Curvature
λ	Evaluates to $\frac{\pi}{2}$ if H^* is empty or $\Delta(\mathbf{p}_i^{\mathrm{H}})$ otherwise
${}^{\mathcal{M}}\theta_i^{\mathrm{S}}$	Angle towards $\mathbf{p}_i^{\mathrm{S}}$ relative to frame $\mathcal{M}$
${}^{\mathcal{M}}\theta_{\mathrm{l}}(\mathrm{g})$	Angle towards $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ relative to frame $\mathcal{M}$
${}^{\mathcal{M}}\theta_{\mathrm{r}}(\mathrm{g})$	Angle towards $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ relative to frame $\mathcal{M}$
μ	Coefficient of friction
$\hat{\mathrm{N}}$	Union of subsets $\hat{N}_{\mathrm{R}}$ and $\hat{N}_{\mathrm{L}}$
N	Set of obstacles in collision while traveling to $\mathbf{p}_t$
$\mathrm{N}_{\mathrm{col}}$	Number of Collisions
$\hat{N}_{\mathrm{L}}$	N_{L} excluding threats satisfying $\left\lvert T(y_i^{\mathrm{S}})\right\rvert > \left\lvert T(y_{\mathrm{c}}^{\mathrm{N_L}})\right\rvert$
N_{L}	Left-subset: threats located to the right of $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$
$\hat{N}_{\mathrm{R}}$	N_{R} excluding threats satisfying $\left\lvert T(y_i^{\mathrm{S}})\right\rvert > \left\lvert T(y_{\mathrm{c}}^{\mathrm{N_R}})\right\rvert$
N_{R}	Right-subset: threats located to the right of $\overrightarrow{\mathbf{p}_r\mathbf{p}_t}$
$\mathcal{O}$	Set of obstacles in the workspace
$\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathcal{C})]$	Obstacles in collision with the boundary of the robot while traveling directly towards $\mathbf{p}_{\mathrm{s}}(\mathcal{C})$
$\mathcal{O}_{\mathrm{collision}} : [\mathbf{p}_r \rightarrow \mathbf{p}_x]$	Obstacles in collision with the boundary of the robot while traveling directly towards $\mathbf{p}_{\mathrm{x}}$

Notation	Explanation
O^{+}	List of obstacles falling to the left of $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, such that the angular distance traveled does not exceed π
ω	Angular or rotational velocity
$\omega_{\max}$	Maximum angular speed
$\mathbf{p}_{\mathrm{c}}$	Obstacle stoint closest to the robot
$\mathbf{p}_{\mathrm{c}}^{\mathrm{N_L}}$	Obstacle point closest to the robot and contained in N_{L}
$\mathbf{p}_{\mathrm{c}}^{\mathrm{N_R}}$	Obstacle point closest to the robot and contained in N_{R}
$\mathbf{p}_{\mathrm{c}}^{\hat{\mathrm{N}}_{\mathrm{L}}}$	Obstacle point closest to the robot and falling in $\hat{N}_{\mathrm{L}}$
$\mathbf{p}_{\mathrm{cg}}(\mathrm{g})$	Scan point creating the gap g side closer to the goal
$\mathbf{p}_{\mathrm{cr}}(\mathrm{g})$	Scan point creating the gap g side closer to the robot
$\mathbf{p}_{\mathrm{fg}}(\mathrm{g})$	Scan point creating the gap g side farther from the goal
$\mathbf{p}_{\mathrm{fr}}(\mathrm{g})$	Scan point creating the gap g side farther from the robot
$\mathbf{p}_g$	Location of the goal
$\hat{\mathbf{p}}_g$	Instantaneous location of the goal
$\mathbf{p}_i$	An obstacle point
$\mathbf{p}_{i-}^{S}$	Right neighborhood of point $\mathbf{p}_i^{\mathrm{S}}$
$\mathbf{p}_i^{\mathrm{H}}$	Any threat contained in H
$\mathbf{p}_i^{\mathrm{H}^*}$	The threat closest to $\mathbf{p}_i^{\mathrm{H}}$ and contained in H^*
$\mathbf{p}_i^{*}$	Obstacle point closest to $\mathbf{p}_i$ and located in $\mathcal{R}^*$
$\mathbf{p}_i^{\mathrm{S}}$	A laser scan point
$\mathbf{p}_i^{\mathrm{S}-}$	Sequence of points located to the right of $\mathbf{p}_i^{\mathrm{S}}$
$\mathbf{p}_i^{\mathrm{S}+}$	Sequence of points located to the left of $\mathbf{p}_i^{\mathrm{S}}$
$\mathbf{p}_{i+}^{\mathrm{S}}$	Left neighborhood of point $\mathbf{p}_i^{\mathrm{S}}$
$\mathbf{p}_{\mathrm{l}}(\mathrm{g})$	Scan point creating the left side of gap g
$\mathrm{P}_{\mathrm{len}}$	Path Length
$\mathbf{p}_{\mathrm{nav}}$	Side of the selected gap that the robot circumnavigates
$\mathbf{p}_r$	Location of the robot
$\mathbf{p}_{\mathrm{r}}(\mathrm{g})$	Scan point creating the right side of gap g
${}^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_i}(\mathrm{g})$	Visibility angle of $\mathbf{p}_i$ with respect to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$
$\mathbf{p}_{\mathrm{s}}(\mathrm{g})$	An instantaneous subgoal within gap g
$\mathbf{p}_t$	Target which can be the goal or the subgoal based on checking the path to the goal cerriterion.

Notation	Explanation
$\mathbf{p}_{\mathrm{vg}}$	Target location after rotating it by Ψ_{vg}
$\mathrm{proj}(\theta)$	A function normalizes θ to the range $[-\pi, \pi[$
Ψ	Total rotation angle $(\Psi_{\mathrm{sg}} + \Psi_{\mathrm{vg}})$
ψ_i	Rotation angle corresponding to threat $\mathbf{p}_i^{\mathrm{S}}$.
Ψ_{L}	Weighted average rotation angle caused by threats in $\hat{N}_{\mathrm{L}}$
$\Psi_{\max}$	The larger absolute value among both Ψ_{L} and Ψ_{R}
Ψ_{R}	Weighted average rotation angle caused by threats in $\hat{N}_{\mathrm{R}}$
Ψ_{sg}	Gap rotation angle
Ψ_{vg}	Collision avoidance rotation angle
q	A robot's configuration
$\mathbb{R}$	Real number set
R	Robot radius
$r_{\mathrm{c}}^{\hat{\mathrm{N}}}$	Distance to the closest threat that is contained in $\hat{\mathrm{N}}$
$r_{\mathrm{fg}}(\mathcal{C})$	Distance to the side of $\mathcal{C}$ farther from $\mathbf{p}_g$.
$r_{\mathrm{cg}}(\mathcal{C})$	Distance to the side of $\mathcal{C}$ closer to $\mathbf{p}_g$.
(r_i, θ_i)	Polar coordinates of $\mathbf{p}_i$
r_i^{H}	Distance to $\mathbf{p}_i^{\mathrm{H}}$
r_i^*	Distance to $\mathbf{p}_i^*$
r_i^{S}	Distance to scan point $\mathbf{p}_i^{\mathrm{S}}$
$r_{\max}$	Maximum range of the laser scanner
$r_{\min}$	Distance to the obstacle point closest to the robot boundary
$\mathrm{R}_{\mathrm{obs}}$	Risk with Respect to Obstacles
$r_{\mathrm{r}}(\mathrm{g})$	Distance to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$
$r_{\mathrm{s}}(\mathrm{g})$	Distance to $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$
r_t	Distance to $\mathbf{p}_t$
$r_x(\mathrm{g})$	Distance to point $\mathbf{p}_x(\mathrm{g})$
$\mathcal{R}^-$	Region of the workspace located to the right of $\mathbf{p}_t$
$\mathcal{R}^*$	Region of the workspace that does not include $\mathbf{p}_{\mathrm{c}}$
$\mathcal{R}^+$	Region of the workspace located to the left of $\mathbf{p}_t$
S	List of laser scan points
$\mathcal{S}$	Safe gap
$\mathrm{S}_{\mathrm{lat}}$	Lateral Stress

Notation	Explanation
$\mathrm{S}_{\mathrm{tng}}$	Tangential Stress
$\mathcal{S}^1$	Unit circle
$\mathrm{sat}_{[a,b]}(x)$	A function limits x between a and b
t	Time
$T(x_i^{\mathrm{S}})$	X coordinate of $\mathbf{p}_i^{\mathrm{S}}$ relative to the robot-target frame
$T(y_i^{\mathrm{S}})$	Y coordinate of $\mathbf{p}_i^{\mathrm{S}}$ relative to the robot-target frame
t_a	Latency time
t_c	Computation time
$\mathcal{T}_i$	Circular path that the robot follows to reach point $\mathbf{p}_i$
T_{L}	Total number of threats contained in $\hat{N}_{\mathrm{L}}$
T_{R}	Total number of threats contained in $\hat{N}_{\mathrm{R}}$
$\mathrm{T}_{\mathrm{tot}}$	Total Execution Time
θ	Orientation of the robot relative to a global coordinate system
θ_{c}	Angle towards $\mathbf{p}_{\mathrm{c}}$
θ_{center}	Angle towards the center point between $\mathbf{p}_i$ and $\mathbf{p}_i^*$
$\theta_{\mathrm{cg}}(\mathcal{C})$	Angle towards the side of $\mathcal{C}$ closer to $\mathbf{p}_g$.
$\theta_{\mathrm{fg}}(\mathcal{C})$	Angle towards the side of $\mathcal{C}$ farther from $\mathbf{p}_g$.
θ_g	Angle towards the goal
$\theta_i^{\mathrm{H}^*}$	Angle towards $\mathbf{p}_i^{\mathrm{H}^*}$
θ_i^{S}	Angle towards scan point $\mathbf{p}_i^{\mathrm{S}}$
θ_i^*	Angle towards $\mathbf{p}_i^*$
θ_{mid}	Angle towards the center of the gap.
$\theta_{\mathrm{r}}(\mathrm{g})$	Angle towards $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$
$\theta_{\mathrm{s}}(\mathrm{g})$	Angle towards $\mathbf{p}_{\mathrm{s}}(\mathrm{g})$
$\theta_{\mathrm{s}}(\mathcal{S})$	Angle towards the subgoal associated with gap $\mathcal{S}$
θ_{scs}	Angle towards $\mathcal{C}$ keeping a safe distance to $\mathbf{p}_{\mathrm{cg}}(\mathcal{C})$
θ_t	Angle towards $\mathbf{p}_t$
θ_{traj}	Trajectory angle
$\theta_x(\mathrm{g})$	Angle towards point $\mathbf{p}_x(\mathrm{g})$
$\mathrm{U}_{\mathrm{att}}(\mathbf{p}_r)$	Attractive potential to the goal

Notation	Explanation
$\mathrm{U}^{i}_{rep}(\mathbf{p}_r)$	Repulsive potential corresponding to obstacle $\mathbf{p}_i$
Υ	Function evaluates to 1 or -1 based on the location of the gap side that the robot circumnavigates
v	Linear velocity
V_a	Set of admissible velocities in the DWA approach
V_d	Dynamic window in the DWA approach
v_i	Initial velocity
v_{limit}	Speed limit based on the distance to nearby obstacles
v_{max}	Maximum linear speed
V_r	Area of considered velocities within the dynamic window
V_s	Set of all possible velocities in the DWA approach
$\mathcal{VO}_i$	Velocity obstacle associated with obstacle $\mathcal{O}_i$
$\mathcal{W}$	Workspace
$w(\mathrm{g})$	Width of gap g, i.e. $w(\mathrm{g}) = \|\mathbf{p}_{\mathrm{l}}(\mathrm{g}) - \mathbf{p}_{\mathrm{r}}(\mathrm{g})\|$
$\mathrm{W}(\Psi_{\mathrm{L}})$	Weight associated with Ψ_{L}
$\mathrm{W}(\Psi_{\mathrm{R}})$	Weight associated with Ψ_{R}
w_i	Weight corresponding to ψ_i
W_{L}	Total weight associated with threats falling in $\hat{N}_{\mathrm{L}}$
$w^{\mathrm{L}}_{\mathrm{max}}$	Maximum weight assigned to threats located on $\hat{N}_{\mathrm{L}}$
$\hat{x}_g$	x coordinate of $\hat{\mathbf{p}}_g$
(x_i, y_i)	Cartesian coordinates of $\mathbf{p}_i$
(x^*_i, y^*_i)	Cartesian coordinates of $\mathbf{p}^*_i$
$(x^{\mathrm{S}}_i, y^{\mathrm{S}}_i)$	Cartesian Coordinates of scan point $\mathbf{p}^{\mathrm{S}}_i$
(x, y)	Position of the robot relative to a global coordinate system
$y_{\mathrm{c}}^{\mathrm{N_L}}$	Y-coordinate of $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_L}}$
$y_{\mathrm{c}}^{\mathrm{N_R}}$	Y-coordinate of $\mathbf{p}_{\mathrm{c}}^{\mathrm{N_R}}$
Z_{w}	Zero Crossings
ζ_{acc}	Rotational Jerk Cost
$\hat{\zeta}_g$	Angle to line L_v in the velocity space from figure 6.8

List of Abbreviations

Abbreviation	Explanation
AG	Admissible Gap
APF	Artificial Potential Field
ARM	Arc Reachable Manifold
CG	Closest Gap Navigation
CVM	Curvature Velocity Method
DWA	Dynamic Window Approach
FFOV	Full Field of View
GND	Global Nearness-Diagram
ICS	Inevitable Collision States
LCM	Lan-Curvature Method
LFOV	Limited Field of View
LTG	Local Tangent Graph
ND	Nearness-Diagram Navigation
ND+	Nearness-Diagram Navigation Plus
NLVO	Non Linear Velocity Obstacles
PRM	Probabalistic Roadmap
RRT	Rapidly-Exploring Random Tree
RVO	Reciprocal Velocity Obstacles
SG	Safe Gap Navigation
SIPP	Safe Interval Path Planning
SND	Smooth Nearness-Diagram
TGF	Tangential Gap Flow Navigation
VFF	Virtual Force Field
VFH	Vector Field Histogram
VO	Velocity Obstacles

List of Tables

5.1 Performance assessment of the proposed TGF approach for experiments 1 - 6, presented in section 5.2, using the metrics defined in sections 5.3.1 - 5.3.5 (results of experiments 1 - 4 are reprinted from [MFM16] with permission from Elsevier). 106
5.2 Performance evaluation results for the experiments presented in section 4.3 from chapter 4, using the metrics defined in sections 5.3.1 - 5.3.5. 110
5.3 Performance assessment of the TGF-controller for scenarios 3 and 4 from section 5.2 (reprinted from [MFM16] with permission from Elsevier). As a reference, the results of the ND-controller from table 5.1 are listed, too. 111
5.4 Performance assessment of the TGF-controller for scenarios 2 and 3 from section 4.3. As a reference, the results of the ND-controller from table 5.2 are listed, too. 112

6.1 Performance assessment results of the proposed "AG approach" for experiments 1 - 6 from section 6.5 (reprinted from [MFM18], with permission from Elsevier). 151

List of Figures

1.1 Our mobile robot, GETbot, moving through a cluttered environment. 3

2.1 Main stages of the navigation process (originally from [SN04]). . 10

2.2 A triangular mobile robot $\mathcal{A}$ (left image) that is allowed to translate freely in a two-dimensional space at a fixed orientation. The reference point of $\mathcal{A}$ is marked as a small circle. The configuration space obstacles $\mathcal{C}_{\text{obstacles}}$ (right image) is obtained by enlarging the workspace $\mathcal{W}$ (hatched area) by the shape of $\mathcal{A}$ (middle image) (originally from [Lat91]). 13

2.3 Path planning for a triangular robot that is allowed to translate and rotate in a two-dimensional space. The configuration space in such a case is $\mathbb{R}^2 \times \mathcal{S}^1$, where $\mathcal{S}^1$ is the unit circle. Planning a path for the triangular-shaped robot in the workspace (left image) is equivalent to planning a path for a point-like robot in the configuration space (right image) (originally from [Pla10]). 14

2.4 A differential-drive mobile robot, which can only move perpendicular to the wheels axis. 16

2.5 Bug 1 algorithm with (H_1, H_2) described as hit points, and (L_1, L_2) described as leave points (from [SN04]). 28

2.6 Bug 2 algorithm with (H_1, H_2) described as hit points, and (L_1, L_2) described as leave points (from [SN04]). 29

2.7 Artificial Potential Field. (a) Typical potential fields; an attractive force is acted by the goal and a repulsive force is acted by obstacles (from [Kum15]). (b) A robot experiences a local minimum while approaching a U-shaped obstacle. This problem occures if the attractive force gets symmetric to the repulsive force (originally from [Rib05]). 31

2.8 Mapping of active cells onto the polar histogram (originally from [BK91]). 33

2.9 Velocity space (from [FBT97]). 34

2.10 Dynamic window (from [FBT97]). 35

2.11 The set of forbidden robot velocities ($v_1 \cup v_2$). In order to guarantee a safe motion, a control velocity must be chosen outside of this set. (originally from [MLL08]). 37

2.12 Overview of the Nearness-Diagram (ND) navigation approach (originally from [MLL08]). (a) Diagram showing the design of the ND method [MM04] following the *situated-activity* paradigm. Based on the sensor readings and the locations of the robot and the goal, one situation is chosen and the associated motion law is computed. (b) An example shows how to determine the motion direction. First, the situation is identified: the security zone is obstacle-free, the motion region is wide, and the target does not fall within the motion region. In this case, the situation is HSWR. Second, the suitable action is executed computing the most promising motion direction θ_{sol}. 40

3.1 Collision check along the "direct path" towards two locations, $\mathbf{p}_x$ and $\mathbf{p}_y$. Line $\overline{\mathrm{OL}}_1$ that passes through $\mathbf{p}_i$ intersects E_1 in $\mathbf{p}_1$. Obstacle $\mathbf{p}_i$ causes collision with the direct path towards $\mathbf{p}_x$ since it is located on the dark red line segment $\overline{\mathbf{p}_1\mathbf{p}_1'}$. Line $\overline{\mathrm{OL}}_2$ that goes through $\mathbf{p}_j$ hits E_4 in $\mathbf{p}_2$. Obstacle $\mathbf{p}_j$ is collision-free while traveling towards $\mathbf{p}_y$ since it is not located on $\overline{\mathbf{p}_2\mathbf{p}_2'}$. None of the robot edges intersects $\overline{\mathrm{OL}}_3$, so $\mathbf{p}_k$ is collision-free while traveling towards $\mathbf{p}_x$. The path towards $\mathbf{p}_y$ is free, while the path towards $\mathbf{p}_x$ is in collision with the red obstacle points (adapted from [MFM13b] with permission from IEEE). 46

3.2 Finding out gaps. Firstly, the gaps marked as 1 - 4 and visualized by green arrows are found out by the "forward search" and the gaps marked as 5 - 8 and visualized by red arrows are extracted by the "backward search". Secondly, gaps 1, 4, and 6 are discarded as they are located within gaps 8, 5, and 3, respectively. Gap 5 is also discarded since its width is less than $2R$. Then, the closest gap (gap 8) is selected for navigation (adapted from [MFMJ10] with permission from IEEE). 49

3.3 Classifying gaps based on their location and angular width. Gaps 1-3 are front gaps, whereas gap 4 is a rear gap. Gaps 1 and 4 are in a good vision state, while gaps 2 and 3 are in a weak vision state (adapted from [MFM13b] with permission from IEEE). . . 51

3.4 Determining the safe gap. The closest gap and the safe gap are denoted by $\mathcal{C}$ and $\mathcal{S}$. The robot is driven towards $\mathcal{S}$ rather than $\mathcal{C}$, ensuring a safer behavior and providing a gradual change in the steering angle (adapted from [MFM13b] with permission from IEEE). 53

3.5 Determining the subgoal corresponding to gap g. For clarity, the notation representing the gap, (g), is dropped, e.g. $r_{\mathrm{s}}(\mathrm{g})$ becomes r_{s}. 55

3.6 Scenario 1 simulations. (a) Path generated by the CG method, where rapid changes in the direction of motion occurs. (b) Smoother path generated by SG. (c) Speed profile of the CG method. (d-g) Snapshots of the SG simulation. The goal and obstacles are visualized by magenta circle and black lines, respectively. The closest gap $\mathcal{C}$ and the safe gap $\mathcal{S}$ are shown by green and blue line segments, where subgoals are represented by small circles on the center of gaps. The obstacle points in collision with the direct path towards $\mathcal{C}$ are visualized by red color. (h) Speed profile of the SG method (reprinted from [MFM13b] with permission from IEEE). 58

3.7 Scenario 2 simulations. (a-d) Paths generated using the implementation of ND+, SND, CG, and SG, respectively. (e-h) Linear and angular velocities visualized against the time elapsed for ND+, SND, CG, and SG, respectively. 60

3.8 Experiments (reprinted from [MFM13b] with permission from IEEE). (a) Environment setup (b) Path generated by applying the CG approach, where the robot almost touched the obstacle marked as A. (c) Path generated by SG achieving smoother and safer behavior. (d) Speed profile for CG. (e) Speed profile for SG. 62

4.1 Visualization of three motion situations based on the "Path-to-goal" and "Safety" criteria. (a) "Free-path" and "High-safety": this situation does not require any action. (b) "Dangerous-path" and "High-safety": in this case, a rotation to the goal position is performed, driving the robot towards a subgoal within the closest gap. (c) "Low-safety": a temporary rotation to the goal (resp. subgoal) position is performed so that the robot avoids collisions with obstacles (adapted from [Muj10] and from [MJFM13] with permission from IEEE). 65

4.2 Computing the gap rotation angle Ψ_{sg}. The closest gap $\mathcal{C}$ is assumed narrow. Therefore, Ψ_{sg} is determined in such a way that the robot passes through the gap center $\mathcal{C}$. 67

4.3 Computing Ψ_{vg} based on the tangential navigation concept for different cases. (a, b) $\text{sgn}(\theta_t) \neq \text{sgn}(\theta_c)$ where $|\gamma| < \pi$ for (a) and $|\gamma| \geq \pi$ for (b). (c, d): $\text{sgn}(\theta_t) = \text{sgn}(\theta_c)$ and $|\theta_c| \geq |\theta_t|$ where $\theta_c < 0$ for (c) and $\theta_c > 0$ for (d). (e, f): $\text{sgn}(\theta_t) = \text{sgn}(\theta_c)$ and $|\theta_c| < |\theta_t|$ where $\theta_c > 0$ for (e) and $\theta_c < 0$ for (f). (adapted from [Muj10] and from [MFM16] [MJFM13] with permissions from Elsevier and IEEE). 69

4.4 Oscillations that may occur in a tight passage by employing the "tangential navigation" concept. (a) The closest obstacle point is located on side A. This imposes a rotation to $\mathbf{p}_t$ by Ψ_{vg} and the robot navigates tangential to A accordingly. (b) The robot gets closer to side B. At that point, the new Ψ_{vg} causes the robot to move tangential to B. (c) The robot gets closer to side A again, moving tangential to it. (d) Fulfilling the *leaving condition*, and in turn guiding the robot directly towards $\mathbf{p}_t$ (adapted from [MFM16] with permission from Elsevier). 71

4.5 Considering the clearance to both sides of a target $\mathbf{p}_t$ while computing the avoidance trajectory. The line towards $\mathbf{p}_t$ divides the workspace into two regions, $\mathcal{R}^+$ and $\mathcal{R}^-$. It is obvious that $\mathbf{p}_i$ is located in $\mathcal{R}^+$. Hence, while computing Ψ_{vg} associated with $\mathbf{p}_i$ the clearance between $\mathbf{p}_i$ and the obstacles located in $\mathcal{R}^-$ (visualized by green and orange) is considered. Since our objective is to drive the robot towards $\mathbf{p}_t$, all obstacles making an angular distance $> \pi$ with $\mathbf{p}_i$ are excluded (such as those visualized by orange). Among the remaining obstacles, the closest to $\mathbf{p}_i$ is selected (denoted $\mathbf{p}_i^*$ and visualized by dark green). 72

4.6 Computing $\Delta(\mathbf{p}_i)$. $\mathbf{p}_i$ is closer to $\mathbf{p}_r$ than $\mathbf{p}_i^*$. Hence, $\Delta(\mathbf{p}_i)$ is set such that $\frac{1}{2}\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is maintained to $\mathbf{p}_i$ as visualized in (a). If $\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is high, the maintained distance is limited to d_{safe}, see (b). 74

4.7 Computing $\Delta(\mathbf{p}_i)$. $\mathbf{p}_i^*$ is closer to $\mathbf{p}_r$ than $\mathbf{p}_i$: $\Delta(\mathbf{p}_i)$ is set such that the robot moves towards the center point between $\mathbf{p}_i$ and $\mathbf{p}_i^*$, see (a). If $\|\mathbf{p}_i - \mathbf{p}_i^*\|$ is high, the maintained distance is set to d_{safe}, see (b). 75

4.8 The target $\mathbf{p}_t$ is translated to a safer location between both sides of the closest gap $\mathcal{C}$ if the holonomic path towards $\mathbf{p}_t$ is unsafe. This step is necessary to determine the set of obstacles that may cause collision with the robot while guiding it towards $\mathbf{p}_t$. For clarity, the notation representing the closest gap ($\mathcal{C}$) is removed, e.g. $\theta_{\text{cg}}(\mathcal{C})$ is abbreviated as θ_{cg} (adapted from [MFM16] with permission from Elsevier). 76
4.9 A robot navigates towards a given goal location (adapted from [MJFM13] with permission from IEEE). 81
4.10 Test 1. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE). 86
4.11 Test 2. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE). 87
4.12 Test 3. (a) Experimental setup. (b-e) Paths generated by (b) ND+, (c) CG, (d) SG and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) CG, (h) SG and (i) TGF (from [MFM15] with permission from IEEE). 88

5.1 Scenario 1 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF. 92
5.2 Scenario 2 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF. 93
5.3 Scenario 3 (reprinted from [MFM16], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ND+, (c) SND, (d) CG, and (e) TGF. (f-i) Speed profiles for (f) ND+, (g) SND, (h) CG, and (i) TGF. 94

5.4 Scenario 4 (reprinted from [MFM16], with permission from Elsevier). (a, d) Environmental setup. (b, c, e, and f) Paths generated by (b) ND+, (c) SND, (e) CG, and (f) TGF. (g-j) Speed profiles for (g) ND+, (h) SND, (i) CG, and (j) TGF. 95

5.5 Scenario 5 (reprinted from [MFM16], with permission from Elsevier). (a, b) Environmental setup where (a) Depicts the start of the mission at which two boxes were located in front of the robot and (b) Mimics the moment at which three boxes were pushed towards the corridor once line L was crossed. (c-e) Paths generated by (c) ND+, (d) SND, and (e) CG, where oscillations in motion can be observed. (f) TGF avoided collision on time and smoothly proceeded towards the goal. (g-j) Speed profiles for (g) ND+, (h) SND, (i) CG, and (j) TGF. 97

5.6 Scenario 6. (a-c) Environmental setup. (d-g) Paths generated by (d) ND+, (e) SND, (f) CG, and (g) TGF. (h-k) Speed profiles for (h) ND+, (i) SND, (j) CG, and (k) TGF. 98

5.7 Scenario 7. (a-c) Environmental setup. (d-g) Snapshots of the experiment taken whenever the robot passed through door D1, at which several students closed the robot's path for a while. (h-i) Snapshots of the experiment show how the robot navigated through door D3. 100

5.8 Visualization corresponding to Experiment 7. (a) Trajectory generated by the robot applying the TGF method. (c,d) Recorded motion commands plotted against the time elapsed. 101

5.9 Scenario 3 and 4 from section 5.2, but with $(0.7\,m/s, 1.3\,rad/s)$ speed limits (reprinted from [MFM16], with permission from Elsevier). (a, b) Paths generated in experiment 3 using (a) ND+ and (b) TGF. (c, d) Paths generated in experiment 4 using (c) ND+ and (d) TGF. (e-h) Speed profiles corresponding to the paths shown in (a-d), respectively. 109

5.10 Scenarios 3 and 4 from section 5.2 running TGF, but using the TGF-controller (reprinted from [MFM16] with permission from Elsevier). (a, c) Paths generated in (a) Scenario 3 and (c) Scenario 4, using $(0.5\,m/s, 1\,rad/s)$ speed limits . (b, d) Paths generated in (b) Scenario 3 and (d) Scenario 4, using $(0.4\,m/s, 0.8\,rad/s)$ speed limits. (e-h) Speed profiles corresponding to the paths shown in (a-d). 111

5.11 Scenarios 2 and 3 from section 4.3 running TGF, but using the TGF-controller. (a, c) Paths generated in (a) Scenario 2 and (c) Scenario 3 using $(0.5\,m/s, 1\,rad/s)$ speed limits. (b, d) Paths generated in (b) Scenario 2 and (d) Scenario 3 using $(0.4\,m/s, 0.8\,rad/s)$ speed limits. (e-h) Speed profiles corresponding to the paths shown in (a-d). 112

6.1 Finding out a gap (g) by the "counterclockwise search". The light gray color depicts the area covered by the sensing system. The obstacles are shown by blue regions, where the list of detected depth points S are visualized by small colored circles. See the text for explaining how gap g is detected. For a better visualization, the symbol denoting the gap (g) and the superscript $\mathbf{p_r}$ in the "visibility angle" are eliminated (reprinted from [MFM18], with permission from Elsevier). 119

6.2 Finding out gaps by the AG method. The green and red arrows visualize the gaps that are found by the "counterclockwise" and "clockwise" searches, respectively. In the "gaps reduction" step, each gap depicted by a dashed arrow is eliminated (reprinted from [MFM18] , with permission from Elsevier). 122

6.3 Extracting gaps by different methodologies, including the proposed AG approach. The total number of gaps returned by AG and CG [MFMJ10] are 5 (marked as 1 - 5) and 4 (labeled i - iv), depicted by green and red arrows, respectively. For the ND methods (e.g [MM04] and [DB08]), 14 gaps are detected, marked as A - N (reprinted from [MFM18], with permission from Elsevier). . . 123

6.4 Visualization of the paths followed by a kinematically constrained robot. Circles $\mathcal{T}_1$ and $\mathcal{T}_2$ describe the paths followed by the robot to reach points $\mathbf{p}_1$ and $\mathbf{p}_2$. Whenever $\mathbf{p}_1$ (resp. $\mathbf{p}_2$) is reached, the robot's orientation is θ_1 (resp. θ_2) and the distance traversed is s_1 (resp. s_2). The point closest to $\mathbf{p}_3$ and falling on $\mathcal{T}_2$ is denoted by $\mathbf{p}_3(\mathcal{T}_2)$ (reprinted from [MM17], with permission from IEEE). 125

6.5 Assigning a subgoal $\mathbf{p}_s$ to a gap g in such a way that the robot circumnavigates one of its sides $\mathbf{p}_{\text{nav}}$ while obeying the kinematic constraints. First, we identify circle S whose center is $\mathbf{p}_{\text{nav}}$ and whose radius is $\mathrm{d_s}$. This circle is mutually tangent to two circular paths ($\mathcal{T}_{t1}$ and $\mathcal{T}_{t2}$), each of which lies on the y-axis. The tangent points are labeled $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$. We locate the subgoal at $\mathbf{p}_{t1}$ as it leads to g (reprinted from [MM17], with permission from IEEE). 128

6.6 Collision checking along $\tau\,[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}]$; the path followed to reach $\mathbf{p}_{\mathrm{s}}$. This example consists of three obstacles $\mathbf{p}_1$ - $\mathbf{p}_3$. The circles that pass through them and centered at c_s are $\mathcal{C}(c_s, \mathbf{p}_1)$ and $\mathcal{C}(c_s, \mathbf{p}_3)$. Circle $\mathcal{C}(c_s, \mathbf{p}_1)$ hits $\mathcal{P}_e$ in $\mathbf{p}_e$, where $\mathbf{p}_1$ lies on the arc between $\mathbf{p}_e$ and $\mathbf{p}_e^*$, but $\mathbf{p}_2$ doesn't. Therefore, $\mathbf{p}_1$ is in collision, but $\mathbf{p}_2$ is not. None of the robot edges intersects $\mathcal{C}(c_s, \mathbf{p}_3)$, thus $\mathbf{p}_3$ is collision-free (reprinted from [MM17], with permission from IEEE). 131

6.7 Checking navigability. The robot detects two gaps; g_1 and g_2. It is obvious that the closest gap (g_1) is non-admissible, since $\tau\,[\mathbf{p}_r \rightarrow \mathbf{p}_{\mathrm{s}}(\mathrm{g}_1)]$ is in collision with the obstacle points shown by orange and dark green dots. But, g_1 is navigable because it is possible to create an *admissible* gap (g_1^*) that leads to g_1. See the text for more information on constructing g_1^* (reprinted from [MM17], with permission from IEEE). 137

6.8 Determining the motion control in such a way that the radius of curvature $\hat{r}_g$ is maintained and the maximum possible velocity is respected. Any point on L_v can be a valid motion control as it satisfies $v = w\hat{r}_g$. Here, we specify S_{limit} so that the velocity of the robot is controlled based on the clearance to obstacles (originally from [MMS06]). 139

6.9 Scenario 1 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG. . . . 141
6.10 Scenario 2 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG. . . . 142
6.11 Scenario 3 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG. . . . 143
6.12 Scenario 4 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b-e) Paths generated by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f-i) Speed profiles for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG. . . . 144
6.13 Scenario 5 (reprinted from [MFM18], with permission from Elsevier). (a) Environmental setup. (b) A pedestrian stepped in front of the robot. (c) The pedestrian stepped out of the track. (d) Box B was taken out of the arena, creating gap G2. (e-h) Paths generated by (e) ARM-ND+, (f) DWA-A*, (g) TGF, and (h) AG. (i-l) Speed profiles for (i) ARM-ND+, (j) DWA-A*, (k) TGF, and (l) AG. . . . 146
6.14 Scenario 6 (reprinted from [MFM18], with permission from Elsevier). (a, b) Environmental setup. (c-f) Paths generated by (c) ARM-ND+, (d) DWA-A*, (e) TGF, and (f) AG. (g-j) Speed profiles for (g) ARM-ND+, (h) DWA-A*, (i) TGF, and (j) AG. . 147
6.15 Experiment 7. (a, b) Environmental setup, the robot had to move through the connection between buildings P1 and P7. (c-l) Snapshots of the experiment taken at different locations, showing that the robot was able to react on time avoiding the students who stepped in to close its way to the goal. . . . 149
6.16 Trajectory followed by the robot (a) and the recorded motion commands against the time (b, c) corresponding to experiment 7. . . 150

List of Publications

The work presented in this thesis has been published in the following peer-reviewed international conference proceedings and journals:

[1] M. Mujahed, D. Fischer, and B. Mertsching, "Admissible Gap Navigation: A New Collision Avoidance Approach," *Journal of Robotics and Autonomous Systems*, vol. 103, pp. 93 - 110, May 2018.

[2] M. Mujahed and B. Mertsching, "The Admissible Gap (AG) Method for Reactive Collision Avoidance," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Marina Bay Sands, Singapore), pp. 1916 - 1921, May 2017.

[3] M. Mujahed, D. Fischer, and B. Mertsching, "Robust Collision Avoidance for Autonomous Mobile Robots in Unknown Environments," in *RoboCup 2016: Robot World Cup XX, Springer Lecture Notes in Computer Science*, vol. 9776, November 2017.

[4] M. Mujahed and B. Mertsching, "A New Gap-based Collision Avoidance Method for Mobile Robots," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (Lausanne, Switzerland), pp. 220 - 226, October 2016.

[5] M. Mujahed, D. Fischer, and B. Mertsching, "Tangential Gap Flow (TGF) Navigation: A New Reactive Obstacle Avoidance Approach for Highly Cluttered Environments," *Journal of Robotics and Autonomous Systems*, vol. 84, pp. 15 - 30, July 2016.

[6] M. Mujahed, D. Fischer, and B. Mertsching, "Smooth Reactive Collision Avoidance in Difficult Environments," in *IEEE international Conference*

on Robotics and Biomimetics (ROBIO), (Zhuhai, China), **Best paper finalist**, pp. 1471 - 1476, December 2015.

[7] M. Mujahed, H. Jaddu, D. Fischer, and B. Mertsching, "Tangential Closest Gap Based (TCG) Reactive Obstacle Avoidance Navigation for Cluttered Environments," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (Linköping, Sweden), **Best paper award prize**, pp. 1 - 6, October 2013.

[8] M. Mujahed, D. Fischer, and B. Mertsching, "Safe Gap based (SG) Reactive Navigation for Mobile Robots," in *European Conference on Mobile Robots (ECMR)*, (Barcelona, Spain), pp. 325 - 330, September 2013.

[9] M. Mujahed, D. Fischer, and B. Mertsching, "Robust Navigation in Complex Environments," in *European Navigation Conference 2013 (ENC 2013)*, (Vienna, Austria), pp. 1 - 7, April 2013.

Lebenslauf

Name	Mujahed
Vorname	Muhannad Abdallah Shaker
Wohnort	Wuppertal
Geburtsdatum/-ort	25.09.1980, Hebron, Palästina
Nationalität	Jordanisch
Familienstand	verheiratet

Schulbildung

09/1985-06/1994	Grundschule
09/1994-07/1997	Gymnasium

Studium

09/1997-07/2002	„Bachelor Degree in Computer Systems Engineering“ Palestine Polytechnic University (PPU), Palästina
09/2007-07/2010	„Master Degree in Electronic and Computer Engineering“ Al-Quds University, Palästina

Wissenschaftliche Tätigkeit

09/2002-09/2011	Hebron Fachhochschule
09/2010-02/2011	Al-Quds Open University
10/2011-03/2017	Universität Paderborn

Beruf

Seit 04/2017	Forscher - Entwicklung autonom fahrender Autos, APTIV Services Deutschland GmbH

Sprachkenntnisse Arabisch
Englisch
Deutsch

Ort, Datum Paderborn, 28.05.2019

Unterschrift

Drei Stichwörter für den Deutschen Fakultätentag zur Dissertation:

- Mobiler Roboter
- Kollisionsvermeidung
- Sensorbasierte Bewegungsplanung