Deanship of Graduate Studies

AL-Quds University

A Reactive Obstacle Avoidance Method for Autonomous Mobile Robots

Muhannad A. Mujahed

M.Sc. Thesis

Jerusalem - Palestine

June, 2010

A Reactive Obstacle Avoidance Method for Autonomous Mobile Robots

By

Muhannad A. Mujahed

B.Sc.: Computer Engineering, Palestine Polytechnic University, Palestine

A thesis submitted to
Faculty of Engineering, Al-Quds University
in partial fulfillment of the requirements
for the degree of
Master of Electronic and Computer Engineering

Supervisor: Dr. Hussein Jaddu

Al-Quds University
June, 2010

Electronic and Computer Engineering Master Program
Faculty of Engineering
Al-Quds University

Thesis Approval

A Reactive Obstacle Avoidance Method for Autonomous Mobile Robots

By

Student Name: Muhannad A. Mujahed
Reg. No: 20714152

Supervisor: Dr. Hussein Jaddu

Master thesis submitted and accepted.
Date: _____

The names and signatures of the examining committee members are as follows:

1- Dr. Hussein Jaddu	: Head of Committee	Signature: _____
2- Dr. Labib Arafeh	: Internal Examiner	Signature: _____
3- Prof. Dr. Karim Tahboub	: External Examiner	Signature: _____

Dedication

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

Wonderful grandmother, for her love and support;

Sister and brothers, for their encouragement;

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

Thank you all, I love you!

Declaration

I certify that this thesis submitted for the degree of Master is the result of my own research, except where otherwise acknowledged, and that this thesis (or any part of the same) has not been submitted for higher degree to any other university or institution.

Signed:.....

Muhannad A. Mujahed

Date:.....

Acknowledgements

I would like to express my greatest gratitude to the following people that helped me to turn a dream into reality, be it through scientific as well as moral support. Without their guidance and help the work presented in this thesis would not have been possible.

The first person that comes to mind is my principal adviser Dr. Hussein Jaddu, who believed in me from the very beginning and allowed me to develop professionally as well as individually. With his patience, guidance, deep vision, support, constant encouragement, this thesis has been done what it is today. I am very glad and it has been an honor to work with him over the past year.

I am very fortunate to have had the opportunity to work and interact with the GET lab research group at the University of Paderborn, Germany. The first idea of this thesis started there while I had been assigned to a team preparing the participation of GET Lab in the SICK Robot Day 2009. Also, the practical experiments of this thesis were carried out there through online collaboration. My deepest thanks and Regards to all of you for support and help. In particular, I owe gratitude to the director of GET lab Prof. Bärbel Mertsching who gave me the opportunity to work with her research group. She was so kind and provided me with a lot of help and support while I was there. I will not forget these beautiful days. I also present my sincere thanks to my advisor there Dirk Fischer, who has been suffering me during the experiments of this thesis. Thank you a lot for your guidance, which was extremely helpful. Although I have had little experience in the field of robotics at the beginning, he helped me to go ahead and overcome all difficulties. I am deeply indebted to Irtiza Ali for his valuable interest and support. Even though he was so busy in preparing his PHD defense, he did not hesitate to provide help and give me the best research advice with a high human quality.

Thanks go out to all my teachers at AL-Quds University who do their best to provide me with their help, encouragement and experience. Specifically, I owe thanks to Dr. Salaheddin Odeh for all the help and support he had given me during my graduate period of study.

I would like to thank all of my friends and colleagues, both here at AL-Quds Uneversity and at Paderborn University for their love, help and encouragement. Specifically, I would like to thank Nidal Shakarneh, who was always there. Thank you for your valuable help. I would also like to thank Mohammad Odeh for his suggestions and positive discussion.

Above all I wish to thank my parents, wife, daughter, sister and brothers for their love and support. I am completely indebted to my mom; you pray for me every day to be the best person in your eyes and to succeed in 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 for that. Thanks a lot to my wife and daughter Aseel, who have suffered my unlimited working hours at days and nights. Aseel missed me a lot during the past three years and she was always asking me: why don’t you stay with us, dad?

ملخص الرسالة

حظي الرجل الآلي المتحرك ذاتيا (الذكي) اهتماما كبيرا في السنوات الأخيرة الماضية، بالخصوص مع تطور وانتشار التطبيقات المتعددة التي يكون وجود الإنسان فيها خطر أو صعب مثل البحث والإنقاذ، التنظيف، أو الاستكشاف. معظم هذه التطبيقات تتطلب العمل في بيئة مجهولة، غير محددة، أو بالغة التعقيد.ابتكار تقنية ملاحة (navigation) فعالة تستطيع تسيير الرجل الآلي بطريقة آمنة في مثل هذه الأماكن ما زال عملية صعبة ومشكلة تحتاج البحث.

في هذه الرسالة تم تطوير وتنفيذ (implement) طريقة جديدة لتجاوز العوائق بشكل متفاعل (reactive)، تسمى ملاحة الفجوة الأقرب (Closest Gap Navigation)، للرجل الآلي الذي يتحرك في بيئة معقدة ومزدحمة بالعوائق. الإبداع في هذه التقنية يتمثل في إيجاد وسيلة جديدة لتحليل الفجوات أمام الرجل الآلي من شأنها تقليل عدد هذه الفجوات بالمقارنة مع الطريقة المتبعة في التقنية المعروفة في هذا المجال: (Nearness-Diagram Navigation)، خصوصا في السيناريوهات المعقدة. بالإضافة إلى ذلك، تم أخذ العرض أو النطاق الزاوي (angular width) للفجوة المختارة بعين الاعتبار. أدى ذلك إلى تخفيف التأرجح في حركة الرجل الآلي (oscillations) و تقليل التعقيدات الحسابية، أيضا تم الوصول إلى أداء أكثر سلاسة (smoother). هذه التقنية الجديدة تضبط قانون التحكم المقترح في تقنية ال (Smooth Nearness Diagram Navigation) من أجل إيجاد مسارات أكثر أمنا للرجل الآلي بالأخذ بعين الاعتبار نسبة التهديدات (العوائق) على جانبيه ودفعه أو حرفه بصوره أشد كلما اقترب من العائق أكثر. لذلك، مشكلة التوقف التام (deadlock) التي تحصل في الممرات الضيقة، عندما يكون هناك عوائق كثيرة في جانب من جوانب الرجل الآلي وعوائق قليلة في الجانب الأخر، قد تم حلها دون التأثير على سلاسة التحرك.

يضاف إلى ذلك، في هذه الرسالة أيضا تم دمج طريقة تحليل الفجوات، المقترحة في تقنية ملاحة الفجوة الأقرب (CG method) مع تقنية الهروب المماسي (Tangential Escape method). هذا الدمج، والذي سمي تقنية ملاحة الفجوة الأقرب المماسية (Tangential Closest Gap Navigation)، أدى إلى مسارات أسرع وأقل تأرجحا (less oscillatory) للرجل الآلي. أيضا، تم اشتقاق أوامر التحكم بحيث يكون النظام المتحكم مستقرا وقد تم إثبات ذلك باستخدام نظرية ليابونوف (Layapunov)، والتي تضمن أن الرجل الآلي يصل أي هدف ممكن وصوله.

هذه الرسالة تقدم أيضا تحسينات أخرى على تقنية ملاحة الفجوة الأقرب المماسية (TCG method). التقنية المطورة، والمعروفة بتقنية ملاحة الفجوة الأقرب المماسية السلسة (Smooth Tangential Closest Gap Navigation)، تأخذ بعين الاعتبار كل العوائق الواقعة ضمن مسافة آمنة حول محيط الرجل الآلي (تكون معرفة مسبقا)، وليس فقط العائق الأقرب، في حساب أوامر التحكم. وبذلك، هذه التقنية قادرة على توليد مسارات أكثر سلاسة للرجل الآلي، خصوصا عندما يكون شكل العوائق غير مضلع. بالإضافة إلى ذلك، هذه التقنية تقوم بتحسين درجة أمان المسارات المتولدة باستخدام ال (TCG method) من خلال حفظ مسافة آمنة بين الرجل الآلي والعوائق بينما يسير بجانبها في الممرات الواسعة. بالنسبة للممرات الضيقة، الرجل الآلي يتحرك فيما بين العوائق الموزعة على جانبيه الأيمن والأيسر.

قمنا بتقديم نتائج المحاكاة (simulation) والتجارب العملية من أجل عرض سلوك التقنيات المقترحة وإظهار قدرتها بالمقارنة مع تقنيات وطرق أخرى.

Abstract

Autonomous mobile robots have been given a lot of interest in the last few years, particularly with the evolution of application fields where human presence is dangerous or difficult such as search-and-rescue, cleaning or exploration. Most of these applications require operation in unknown, uncertain and densely cluttered environments. Developing a satisfactory navigation method that can drive a mobile robot safely in these environments is still a challenging problem.

In this thesis, a new reactive obstacle avoidance approach, entitled Closest Gap Navigation (CG), for mobile robots moving in cluttered and complex environments was developed and implemented. The novelty of this approach lies in the creation of a new method for analyzing openings in front of the robot that highly reduces their number as compared with the well known Nearness-Diagram Navigation (ND) technique, particularly in complex scenarios. Moreover, the angular width of the chosen (selected) gap with respect to the robot vision is taken into consideration. Consequently, oscillations are alleviated, the computational complexity is reduced and a smoother behavior is achieved. Our technique adjusts the motion law proposed in the Smooth Nearness-Diagram Navigation (SND) method to generate safer paths for the robot by considering the ratio of threats on its sides and applying stricter deviation against an obstacle as it gets closer to the robot. Hence, the problem of deadlock occurring in narrow corridors, with high threats on one side and low threats on the other, is solved without affecting the smoothness behavior.

In addition, in this thesis the new method for analyzing gaps, proposed in the Closest Gap (CG) approach, is integrated with the Tangential Escape (TE) scheme. This combination, named Tangential Closest Gap Navigation (TCG), results in faster and less oscillatory robot paths. Moreover, motion commands are derived with proven stability in the Layapunov sense for the whole control system, which ensures that the robot reaches any reachable goal.

Also, this thesis introduces further enhancements for the Tangential Closest Gap Navigation (TCG) approach. The enhanced method, entitled Smooth Tangential Closest Gap Navigation (STCG), considers all obstacle points falling within a pre-defined safe distance of the boundary of the robot, not just the closest one, in calculating the motion commands.

Hence, this technique is capable of generating smoother robot paths, particularly for non-polygonal obstacle shapes. Furthermore, it improves the safety of paths generated by the TCG through keeping a safe distance between the robot and obstacles while following their contour in wide corridors. For narrow corridors, the robot moves between the obstacles on both sides of the robot heading direction.

Simulation and experimental results are presented to show the performance of the proposed approaches and to demonstrate their power as compared with other methods.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

As a result of the exponential growth in science and technology in the past few years, particularly with the advent of computers, robots have achieved their greatest success in the world of industrial manufacturing and started to appear in the daily life. Robotics researchers do their best efforts to find out new creative ideas every day. Robots have come a long way from being just machines capable of performing predefined tasks; they are now more smart, reliable and versatile than ever before [1].

Nowadays, a shift from remotely operated to autonomous mobile robots has been considered. Autonomous mobile robots are able to maintain a sense of position and to navigate without human intervention [2]. This is necessary for applications that can be found in fields where human presence is dangerous, difficult or the tasks to be carried out are impossible to be accessed by people [3]. Various examples of such applications can be found in the real life: transportation [4], search and rescue [5, 6], industrial applications [7], cleaning [8], helping surgeons in operations such as performing laparoscopic surgery [9], military [10, 11], exploration; exploring inside a volcano [12] or exploring another planet in the space [13].

Usually, the real world environment where the mobile robot moves through in order to carry out the tasks specified for an application is unstructured, hazardous or cluttered. In addition, unknown or unpredictable obstacles (that can have an arbitrary size or shape) may block the trajectory of the robot during operation. In this case, the navigation challenge is to find a method to generate a collision-free path for the robot, from its current position to a

desired goal, while avoiding static or dynamic obstacles which can be distributed randomly in its way such as: humans, tables, stones and even other robots operating in the same area.

Motion planning (global) [14] techniques deal with the navigation problem on a larger scale in which a pre-defined map is used to calculate an optimal path for the robot to reach its goal. This path is computed off-line with previously known obstacles and static environment. However, for most applications in mobile robotics the environment is partially or completely unknown and changes with time. In this regard, motion planning algorithms cease to function properly and the robot is fated to collide with obstacles [15]. To overcome this limitation the motion techniques must depend on sensors detecting instantaneous changes in the environment or obstacles appearing periodically, with a specified radius of vision. Under such circumstances robots are capable of perceiving the environments, reacting to unpredictable changes and re-planning dynamically in order to safely reach their goal. This can be achieved by local reactive navigation (obstacle avoidance) methods. In these approaches, only a small fraction of the world model is required. As a consequence, fast obstacle avoidance can be achieved with low computational complexity.

Many existing obstacle avoidance methods have problems in dealing with dense and cluttered environments (as the one shown in Fig. 1.1), which is usually the case in most robotic applications [15]. Some drawbacks of these approaches are: the local minima problem, deadlock, oscillatory behavior and the computational complexity. It is still an open research problem to find an efficient reactive navigation technique for autonomous mobile robots that will enable them to safely move in such environments.

In the light of what have been mentioned above, the major goal of this thesis is to contribute to the efforts to develop a new reactive collision avoidance method for robots that move in arduous and complex environments. This method, as will be shown hereinafter, aimed to avoid the limitations of the existing approaches mentioned above.

Figure 1.1: Typical office environment where people, chairs, tables and doors can be distributed randomly. Office environments are considered highly cluttered.

1.2 Thesis Contribution

In this thesis, a new reactive collision avoidance approach for mobile robots moving in cluttered and complex environments was developed and implemented. The proposed method has several advantages compared with previous works that are designed to operate under such environments (e.g. the Nearness-Diagram Navigation (ND) method [15]). The major advantages of this method are: it reduces the computational complexity required to perform decisions, achieves faster and smoother behavior, alleviates oscillations and avoids deadlocks. Simulation and experimental results demonstrate the power of the proposed approach.

The contributions of this thesis can be stated as follows:

- Develops a new method for fetching and analyzing openings surrounding the robot. This method highly reduces the number of gaps detected and as a result reduces the computational complexity and alleviates oscillations.
- Considers the angular width of the chosen gap[1] with respect to the robot vision in calculating the best heading direction. Hence, safer and smoother robot trajectories are achieved.

[1] The chosen gap is the gap which makes the progress towards the specified goal.

- Improves the safety of paths generated by the Smooth Nearness-Diagram Navigation (SND) method proposed in [16] by considering the ratio of threats (obstacles) on the two sides of the robot and applying stricter deviation against an obstacle as it gets closer to the robot.
- Generates faster and less oscillatory robot paths by integrating the new method proposed for analyzing gaps (mentioned as the first contribution) with the Tangential Escape (TE) approach proposed in [17].
- Adapts the integrated method (mentioned in the previous point) by considering all obstacle points falling inside a pre-defined safe distance around the robot, not just the closest one, in developing the motion law. Furthermore, the motion commands are chosen in the integrated method where the whole control system is asymptotically stable in the Layabunov sense.
- The adapted approach (mentioned in the previous point) achieves safer robot trajectories through forcing the robot to keep a safe distance to obstacles while following their contour. If the corridor is narrow, the robot moves in the mid of the distance between the obstacles on its sides.

1.3 Thesis Organization

The remaining chapters of this thesis are organized as follows:

Chapter 2 presents an overview of the autonomous mobile robot navigation. In this chapter, several concepts and design issues are introduced. In addition, a survey of the most popular navigation methods is discussed showing the advantages and disadvantages of each approach. The Navigation techniques are divided in this chapter, as in the literature, into global (motion planning approaches) and local (reactive navigation methods). The reactive navigation methods (our thesis belongs to this group of methods) are also classified to directional and velocity space approaches. The place of this thesis as compared with other reactive approaches is shown in this chapter.

Chapter 3 addresses a new reactive collision avoidance approach for mobile robots moving in cluttered and complex environments, entitled Closest Gap (CG) method. In this chapter, a new method for analyzing openings in front of the robot is presented. Moreover, the angular width of the chosen (selected) gap with respect to the robot vision is taken into consideration. This chapter also describes how the motion law proposed in the Smooth Nearness-Diagram Navigation (SND) method is adapted to avoid the problem of trapping the robot, which occurs in narrow corridors with huge number of obstacles on one of its sides as compared with the other. Finally, this chapter shows simulation and experimental results that demonstrate the power of the proposed approach.

Chapter 4 introduces an improvement to the reactive collision avoidance approach addressed in chapter 3, which we call Tangential Closest Gap (TCG) method. In this chapter, two concepts are integrated: The Closest Gap (CG) for fetching and analyzing openings surrounding the robot and the Tangential Escape (TE) for reactive obstacle avoidance navigation. In addition, the stability of the control system is proved in the Layapunov sense. Simulation and experimental results are also shown in this chapter.

Chapter 5 proposes further enhancements of the real-time obstacle avoidance technique presented in chapter 4. We refer to the enhanced approach as Smooth Tangential Closest Gap (STCG) method. In this chapter, all obstacle points falling within a pre-defined safe distance of the boundary of the robot are considered, not just the closest one as in the TCG approach proposed in chapter 4. Moreover, this chapter improves the safety of paths generated by the TCG through keeping a safe distance between the robot and obstacles while following their contour. Simulation results are also presented to show the performance of the proposed approach.

Chapter 6 draws the conclusions of this thesis and presents recommendations for future works.

Chapter 2

Autonomous Mobile Robot Navigation

2.1 Introduction

The basic aim of a mobile robot is to navigate to a specified target as efficiently (choose an optimal path) and as safely (avoid collisions) as possible. Navigation refers to choosing the best path to guide the robot from its current position to a desired goal. Several constraints complicate the navigation problem and cause inaccuracy in decision. These constraints include: The computational power (prohibits us from achieving real time performance), the presence of obstacles (may be dynamic ones), the uncertainty caused by using multiple sensing modalities by the robot for perception, the errors caused by the robot mechanical system (e.g. not good breaking system), and the weakness of sensing devices themselves [18]. One can refer to [19] for more information regarding these constraints.

Consider the uncertainties caused by the above mentioned constraints; it is hard to autonomously move a mobile robot from one location to another. However, the complexity of navigation can be reduced by dividing it into smaller portions (modules) to deal with them independently of each other and then combine the solutions. J. Leanard and H. Durrant-white [20] summarized the problem of navigation into answering the following three questions: “where am I?”, “where am I going?” and “how should I get there?”. These questions define the three problems, respectively: *robot localization* [21], the mobile robot should identify its current position in an environment from the sensor data; *goal recognition*, the robot must specify the goal position based on sensor observations and goal specifications; and *motion planning*, the robot should plan a path to reach its goal. Solving these problems depends mainly on the data coming from sensors, which is usually known as *robot perception*. This defines the fourth major portion of the navigation process as stated in [22] (Fig. 2.1).

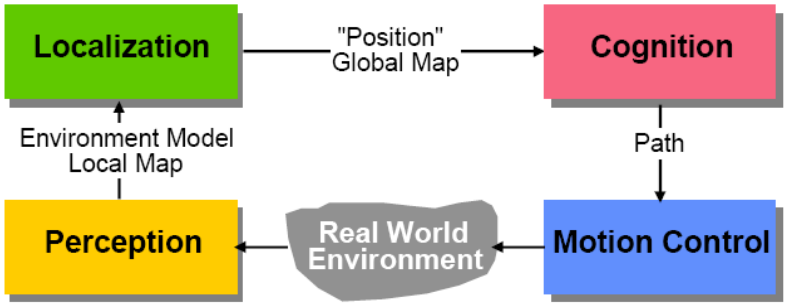


Figure 2.1: Four main stages of the navigation process [82].

The third problem (motion control) 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 is directed to [22]. The next section discusses some concepts and design issues that are necessary to understand the next chapters, whereas section 2.3 surveys the most popular autonomous mobile robot navigation methods.

2.2 Concepts and Design Issues

This section presents the most important concepts necessary for designing a navigation technique. It discusses the main characteristic of a mobile robot including the constraints that add extra difficulties for designing a general algorithm, the important sensors attached to the robot and used to perceive an environment and the software tool used to virtually create scenarios in order to apply the implemented algorithm and get the results.

2.2.1 Mobile Robot Characteristics

Robotics is a multi-faceted field covering several aspects or characteristics [23]. In this subsection, we present three important aspects since they are correlated to the thesis subject.

2.2.1.1 Mobility

Robots are manufactured with moving parts to be able to carry out their missions instead of humans. Mobility means that the robot can move in its entirety on the ground (i.e. have wheels, legs or flying capability). In this regard, the robot can perform tasks easily and efficiently with the added advantage of flexibility in executing new or complex missions.

Furthermore, mobile robots can easily be adapted to operate in different environments and for various purposes (no need to design a special robot for each environment). These robots can cooperate with humans and work together in a shared workspace where it is needed [24].

Mobility is a key property for robots that perform fetch and carry tasks such as exploration, office or home assistant, rescuing and transportation. However, most of the industrial robots are stationary and have only moving arms to do specific tasks.

2.2.1.2 Autonomy

Autonomous mobile robots are robots that can perform tasks without human guidance or control. The degree of autonomy is different from one robot to another depending on the application. To achieve a high autonomy, the robot should be able to adapt well to unpredictable changes in the environment [23]. This is obtained by using sensing devices such as sonar sensors and cameras. However, a lot of disturbances and uncertainties (as those mentioned in section 2.1) may affect the autonomy of a robot and this causes a limitation in achieving robustness.

2.2.1.3 Holonomic and Non-holonomic

The terms holonomic and non-holonomic are often used in mobile robotics. It is helpful to discuss their use in developing a motion planning algorithm [24]. A holonomic system is one in which the number of degrees of freedom are equal to the number of coordinates needed to specify the configuration of the system.. In the field of mobile robotics, any mobile robot with three degrees of freedom of motion in the plane has become known as a holonomic mobile robot [24].

In holonomic robots, path integrals depend only on the initial and final positions [25] (independent of the trajectory followed between the two positions). Therefore, the robot can move to any position following any direction [26] (it can move forward, backward or sideway). This makes motion planning easier to implement since the robot can accelerate in any direction without any constraints. An example of a holonomic system is a person walking on the ground; he can instantly go towards the left or right, as well as moving forwards or backwards.

The non-holonomic robots can go anywhere but not following any trajectory [25]. An automobile is an example of a non-holonomic system where the movement can be only forwards or backwards but not to the side (as shown in Fig. 2.2 [26]). The non-holonomic robots are more popular because of their simplicity and ease of control. However, they make the implementation of motion planning algorithms more difficult since they cannot perform movements in all directions.

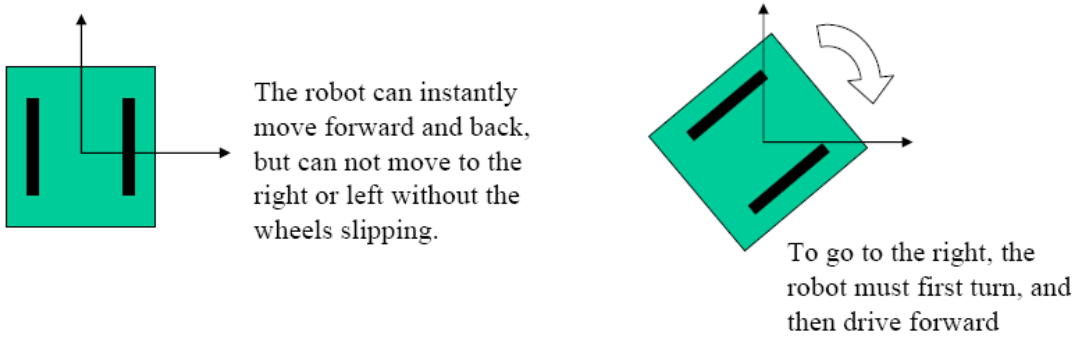


Figure 2.2: A non-holonomic mobile robot moves only forward and backward but not to the side [26].

2.2.2 Robot Shape, Kinematic and Dynamic Constraints

There are three aspects that add constraints on the motion of a mobile robot; namely, shape, kinematics and dynamics. Most of the researchers do not take these constraints into consideration while developing their obstacle avoidance algorithms and this leads to inaccuracy in the robot navigation.

The shape and the kinematics are considered as a geometric issue where the robot configurations in collision are represented given the admissible trajectories [25]. Many existing reactive navigation algorithms assume a differential drive and a circular shape robot whereas most of the real robots do not have these configurations. The dynamic constraints generally imply controlling the speed of the robot taking into account the current velocity, the maximum acceleration and the distance to obstacles as follows [25]:

- Restrict velocities to those that can be reached within a short time interval given the limited accelerations of the robot and the holonomicity constraints.
- Choose velocities that ensure safe trajectories where the robot can stop before reaching the closest obstacle surrounding it.

2.2.3 An Overview of Mobile Robot Sensors

For a mobile robot to perform tasks properly and avoid the danger of colliding obstacles while moving, it is necessary to perceive the world surrounding it. This can be achieved by sensors detecting position, velocity and instantaneous changes in the environment. There are various types of sensors that the mobile robot uses in the perception process. However, it is difficult to effectively combine the outputs of these sensors and it is still an open research area [27]. The following explains the most important sensors used in mobile robots, among others. Other sensor types and more details regarding this subject can be found in [28].

2.2.3.1 Shaft Encoders

Shaft encoders provide information about the distance traveled by a mobile robot, relative to a reference point, through measuring the number of revolutions of its motor. The robot uses the output of the shaft encoders to determine the current position of the robot based on a process called *dead reckoning* [29]. This process may provide imprecise information due to various reasons such as wheels slipping, sampling of the encoder and locomotion errors[2]. As a consequence, getting more accurate results (i.e. estimating the actual position of the mobile robot) needs the help of environmental sensing.

2.2.3.2 Range Sensors

These sensors are used for measuring the distance to objects surrounding the robot in the space. They work by emitting a signal and then process the reflected signal to calculate the distance towards the obstacle that causes this reflection. The laser range finder and ultrasonic (sonar) sensors are the most important ones from among others.

Ultrasonic sensors work by generating sound waves toward an object and then detect the echo which is received back by the sensor. They determine the distance to the object by calculating the time interval between sending the signal and receiving the echo.

The laser rangefinder is a time-of-flight sensor that achieves significant improvements over the ultrasonic range sensor owing to the use of laser light instead of sound [22]. This

[2] Locomotion is the process of designing robot appendages and control mechanisms to allow robots to move efficiently [86].

sensor sends a laser pulse towards the object and then some light is sent back to a detector that measures the time of reflection (using phase-shift measurement [22]). Because lasers use light instead of sound, they can measure both very fast and with a detailed description of the field of view [3][30].

2.2.3.3 Cameras

Camera based sensing, termed *computer vision* [31], has not been widely used in mobile robots until recently due to the computational power needed for image processing as well as the high cost. However, the past decade has seen a rapid development regarding this method. When compared with other sensing techniques, vision based methods to navigation provide detailed information about the environment, which may not be available using combinations of other types of sensors [32].

Two types of vision based sensing exist; global and local [27]. In the global vision, the camera covers a large area in which the robot and the environment are in the scene (it is placed external to the robot). In the local vision, the camera is attached to the robot where it is used to explore the way locally.

2.2.4 Player/Stage Project

Player is an open-source software system developed at the University of Southern California Robotics Research Labs. It is a network oriented server [33] that allows control of robotic sensors and actuators. The player robot server offers a TCP socket interface to clients enabling them to access these sensors and actuators. Clients connect to Player and communicate with the devices by exchanging messages with Player over the TCP socket [23].

Accompanying Player is the robot simulator *Stage*, which enables developers to control virtual robots navigating inside a virtual environment. Various sensor models are also provided, including sonar, laser rangefinder, and odometry. Player clients which are developed using Stage will work with little or no modification with the real robots and vice versa [34].

[3] This refers to the fact that the detector measures both the brightness and the phase of the return signal.

The Player robot server and its simulation packages such as stage have become very popular and widely used in the world. The player/stage can run primarily under Linux or UNIX variants (Solaris and FreeBSD). It can also run under windows with the help of Cygwin virtual environment [35]. Developers can use any programming language such as C++ or Java in implementing their client programs.

2.3 State-of-the-Art

Mobile robot navigation is the process of generating a continuous path between an initial position and a prescribed goal location. Along such a path, the robot must avoid colliding obstacles, based on its knowledge and the sensorial information of the surrounding environment, so that it could reach the target position as safely and efficiently as possible.

The techniques that generate collision-free motion can roughly be classified to *global planners* (motion planning) and *local planners* (reactive navigation methods) [36]. Global methods are based on *a priori* information whereas local methods are based on *sensory* information.

In the following, a brief general overview of the motion planning approaches is introduced. Following this presentation we shed the light on the most significant reactive navigation approaches that motivated us to formulate this work, showing their advantages and disadvantages.

2.3.1 Motion Planning Techniques

Motion planning techniques attempt to generate an efficient collision-free path from the robot actual location to the goal while avoiding a static (previously known) set of obstacles. The advantage of these methods is that they can provide a global optimal solution for reaching the goal. However, motion planning algorithms require relatively complete knowledge about the world model with previously known obstacles. In fact, in mobile robots operating in unstructured environments, or in service and companion robots, the a priori knowledge of the environment is usually absent or partial [37]. Hence, new sensory perceptions must be integrated into a model. This will be time consuming and improper to handle real-time requirements. The classical motion planning planners can be categorized, as stated in [36],

into three major methods: *roadmap, cell decomposition* and *potential field* methods. The potential field method can be used as a global planner and a local planner as well [38]. The description of this method will be presented in the reactive navigation techniques section.

2.3.1.1 Roadmap Motion Planning

Roadmap approaches capture the connectivity of the robot's free space in a network of 1D curves or lines [22]. Whenever a roadmap is constructed, the best path that drives the robot from an initial position to its destination is selected. Examples of roadmap algorithms include *visibility graph and Voronoi diagram.*

- **Visibility graph:** Consists of all straight line segments connecting vertices that can see each other [39]. The shortest path from the robot's initial position to the goal position along the roads of the visibility graph is then calculated. Although this algorithm drives the robot extremely fast, it takes the robot as closely as possible to obstacles while navigating towards the goal.

- **Voronoi diagram:** Consists of the lines and curves formed by points in the free space that are equidistant from the nearest obstacles. This method achieves safe paths since it maximizes the distance between the robot and obstacles on the way to the goal. However, it is usually far from optimal in the sense of total path length [22].

2.3.1.2 Cell Decomposition Motion Planning

Cell decomposition algorithms split the space into two parts; obstacle space and free space. The free space is divided into simple connected regions called *cells*. The *connectivity graph* is then generated which describes the adjacency relation between free cells. After that, the best path is chosen which is formed from a sequence of adjacent traversal cells linking the initial and goal positions. Cell decomposition algorithms come in two varieties [40]: exact decomposition and approximate decomposition.

- **Exact decomposition:** The free space is divided into exact polygonal cells. The resulting configuration space will contain either completely free or completely

occupied cells. This method is good only in large, sparse environments in which the number of cells is small and as a result the computational complexity is reduced [22].

- **Approximate decomposition:** The whole space is split into very small cells (grid based environment). Any cell containing obstacles is deleted. The complexity in this method is not dependent on the density of the environment or the shape of obstacles as in the exact decomposition [22].

2.3.2 Reactive Navigation Techniques

Local reactive navigation methods depend on a perception-action process [41] that continuously generates collision-free motion commands dependent on feedback sensors (e.g. laser range finder [42]). Based on these observations, a robot can respond to dynamic obstacles, and work especially well in an unknown environment. Furthermore, only a small fraction of the world model is required in these approaches. As a consequence, a fast obstacle avoidance can be achieved with low computational complexity. The drawback is that these methods use a local fraction of the information available (sensory information). Then, it is difficult to obtain optimal solutions, and to avoid the trap situations [43].

2.3.2.1 Bug Algorithms:

Bug algorithms (e.g. [44 - 46]) are the simplest obstacle avoidance techniques that drive the robot safely towards its goal. The basic idea of these algorithms is to move toward the goal unless an obstacle is encountered, in which case, circumnavigate the obstacle until motion toward the goal is once again allowable [47] (a leaving condition is satisfied). The transition between the two cases is governed by a globally convergent criterion [48]. There are many variants of these algorithms (e.g. Bug 1 and Bug 2). The main difference between them is in defining the leaving condition.

With Bug 1 algorithm, as soon as an obstacle is detected at point H_i (hit point); the robot fully circles the obstacle in order to determine the closest point to the goal on its perimeter L_i (leave point). Then, the robot goes to this point and departures the obstacle towards the goal along a new line [49]. Fig. 2.3 shows a situation with two obstacles generated by the Bug 1 algorithm where the hit and leave points are identified.

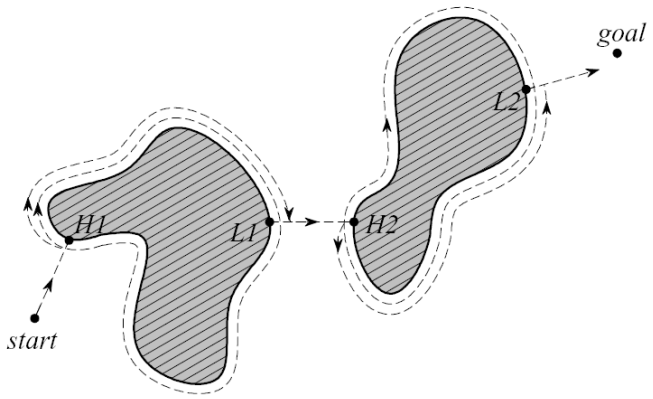


Figure 2.3: Bug1 algorithm with H1, H2, hit points, and L1, L2, leave points [22].

For Bug 2 algorithm, if the robot encounters an obstacle, it follows its contour starting at the hit point H_i until the line to the target is crossed at a point closer to the goal than the hit point (the leave point L_i). Then the robot resumes the progress along the line to the goal. This algorithm leads to shorter paths than Bug 1 but in some cases the path may get longer than Bug1 [49]. Fig 2.4 shows the path generated by the Bug 2 algorithm with the same situation of Fig. 2.3.

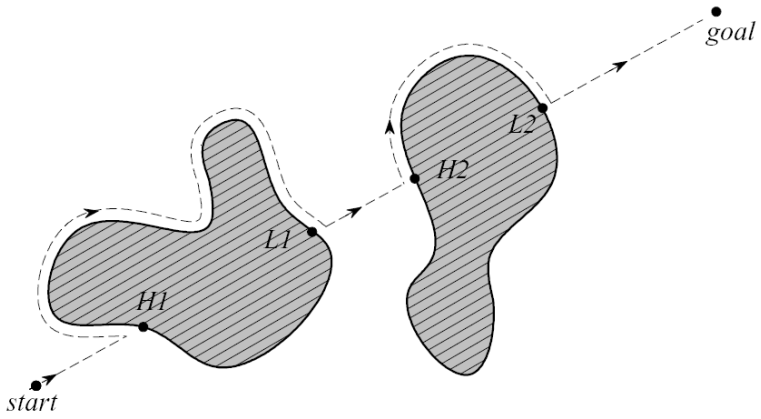


Figure 2.4: Bug 2 algorithm with H1, H2, hit points, and L1, L2, leave points [22].

Although Bug algorithms ensure that the robot will reach any reachable goal, they depend strongly on sensor accuracy and assume that all information about the environment is available. Moreover, the robot is assumed to be as a point in the configuration space without considering the robot kinematics.

2.3.2.2 Potential Field Methods

Potential field methods PFMs were firstly proposed by khatib [50] using the gravitational force field principle to generate collision free motion. This method assumes that the robot is a particle (point) moving in the space where the goal generates an attractive potential while each obstacle generates a repulsive potential (as shown in Fig. 2.5 [51]). If the potential field is considered as an energy field, its gradient will be a force. In this regard, obstacles exert a repulsive force while the target asserts an attractive force on the robot. The resultant vector sums of these forces are used to compute the robot's steering direction and motion equations. The robot terminates the motion when it reaches a point where the gradient is zero, signifying a minima [51]. If it occurs when reaching the goal, it is called a global minimum (the required state); else, it will be a local minimum which is undesired and traps the robot.

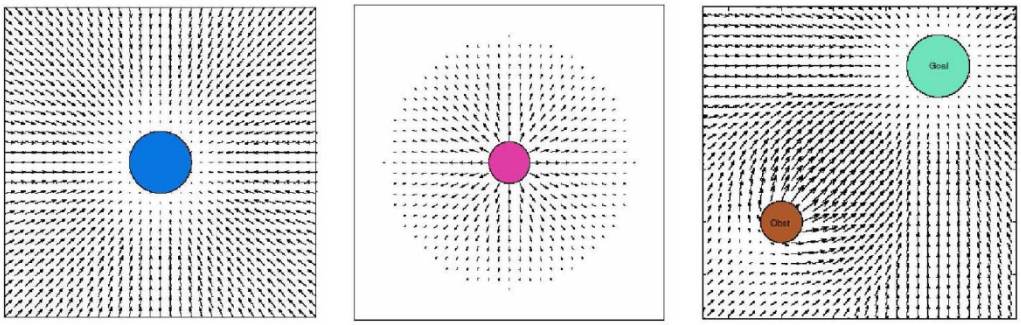


Figure 2.5: Typical potential fields [51].

The potential field method is considered fast and computationally efficient. However, Koren and Borenstein analyzed this method in [52] and explained its inherent limitations with experimental results. The well-known problem with PFMs is the trap situations due to local minima. A trap-situation occurs when the robot reaches a deadlock where the robot stops before reaching its goal. Many situations can cause traps such as navigating inside a U-shaped obstacle (Fig. 2.6a [37]) or reaching a situation where the total repulsive forces due to obstacles is symmetric to the attractive force due to the goal (Fig. 2.6b [37]). The local minima problem can be alleviated by random walks as presented in the Randomized Potential Field method [53]. Other problems may arise using this technique such as: No

passage between closely spaced obstacles, oscillations in the presence of obstacles and oscillations in narrow passages [52].

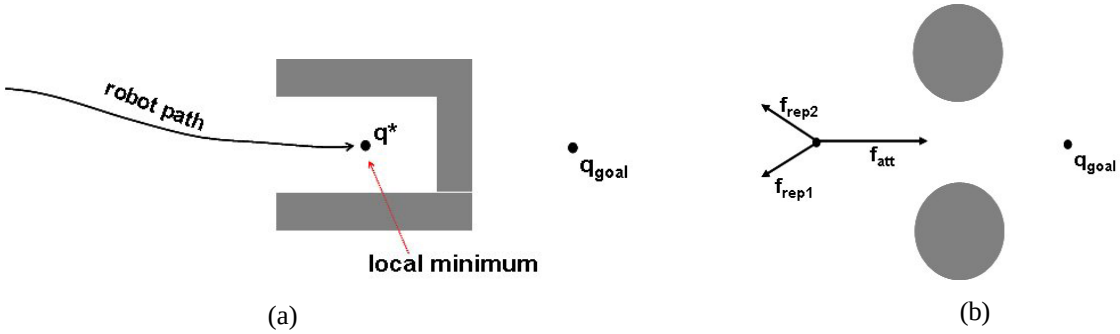


Figure 2.6: Local minimum situations. (a) Local minimum of total potential due to a U-shaped obstacle [37]. (b) Local minimum of total potential due to environment symmetry [37].

The Potential Field approach has been widely used by a large number of researchers. Therefore, many variations of this method have been developed and used (e.g. [54 - 61]). In the following two subsections, the Vector Force Field VFF method [60] and the Vector Field Histogram method [61] are presented, among others, since they are the most well known ones.

2.3.2.3 Virtual Force Field (VFF) Method

The Certainty Grid concept had been developed by Moravec and Elfes [62, 63] which represents the environment of the robot by a two-dimensional array of square elements called cells. Each cell contains a *certainty value* $c_{i,j}$ that shows the measure of confidence that an obstacle is located in the cell [61]. This method is mainly used to limit the inaccuracy of sensor readings. Using this idea for obstacle representation in combination with the potential field method for navigation, J. Borenstein and Y. Koren [60] developed the Virtual Force Field (VFF) concept. During its movement the robot maps the sensor readings into the Certainty Grid (named histogram grid in this method). A frame area around the robot in the Certainty Grid, named *active window*, is checked for the occupied cells (active cells). For each sensor reading, the VFF increments only one cell in the *histogram grid*. Therefore, it reduces the computational overhead as compared with the Certainty Grid method.

Each active cell asserts a repulsive force on the robot. The magnitude of this force is proportional to the certainty value, and is inversely proportional to the square of the distance between the cell and the center of the robot d^2 [60, 61]. The resultant repulsive force F_r is calculated by adding all repulsive forces. This value is then summed to the attractive force F_t, which pulls the robot to the goal, yielding the resultant force vector R as explained in Fig. 2.7.

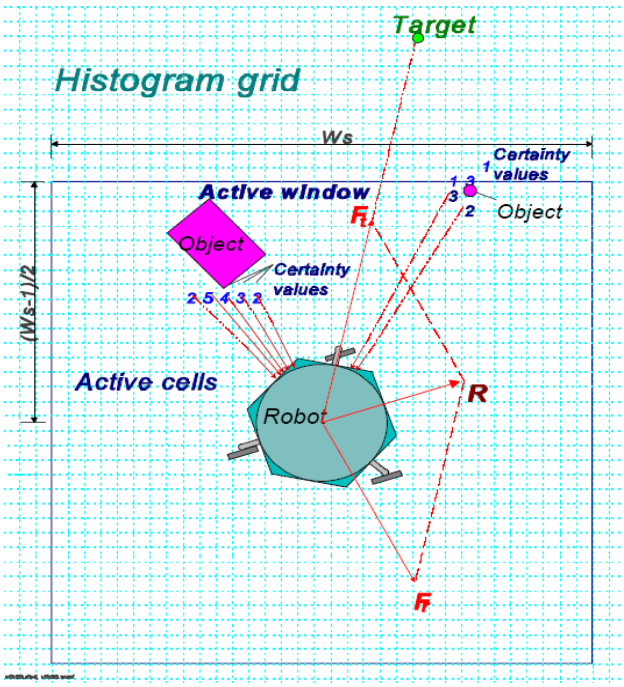


Figure 2.7: The Virtual Force Field concept: Each active cell exerts a repulsive force on the robot proportional to $c_{i,j}$, and inversely proportional to d^2. The total of the repulsive forces from all cells are added to the attractive force towards the goal yielding the force vector R [60].

The main advantage of this approach over the original potential field method [50] appears in reducing the sensor uncertainties due to its probabilistic nature. However, it has the same drawbacks of the potential field method addressed in [52] and mentioned above. Moreover, choosing a suitable cell size is a hard issue; a large one causes strict changes in the resultant force vector R, which decrease smoothness, while a small one adds a computational overhead.

2.3.2.4 Vector Field Histogram (VFH) Method

One of the inherent problems of the VFF method is the excessive drastic data reduction, which occurs as a result of reducing hundreds of data points (adding the repulsive forces from histogram grid cells to get the total repulsive vector) in one step to calculate the resultant force vector. The VFH method [61] comes to overcome this limitation by employing a *two-stage data reduction*, rather than the single stage introduced in the VFF method. The original paper on VFH [61] summarizes this method as follows:

- Builds a two-dimensional Cartesian histogram grid which is continuously updated from sensorial data (as in VFF).
- Chooses an active window around the robot and then maps the two-dimensional histogram active grid into a one-dimensional polar histogram. The resultant polar histogram comprises a number of sectors of a specific width as explained in Fig. 2.8.
- Calculates the best steering angle based on a cost function and then determines the motion commands. The cost function takes into account the goal direction, the robot's current orientation and the wheels angle.

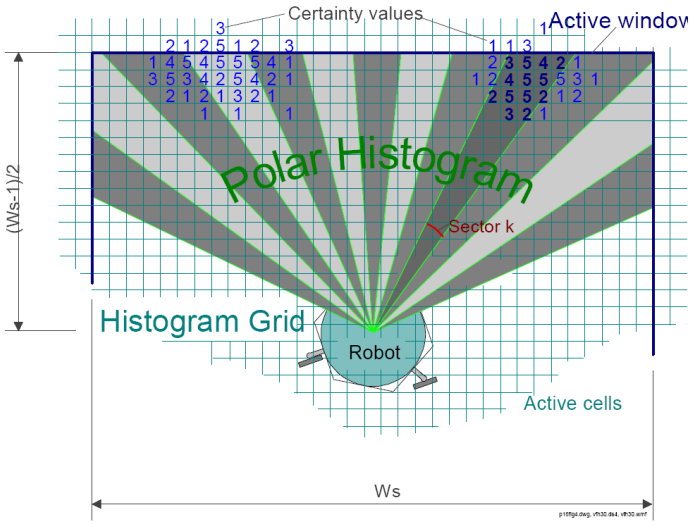


Figure 2.8: Mapping of active cells onto the polar histogram [61].

Despite that this method produces smoother behavior and allows robots to travel at faster speeds without getting unstable [64], it, like the potential field approach, can get trapped in

local minima. An enhancement for the VFH, named VFH+ [65], was introduced. VFH+ takes into consideration the shape of the robot and the robot trajectory. As a consequence, a smoother behavior is achieved. The VFH+ was further developed in [66] to lead to the VFH* method. VFH* adds an A* global search in order to avoid local minima [38]. However, the VFH and even the enhancements VFH+ [65] and VFH* [66] present the difficulty to move between close obstacles due to the tuning of a threshold which depends on the obstacle density.

2.3.2.5 Elastic Bands

The Elastic Band method [67] considers a previously planned path and attempts to deform or modify this path locally during navigation. This enables the robot to adapt to local changes in the environment. Moreover, the new generated path will be smoother and shorter than the original one, keeping in mind the fact that most of path planners generate trajectories with sharp turns. First, an initial path is sampled with bubbles to create an elastic band or bubble band, where each bubble represents free-space in which the robot can travel without collision [68] (as shown in Fig 2.9). Two forces affect the shape of the path; Repulsive force from obstacles which repels the robot away from them, attractive force from neighboring bubbles simulating the tension in a stretched elastic band and removes any slack in the path [67].

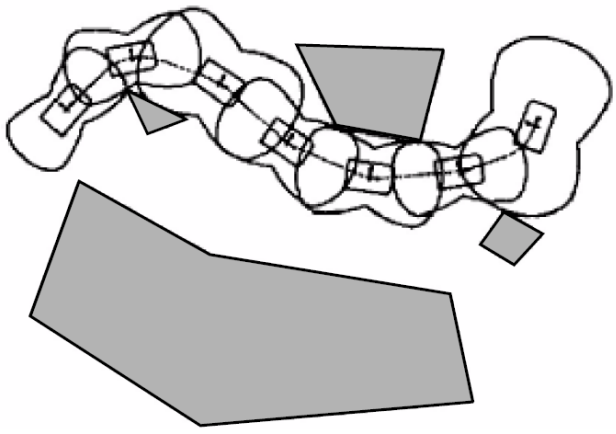


Figure 2.9: A typical bubble band [22].

This method modifies the robot trajectories in real-time to avoid colliding obstacles and overcome sensor inaccuracies, with the added advantage of its ability to consider the actual

shape of the robot. The drawback of this method is that it is applicable only when the environment is known previously. Therefore, it can be used for a fast re-planning phase, during execution, in the presence of dynamic environments [38]. Furthermore, avoiding traps due to U-shaped obstacles dynamically created is not guaranteed in these methods [43]. The original elastic band method [67] was developed for holonomic robots. However, an extension for this method was made to accommodate non-holonomic vehicles in [69].

2.3.2.6 Curvature Velocity (CVM) Method

The Curvature Velocity (CVM) method [70] differs from the above mentioned techniques in that it considers the kinematic and dynamic constraints of the robot while computing the motion commands. It treats the obstacle avoidance as a constrained optimization in the velocity space [71], which consists of rotational velocity w and translational velocity v, assuming that the robot only moves along arcs of circles (called curvatures). Two constraints are taken into consideration:

- The constraint that comes from the physical limitation of the robot in speed and acceleration (i.e. $-v_{max} < v < v_{max}, -w_{max} < w < w_{max}$).
- The constraint that are derived from the sensorial data indicating the presence of obstacles. Obstacles are first represented in the Cartesian space and then mapped to the velocity space by measuring the distance to the obstacle that the robot would travel before hitting it. This is done for all curvatures that lie within c_{min} and c_{max} as shown in Fig. 2.10.

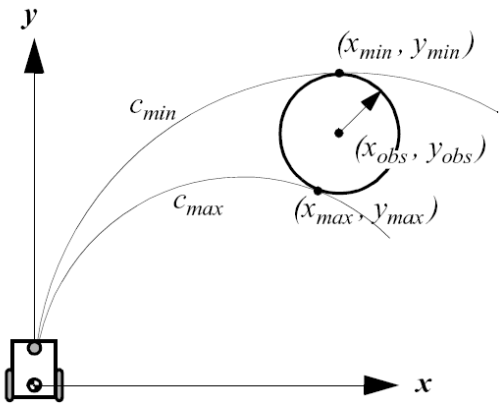


Figure 2.10: Tangent curvatures for an obstacle [70].

After identifying the velocities that satisfy the above constraints, an objective function is maximized, which prefers high speeds, curvatures that travel longer before hitting obstacles and should try to orient the robot to head in the desired goal direction [72].

The CVM method considers the kinematics and dynamics of the robot and as a result it can generate smoother and faster paths. However, assuming only fixed arc robot trajectories and circular shaped obstacles is a simplification in this method which can cause serious problems. The Lane Curvature (LCM) method [73] overcomes these problems by taking the free directions into account in a first step and then the collision free arc length (dynamics). But the problem of trapping in local minima remains a drawback in the two approaches.

2.3.2.7 Dynamic Window (DW) Approach

There are several approaches that take into account the dynamic constraints of the robot and choose a steering command rather than a moving direction (e.g. the CV methods [70, 73] mentioned above). But it was the Dynamic Window Approach [74] that won more popularity in the scientific community [75]. This reactive approach works by adding constraints to the velocity space of the robotic system considered, and then choosing the speed that satisfies all constraints and maximizes an objective function. The velocity space is all possible sets of translational v and rotational w velocities. The DW approach assumes that robots move only in circular arcs in order to accommodate the non-holonomic constraints. As stated in [74], the algorithm solves the problem in two steps:

Search space: Let $\mathbb{V}$ the set of all possible velocities[4], the DW approach, as stated above, considers only circular trajectories. This reduces the search space into a 2D velocity search space (denoted in Fig. 2.11 by $\mathbb{V}_s$). This search space is then reduced to the admissible velocities $\mathbb{V}_a$, which allow the robot to stop before hitting an obstacle, in order to safely drive the robot towards its goal. The search space is further restricted to velocities that can be reached within a short time interval given the vehicle accelerations (the dynamic window $\mathbb{V}_d$ shown in Fig. 2.11). The resulting search space (the white area in the figure) will be:

$$\mathbb{V}_r = \mathbb{V}_s \cap \mathbb{V}_a \cap \mathbb{V}_d \tag{2.1}$$

[4] Using $\mathbb{V}$ here denotes the translational and rotational velocities together (i.e. v and w). By "all possible velocities", we mean velocities that do not exceed the maximum allowed values.

Optimization: The motion commands are chosen from among the set of resultant velocities V by maximizing the following objective function:

$$G(V) = \alpha \cdot \text{Goal}(V) + \beta \cdot \text{Clearance}(V) + \gamma \cdot \text{Velocity}(V). \quad (2.2)$$

This function is a compromise among Goal (V), which selects velocities that make the best progress to the goal, Clearance (V), which favors velocities far from the obstacles, and Velocity (V) that chooses high speeds [25].

The DW approach drives the robot smoothly and at high speed as a result of considering the kinematic and dynamic constraints of the robot. But it can be trapped in local minima since no enough information about the environment is available. To overcome this limitation, an enhancement of the DW approach was introduced in [76], which generates a collision-free motion by considering the information of a real map of the environment plus data from the sensors (adds global thinking to the algorithm). But this will make it suitable for static or low dynamic environments only.

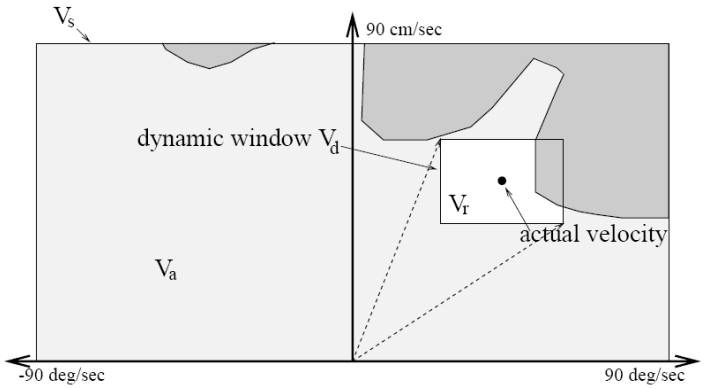


Figure 2.11: The search space in the dynamic window approach [74].

2.3.2.8 Nearness-Diagram Navigation (ND) Method

It is still a robotic challenge to safely drive the robot in *very cluttered, dense and complex environments*, which are usually the case in most robotic applications [43]. However, good results in very cluttered, complex and dense environments have been reported using the Nearness Diagram Navigation (ND) method [15].

The Nearness-Diagram (ND) is a reactive collision avoidance technique that performs a high-level information description from sensory data to obtain a motion command later on.

This method is similar to the earlier developed Vector Field Histogram approach [61] in extracting a description of regions surrounding the robot in order to select the best one (see Fig. 2.12). The ND divides navigation behavior into five situations to take action as needed based on the situated-activity paradigm design [77]. Afterwards, authors of ND reformulated the motion laws and added a sixth situation to lead to the ND+ method [78].

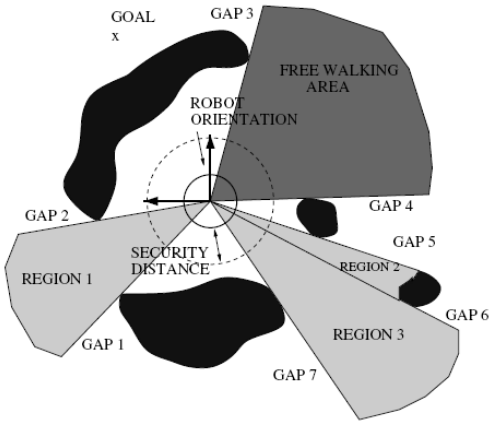


Figure 2.12: Analyzing openings and choosing the best one (free walking area) in ND [15].

In the first step, the data coming from sensors are analyzed to determine the structure of obstacles surrounding the robot. Based on these analyses the current robot situation is identified dependent on four criteria; safety criterion, dangerous obstacle distribution criterion, goal within the free walking area[5] criterion, and free walking area width criterion. Associated to each situation there is an action that computes the motion to adapt the behavior to the case represented by each situation [25]. The following defines the six situations and their associated actions as stated in [15] (see Fig. 2.13):

- High safety goal in region (HSGR): The situation is HSGR when the goal position lies inside the free walking area and there are no obstacles within the security zone around the robot. The associated action will drive the robot towards the goal location.
- High safety wide region (HSWR): The situation is HSWR when the free walking area is wide. The associated action moves the robot towards the side of the free walking

[5] The free walking area defines the “navigable” opening closest to the goal.

area closest to the goal, plus a deviation to prevent obstacles on that side from entering the security zone.

- High safety narrow region (HSNR): The situation is HSNR if the free walking area is narrow. The action moves the robot towards the center of the free walking area.
- Low safety goal in region (LSGR): The situation is LSGR when the goal location lies inside the free walking area and there are obstacles within the security zone. The associated action drives the robot towards the goal plus a deviation that depends on the distance to the closest obstacle.
- Low safety 1 side (LS1): The situation is LS1 if there are obstacles inside the security zone on one side of the free walking area. The associated action is the same as in HSWR plus a deviation dependent on the distance to the closest obstacle.
- Low safety 2 side (LS2): The situation is LS2 if there are obstacles inside the security zone on the two sides of the free walking area. The associated action is the same as in HSNR plus a deviation dependent on the distance to the closest obstacle.

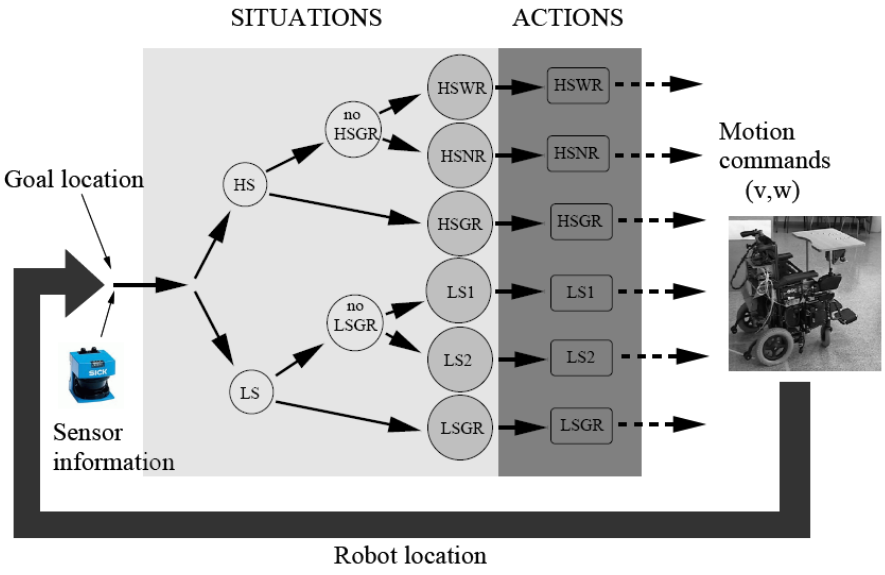


Figure 2.13: The Nearness-Diagram Navigation method design [78].

The robot shape, kinematic and dynamic constraints are taken into consideration in [79]. Also, a global reasoning was added to this approach, named Global Nearness-Diagram

(GND), in [75]. The Nearness-Diagram Navigation method has a lot of advantages over all previously mentioned approaches [15]; avoiding trap situations, generating oscillation-free motion, possibility of choosing a motion direction far away from the goal or towards the obstacles, overcomes the difficulty of tuning parameters which is a problem in most of the existent approaches. Robust ND-based navigation has been demonstrated in very dense, cluttered and complex environments [80]. However, the drawback of this method is that it generates slower and less smooth trajectories as compared with other techniques (e.g. the Dynamic Window (DW) approach [74]).

The Nearness-Diagram Navigation+ (ND+) method was further developed in [16] and entitled the Smooth Nearness-Diagram Navigation (SND) method. This method proposes a single motion law to be applicable to all possible configurations of surrounding obstacles [16] (rather than the six motion laws in ND+). In this regard, extracting the motions commands are dependent on all obstacles falling within the security zone around the robot (it is called threats in this approach), not just the closest two. As a consequence, smoother robot trajectories are achieved.

The SND algorithm can be summarized in four steps as follows [16]:

- The sensorial information is periodically analyzed to determine the structure of obstacles and identify the openings (gaps) surrounding the robot.
- The best heading which makes the progress towards the goal (i.e. the navigable gap closest to the goal) is chosen from among the list of detected gaps. Setting the direction of movement depends on the width of the gap and the goal position.
- The direction of movement is deflected away from all nearby obstacles (i.e. obstacles falling within the security zone around the robot). This is done by computing the total weighted deflection from all obstacles.
- After identifying the net direction, the motion commands are calculated. To maintain safety near obstacles, the translational speed of the robot is controlled according to the closest obstacle.

The SND approach, as mentioned above, achieves smoother robot trajectories than the ND+ method. However, while preparing for a competition organized by SICK AG[6], we carried out some experiments on a real robot using the SND algorithm. These experiments demonstrated that the robot can be trapped in narrow corridors where a large number of obstacles exist on one side of the corridor as compared with the other. Solving this problem was the beginning that motivated us to formulate this work.

2.3.2.9 Overview of Reactive Navigation Methods

Most of the well known reactive collision avoidance methods can be classified as directional approaches and velocity space approaches [81]. All previously mentioned techniques are directional ones except the Dynamic Window (DW) [74, 76] and the Curvature Velocity (CV) [70, 73] methods.

For directional approaches, the navigation problem is divided into two parts [81]. First, sensory information is analyzed for finding a proper direction. Second, the robot is controlled to move towards that direction.

Velocity space approaches take into account the dynamic and kinematic constraints. The velocity space is searched for speeds that satisfy these constraints. The velocity that satisfies all constraints and maximizes an objective function is then chosen to be applied as a motion command for the robot.

Velocity space approaches have faster and smoother behavior than directional methods. In addition, it can avoid severe accelerations and decelerations. However, the local minima problem can appear in these approaches.

[6] Preparing the SND algorithm for the competition, named the SICK Robot Day 2009, assigned to us by the GET lab in Paderborn University, Germany, which was one of the participants of the competition.

Chapter 3

Closest Gap Navigation

3.1 Introduction

As stated in the previous two chapters, robots are usually required to move in very cluttered, dense and complex environments. Navigation in such environments, particularly with the existence of difficulties affecting the navigation process (mentioned in chapter 2), is still a robotic challenge. Many existing reactive navigation methods have problems in dealing with these environments. Some drawbacks of these approaches are: the local minima problem, deadlock, oscillatory behavior and the computational complexity. This chapter, as will be shown later on, introduces a new local reactive navigation scheme that overcomes or at least alleviates these shortcomings. Designing this approach depends mainly on two previous works; the Nearness-Diagram (ND) and the Smooth Nearness-Diagram (SND) navigation methods [15, 16].

We call the *Closest Gap* (CG) the basis for analyzing *gaps* in our design (a gap is a potential free path wide enough for the robot to move through). CG is able to find gaps that are directly in front of the robot and cancels others that are not necessary. Hence, the computational complexity is reduced, oscillations are alleviated and a smooth behavior will be achieved. The navigable gap closest to the goal is then chosen from among these gaps taking into consideration the angular width of the robot vision. The main contribution of this approach in calculating the motion commands is that it considers the ratio of threat counts on the two sides of the robot and provides stricter behavior against the closest obstacles. As a consequence, the robot is capable of avoiding the SND deadlock problem, which occurs in narrow corridors where the difference in the number of threats on its sides is high, while

keeping the smoothness behavior. Hence, a robust reactive navigation algorithm for highly cluttered environments is obtained.

This chapter is organized as follows: Section 3.2 presents the proposed reactive collision avoidance method. Simulations are discussed in Section 3.3, while Section 3.4 shows the experimental results. Finally, Section 3.5 highlights our conclusions.

3.2 The Reactive Obstacle Avoidance Method

The goal of this section is to explain the Closest Gap Navigation method (CG) for avoiding dynamic obstacles in complex and cluttered environments. The CG method works as follows: first, the information from the laser rangefinder sensor is periodically analyzed to identify the gaps surrounding the robot as explained in subsection 3.2.2. In order to reach the goal, the navigable gap closest to the goal is chosen. The direction of motion towards this gap is determined as described in subsection 3.2.3. Finally, the real time deflection from obstacles while moving towards the goal is introduced in subsection 3.2.4.

3.2.1 Definitions

In this subsection, some definitions introduced in [16] are explained since it will be helpful in understanding the other subsections. It is assumed that the positive x axis is in front of the robot and the positive y axis is the normal on its left side. The angles always have an absolute value less than π. Negative angles are on the right side of the robot whereas positive angles are on the left.

Let $\mathbb{S}$ be the unit circle attached to the robot's reference frame. For any two angles $\alpha, \beta \in \mathbb{S}$, the angular distance between them relative to the robot is:

$$\mathrm{dist}(\alpha, \beta) = \min\{\mathrm{dist}_{cc}(\alpha, \beta), \mathrm{dist}_{c}(\alpha, \beta)\} \tag{3.1}$$

where the functions $\mathrm{dist}_{cc}(\alpha, \beta)$ and $\mathrm{dist}_{c}(\alpha, \beta)$ are defined as:

$$\mathrm{dist}_{cc}(\alpha, \beta) = (\beta - \alpha) \bmod 2\pi \tag{3.2}$$

and

$$\mathrm{dist}_{c}(\alpha, \beta) = (\alpha - \beta) \bmod 2\pi. \tag{3.3}$$

Sometimes, an angle θ may become greater than π or less than $-\pi$ during calculations. In order to map this angle into the right value in $[-\pi, \pi[$, the projection function is defined as:

$$\text{proj}(\theta) = \big((\theta + \pi) \bmod 2\pi\big) - \pi \tag{3.4}$$

Finally, the saturation function is used to limit a value between two boundaries. Assume that a < b, the *sat* function is defined as follows:

$$\text{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x \leq a, \\ x, & \text{if } a < x < b, \\ b, & \text{if } x \geq b. \end{cases} \tag{3.5}$$

3.2.2 Analyzing Gaps

The main part in analyzing the data perceived by sensors is to identify the *gaps* surrounding the robot. Before explaining the details, assume the following:

1) The first scan point is (0) and the final one is (n).
2) (L) is the list of obstacle points perceived (detected).
3) The maximum range of the sensor is denoted by $(d_{\max})$.
4) $\text{d}(A, B)$ returns the distance between points A and B.
5) We will take all scan points into consideration (not divide into sectors) in this algorithm.

The *inputs* of the algorithm are:

1) The robot location $(\text{x}_{\text{robot}})$ and robot radius R.
2) The maximum range of the sensor $(d_{\max})$.
3) A list (L) of obstacle points where an obstacle is $(\text{O}_{\text{i}}^{\text{L}})$.

The *output* of the algorithm is the list of gaps detected.

Extracting gaps can be summarized in two steps; the first one implies finding the list of all gaps that are seen by the robot dependent on fetching discontinuities and the other removes unusual gaps from this list. But, before explaining the two steps, let us define two

types of discontinuities that occur between two adjacent scans i and j (assuming the original order of scans in the forward and backward loops explained below).

Type 1 discontinuity: Occurs when the difference between the depth measurements of scans i and j exceeds the robot diameter.

$$d(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) - d(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{i}}^{\mathrm{L}})) > 2R.$$

Type 2 discontinuity: Occurs if one of the two measurements returns the maximum sensor range.

$$\mathrm{d}(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) = d_{\max} \text{ AND } \mathrm{d}(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{i}}^{\mathrm{L}}) < d_{\max}$$

if $j > i$, a *rising* discontinuity occurs at scan i; else, it will be a *descending* discontinuity at scan j. Type 1 discontinuity has a higher priority than type 2.

Now, we present the two steps of the algorithm as follows:

Step 1: We scan for gaps twice; first by detecting rising discontinuities while travelling from scan 0 to $n-1$ (forward search) and then through fetching descending discontinuities travelling from scan $n-1$ to 0 (backward search). Assume that the first rising discontinuity occurs at scan number i in the forward search. This scan determines the first side of the gap (e.g. points D and H in Fig. 3.1). Finding the second side depends on the discontinuity type as shown below.

1) For a type 1 discontinuity:

   Let $\mathrm{S}^{+} = \{i+1, \ldots, n-1\}$ be the set of all scans after scan i. The second side of the gap will be at scan number $j \in \mathrm{S}^{+}$, which satisfies the shortest distance to the first side; the angular distance travelled must be less than π (e.g. Fig. 3.1, point E).

   $$\mathrm{d}(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) \leq \mathrm{d}(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{k}}^{\mathrm{L}}) \text{ AND } \mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{i}}, \theta_{\mathrm{j}}) \leq \pi$$

   where k is any scan number $\in \mathrm{S}^{+}$.

2) For a discontinuity type 2:

   Continue scanning after scan i until a descending discontinuity, either type 1 or type 2, is detected at scan number j. In this case, the second side will be at scan $j+1$ (see point I, Fig. 3.1).

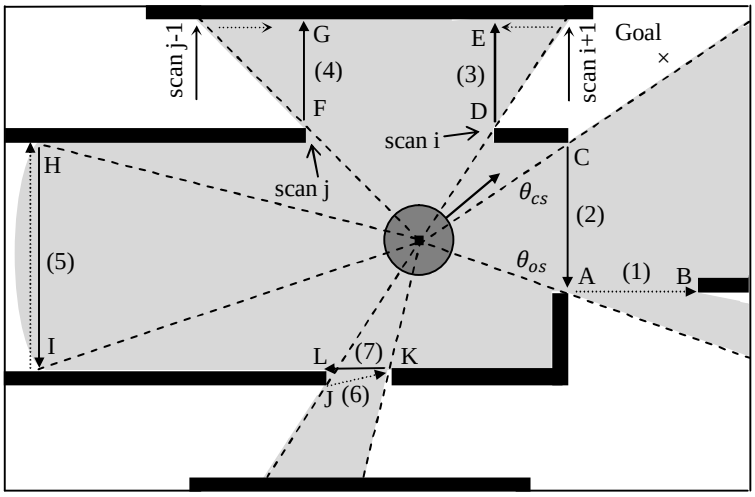


Figure 3.1: Analyzing gaps by the CG method.

In order to find the rest of forward gaps, resume the process starting from scan $j + 1$. This search produces gaps 1, 3, 5 and 6 in Fig. 3.1.

In the backward search, assume that a descending discontinuity occurs at scan number j. The second side of the gap will be at scan $j + 1$ (e.g. points F and I in Fig. 3.1). In order to find the first side, the following is done:

1) For a detected discontinuity of type 1:

   Let $S^- = \{j - 1, \dots, 0\}$ be the set of all scans before scan j. The first side of the gap will be at scan number $i \in S^-$, which satisfies the shortest distance to the second side; the angular distance travelled must be less than π (see point G in Fig. 3.1).

$$d(O_i^L, O_{j+1}^L) \leq d(O_k^L, O_{j+1}^L) \text{ AND } \text{dist}_c(\theta_{j+1}, \theta_i) \leq \pi$$

   where k is any scan number $\in S^-$.

2) For a discontinuity of type 2:

   Pass through the scans that come before scan j until a rising discontinuity, either type 1 or type 2, is fetched at scan number i. This scan determines the second side (e.g. Fig. 3.1, point H). In case of type 2, delete the gap since it has been considered in the forward loop.

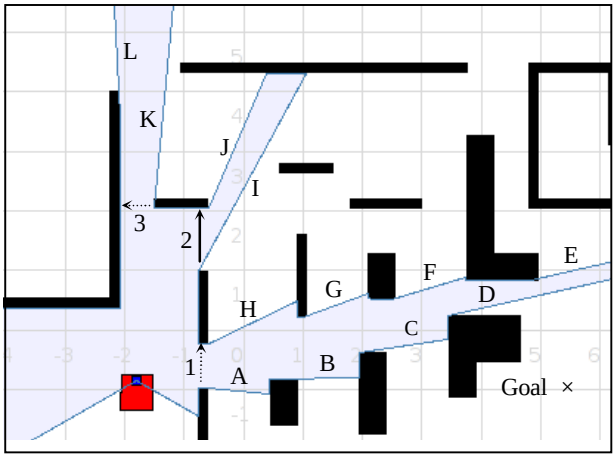


Figure 3.2: Analyzing gaps showing advantages of the CG over the SND algorithms.

In order to find the rest of backward gaps, resume the process from scan number $i - 1$. Gaps 7, 4 and 2 in Fig. 3.1 are fetched in this search.

Step 2: After completing the forward and backward loops, we get the list of gaps (G). Eliminate from G every gap that exists inside another gap (e.g. gaps 1 and 6 in Fig. 3.1), and then remove from the remaining gaps any gap that has a width less than the robot diameter (e.g. gap 7 in Fig. 3.1). The following steps explain the idea.

1) If $\exists a, b \in G / (a_i \geq b_i \text{ AND } a_j \leq b_j)$, then eliminate a,

2) If $\exists c \in \grave{G} / d(c_i, c_j) < 2R$, then eliminate c,

where x_i and x_j denote the first and second sides of gap x, $\grave{G} \in G$ is the list of remaining gaps after step 1.

To demonstrate the strength of this method for analyzing gaps, we took a snapshot of the Player/Stage simulator which shows how gaps are fetched in the SND and the CG methods (Fig. 3.2). The SND method detects twelve gaps which are labeled *A* to *L* while the CG algorithm returns only one gap labeled number 2. This is done in the CG method as follows: The rising discontinuities *A*, *I* and *K* form the gaps 1, 2 and 3 in the forward search. The gaps which are detected during the backward search from the descending discontinuities *L*, *J* and *H* are deleted in step 2 of the algorithm, since they are contained inside gaps 1, 2 and 3. Gaps 1 and 3 are then deleted since their width is less than the diameter of the robot. It is obvious

from the figure that the gaps from *A* to *H* do not have to be considered at this point since gap 1 leads to them. Similar situations may arise many times during navigation, particularly in complex environments. Avoiding this decreases the possibility of oscillations that may occur from the great number of unnecessary gaps. Also, this will reduce the computational complexity needed for calculations.

Once the list of gaps is assembled, the *navigable* one closest to the goal is selected. It is identified by selecting the gap with the side closest to the goal (the angle between this side and the goal is the minimum). Then, this gap is checked if it is navigable. If it is not, another gap is selected in the same manner and the process is repeated until a navigable gap is found, or no gaps exist. We refer to the side of the selected gap closest to the goal by the angle θ_{cs} and the other side of this gap by the angle θ_{os}. The closest gap in Fig. 3.1 is gap 2. The closest side to the goal occurs at point *C* while the other side is at point *A*.

Remark 3.1 (checking navigability): To verify whether a gap is navigable; if the goal location lies inside the *closest gap*, check for an existing path to the goal as stated in the appendix-A of the ND paper [15]. Else, check for an existing path to the middle of the gap that is the point between the first and the second side of the gap.

3.2.3 Determining Motion Direction

In this subsection, a procedure to determine the motion direction, based on the analysis made in subsection 3.2.2, is presented. First of all, if there is a direct way to the goal (see Appendix A for description of the algorithm), do not look at gaps at all and set the motion direction $\theta_{\mathrm{md}} = \theta_{\mathrm{goal}}$. Else, the robot shall pass through the *closest gap* assigned in subsection 3.2.2.

As previously mentioned each gap has two sides, one is to the left of the other. We call it a *left side* of the gap and the other is a *right side*. To go safely through the closest gap as a step towards the goal, we use the two angles defined in [16].

$$\theta_{\mathrm{scs}} = \begin{cases} \theta_{\mathrm{cs}} - \arcsin\left(\frac{R+D_{\mathrm{s}}}{D_{\mathrm{cs}}}\right), & \text{if } \theta_{\mathrm{cs}} \text{ is a left side,} \\ \theta_{\mathrm{cs}} + \arcsin\left(\frac{R+D_{\mathrm{s}}}{D_{\mathrm{cs}}}\right), & \text{if } \theta_{\mathrm{cs}} \text{ is a right side,} \end{cases} \tag{3.6}$$

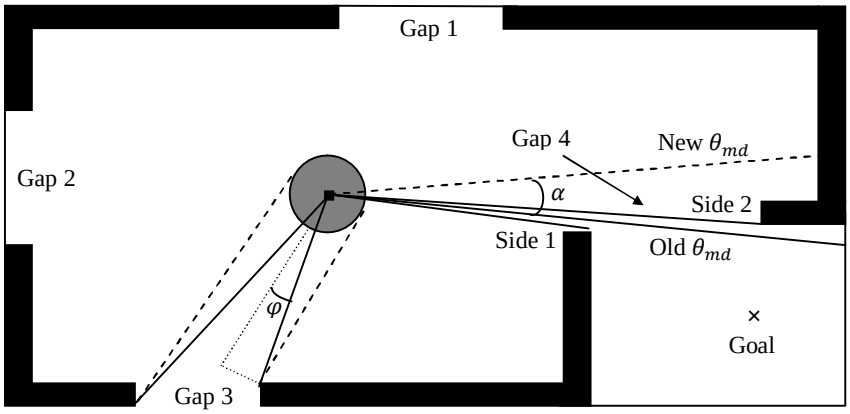


Figure 3.3: Modifying θ_{md} according to the angular width of the closest gap with respect to the robot vision.

$$\theta_{\text{mid}} = \begin{cases} \theta_{\text{cs}} - \text{dist}_{\text{c}}(\theta_{\text{cs}}, \theta_{\text{os}})/2, & \text{if } \theta_{\text{cs}} \text{ is a left side,} \\ \theta_{\text{cs}} + \text{dist}_{\text{cc}}(\theta_{\text{cs}}, \theta_{\text{os}})/2, & \text{if } \theta_{\text{cs}} \text{ is a right side,} \end{cases} \tag{3.7}$$

where D_{s} and D_{cs} are the safe distance and the distance to the obstacle point creating the gap side closest to the goal from the center of the robot. θ_{md} is chosen based on the following: the location of the goal (if θ_{goal} falls between θ_{cs} and θ_{os}, set $\theta_{\text{md}} = \theta_{\text{goal}}$) and the width of the gap (choose θ_{scs} if the gap is wide or θ_{mid} if it is narrow). The following equation explains that:

$$\theta_{\text{md}} = \begin{cases} \theta_{\text{goal}}, & \text{if } \theta_{\text{rs}} \leq \theta_{\text{goal}} \leq \theta_{\text{ls}}, \\ \theta_{\text{mid}}, & \text{if } \text{dist}(\theta_{\text{cs}}, \theta_{\text{mid}}) < \text{dist}(\theta_{\text{cs}}, \theta_{\text{scs}}), \\ \theta_{\text{scs}}, & \text{otherwise}, \end{cases} \tag{3.8}$$

where θ_{rs} and θ_{ls} are the angles toward the left and right side of the closest gap.

Our approach adjusts θ_{md} by adding another angle (α) to provide a smoother and safer behavior than SND and ND+. Fig. 3.3 shows the usefulness of this. It can be seen that four gaps surround the robot. The closest one to the goal is gap 4, which has a narrow width. So, as stated above, the motion direction is through the middle of the gap (θ_{mid}). It is clear that the gap aperture, as seen by the robot, is very small. In this regard, the direction of motion will be nearly towards side (1). The robot will move in this direction till it gets close to the obstacles at this side. Then, the real time obstacle avoidance algorithm introduced hereinafter in subsection 3.2.4 will deflect the direction away from these obstacles. This reduces the smoothness and may cause the robot to collide with obstacles if the robot is fast or the safe

distance is short. Gap 3 has an opening angle wide enough to fit the robot diameter. In this case, there is no problem. To solve this drawback; we propose to modify θ_{md} to let the gap fit the robot diameter.

Let us first define the angle φ which is the minimum to fit the radius of the robot:

$$\varphi = \arcsin\left(\frac{R}{D_{\mathrm{ns}}}\right) \tag{3.9}$$

where R and D_{ns} are the robot radius and the distance from the center of the robot to the gap side closest to the robot.

Now, suppose $\beta = 2\varphi$ is the most narrow angle which fits the robot diameter. The real width of the gap as seen by the robot is defined as follows:

$$w = \begin{cases} \mathrm{dist}_{\mathrm{c}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}}), & \text{if } \theta_{\mathrm{cs}} \text{ is a left side,} \\ \mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}}), & \text{if } \theta_{\mathrm{cs}} \text{ is a right side,} \end{cases} \tag{3.10}$$

Now, α can be defined as:

$$\alpha = \mathrm{sat}_{[0,\beta]}(\beta - w) \tag{3.11}$$

where the sat operator caps α at 0 when $w \geq \beta$ (since there is no need to modify θ_{md} in this case), at β when w = 0 (the maximum) and $(\beta - w)$ when $0 < w < \beta$.

Finally, θ_{md} is adjusted as follows:

$$\theta_{\mathrm{md}} = \begin{cases} \theta_{\mathrm{md}} - \alpha, & \text{if } D_{\mathrm{ls}} < D_{\mathrm{rs}}, \\ \theta_{\mathrm{md}} + \alpha, & \text{else,} \end{cases} \tag{3.12}$$

where D_{ls} and D_{rs} are the distances to the left and right side of the closest gap, respectively, from the center of the robot. In Eq. (3.12) the angle α is added or subtracted dependent on the sides of the closest gap to ensure that θ_{md} will force the robot to move towards the gap.

3.2.4 Real Time Reactive Navigation Method

After analyzing sensory information and getting the motion direction, the robot should be deflected away from obstacles surrounding it during motion. Hence, the direction of motion θ_{md} obtained above is adjusted to avoid the risk of collision with these obstacles (especially dynamic ones). In order to solve this issue, we propose an algorithm which is an evolution of the one introduced in the SND. As compared with the SND, the key difference in this

approach is that it generates safer paths and avoids deadlocks which occur in some cases without affecting the smoothness property as shown later on. The proposed solution is described as follows.

Each obstacle from among N obstacles falling within the safe distance D_s around the robot imposes a threat (t_i) dependent on the proximity of this obstacle to the robot boundary [16].

$$t_i = \text{sat}_{[0,1]} \left(\frac{D_s - D_i}{D_s}\right) \tag{3.13}$$

where D_i is the distance to the i^{th} obstacle point measured from the robot boundary. The value of the threat is 0 when the obstacle is outside D_s and 1 if the robot touches the obstacle.

Dependent on the threat calculated for each obstacle, a deflection from the direction of motion θ_{md} will be applied to avoid each of these obstacles [16].

$$\delta_i = t_i \cdot \text{proj}(\text{dist}_{cc}(\theta_i + \pi, \theta_{md})) \in [-\pi, \pi[\tag{3.14}$$

where θ_i is the angle towards the i^{th} obstacle point and $\text{proj}(\text{dist}_{cc}(\theta_i + \pi, \theta_{md}))$ the path length from the angle opposite the obstacle i to θ_{md} travelling counterclockwise. This value is multiplied by t_i to make the deflection dependent on the proximity of the obstacle to the robot.

In the SND approach, all threats on the two sides of the robot falling within the safe distance are considered while calculating the total weighted deflection. If one side has a large number of obstacles (threats) compared with the other, a high deflection will be applied towards the side with fewer threats. This enforces the robot to hit obstacles on that side if the gap is narrow. The problem increases when D_s is enlarged since it will cover more area containing more obstacles and so the difference between the two sides increases. This drawback does not exist in the ND+ approach since it considers only the closest obstacle point on the left and right of the robot. However, deflecting the direction of motion from the closest obstacle causes sharp changes in the trajectory of the robot which reduces smoothness. This problem can be solved by considering the ratio of threat numbers on the two sides as shown hereinafter. Another drawback increasing the problem stems from the fact that the weight (t_i^2) used in SND is not strictly for close obstacles since it is between 0 and 1. Its square does not differentiate strongly between close and farther obstacles relative

to the robot. As the safe distance increases, the threat difference between two obstacles on different positions decreases causing a promotion of the problem.

Our proposed solution extends the difference between threats and behaves stricter when an obstacle gets closer to the robot. This is achieved through modifying the function calculating the weight as follows:

$$w_{\mathrm{i}} = \frac{1}{(1-t_{\mathrm{i}})^{k}} \tag{3.15}$$

where k defines the strength of the weight. Increasing the value of k ensures safer behavior through moving away from the closest obstacles (it must not be high since it decreases smoothness). $K = 1$ is reasonable but if the safe distance is highly increased, a value of 3 is good. As t_{i} gets closer to 1, the output increases more sharply. This is required to ensure a stricter deflection from the closest obstacles. In this equation t_{i} should not equal 1.

After that, the space is divided into two regions, one to the right and the other to the left of the robot's direction of motion. The total weighted deflection is taken for each side separately as shown below.

Suppose that N_{L} and N_{R} are the number of obstacles inside D_{s} on the left and right sides, respectively. We can define the total weights for all obstacles on the left side as:

$$W_{\mathrm{L}} = \sum_{\mathrm{i}=1}^{N_{\mathrm{L}}} w_{\mathrm{i}} \tag{3.16}$$

The total deflections on the left side can now be defined as the weighted sum of all obstacle deflections on this side.

$$D_{\mathrm{L}} = \sum_{i=1}^{N_{\mathrm{L}}} \frac{w_{\mathrm{i}}}{W_{\mathrm{L}}} \delta_{\mathrm{i}} \in [-\pi, \pi[\tag{3.17}$$

The value of D_{L} is changed to adjust the difference in the number of obstacles inside D_{s}, between the two sides.

$$D_{\mathrm{L}} = D_{\mathrm{L}} / P_{\mathrm{L}} \tag{3.18}$$

where $P_{\mathrm{L}} = N_{\mathrm{L}} / N$. Notice that we set $P_{\mathrm{L}} = 1$ if either N_{L} or N equals zero.

We find $\mathrm{W_R}$ and D_{R} for the right hand side in the same manner as (Eqs. 3.16, 3.17 and 3.18).

Finally, the total net deflection D_{net} is calculated as follows:

$$D_{\text{net}} = \frac{W_{\text{L}}.D_{\text{L}} + W_{\text{R}}.D_{\text{R}}}{(W_{\text{L}} + W_{\text{R}})} \tag{3.19}$$

In order to achieve safety in navigation, the angular trajectory for the robot is the direction of motion θ_{md} modified by the net deflection D_{net}.

$$\theta_{\text{traj}} = \theta_{\text{md}} - D_{\text{net}}. \tag{3.20}$$

The speed of the robot is controlled according to the distance between the robot and the closest obstacle. Suppose that d_{min} is the distance between the closest obstacle and the robot boundary. The maximum speed of the robot should be limited as follows:

$$v_{\text{limit}} = \text{sqrt}\left(1 - \text{sat}_{[0,1]}\left(\frac{D_{\text{vs}} - d_{\text{min}}}{D_{\text{vs}}}\right)\right) . v_{\text{max}} \tag{3.21}$$

where v_{max} is the maximum linear speed of the robot and D_{vs} the velocity safe distance.

Remark 3.2 (comparison to SND): The safe distance used in Eq. (3.21) to limit the speed D_{vs} is different from the one used above to calculate the threats D_{s}. In this regard, we can increase D_{s} to be more reactive for dynamic obstacles without affecting the speed. In the SND, this is not possible for two reasons: enlarging D_{s} will increase the possibility of the deadlock mentioned above and will decrease the speed (may be too slow to move the robot) in cluttered environments. Furthermore, the nonlinear function ($sqrt$) is used to increase the speed of the robot as compared with the linear equation defined in the SND.

In order to calculate the linear and angular speeds, the same equations proposed in [15] and [16] are used.

$$v = \text{sat}[0,1]\left(\frac{\pi/_4 - |\theta_{\text{traj}}|}{\pi/_2}\right) . v_{\text{limit}} \tag{3.22}$$

$$w = \text{sat}[-1,1]\left(\frac{\theta_{\text{traj}}}{\pi/_2}\right) . w_{\text{max}} \tag{3.23}$$

where w_{max} is the maximum orientation (angular) speed of the robot.

3.3 Simulations

In order to explain the advantages of this work as compared with the SND method, we implemented the two approaches in the well known Player/Stage robot software system version 2.1.1 [83] using the same specifications for the laser rangefinder used in our experiments. This sensor scans 683 points over 240° with a maximum range of 5.6 *m*. We used a rectangular differential drive robot with a length of 0.53 *m* and a width of 0.49 *m* in order to imitate the real robot. The maximum linear and angular velocities were set to 0.5 *m*/*s* and 1.0 *rad*/*s* while the safe distance was set to 1.0 *m*.

We show, respectively, four different scenarios applied on various created maps in order to clarify the importance of the proposed approach. The first two simple scenarios show the advantages of the new method for analyzing gaps compared with the ND and SND. The other two scenarios demonstrate the power of this approach in avoiding the deadlock problem mentioned in subsection 3.2.4.

3.3.1 Simulations for Scenarios (1, 2)

This part explains how fetching gaps in the CG alleviates oscillations and achieves safe and smooth trajectory. Scenario (1) is shown in Figs. 3.4(a-d), where the task is to move the robot from the start to the goal locations marked in the map. Using the SND method, the robot reaches an oscillation behavior and stops at the point shown in Fig. 3.4a. This is due to fact that the navigable region closest to the goal is formed by discontinuity points *A* and *B* (the other regions occur at points: *D*, *E* and *F*, *G*). When the robot starts moving towards the rising gap of this region (*B*), another discontinuity appears on point *C* which adds a new region: *C*, *D* (see Fig. 3.4b). Now, this region will be the closest one to the goal which forces the robot to navigate towards its rising gap (*C*). The robot will alternate between these two states without reaching the goal.

The gaps detected using the proposed CG method are marked by arrows 1, 2 and 3 in Fig. 3.4c. In all the cases, the discontinuity point *E* forms gap 2 which is the closest one to the goal. This enforces the robot to navigate through it avoiding all obstacles till reaching the goal as shown in Fig. 3.4d.

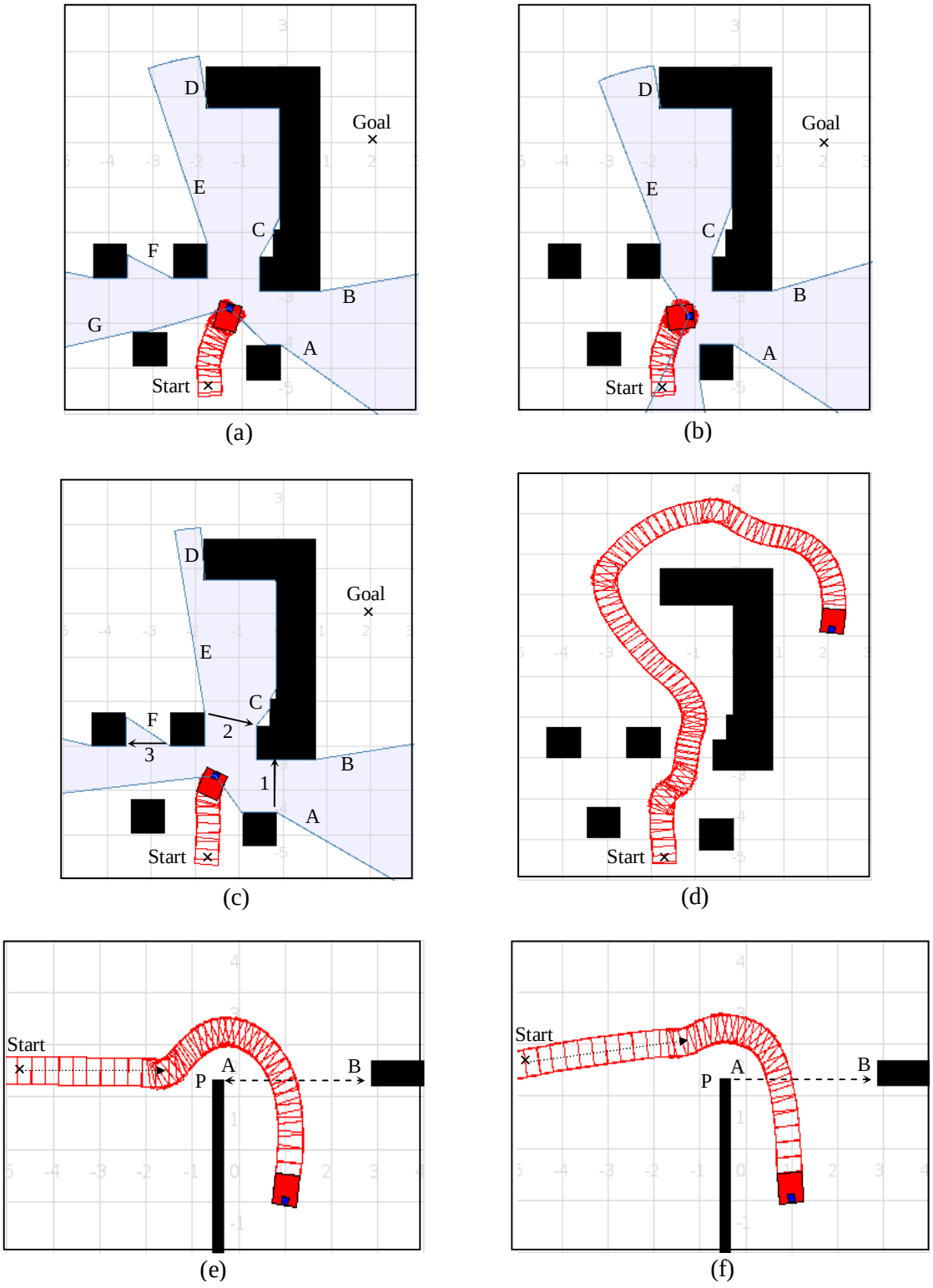


Figure 3.4: CG simulation results part 1. (a-b) Oscillation in behavior using the SND method. (c-d) No oscillation occurs in CG. (e) Trajectory followed with the SND algorithm, where the gap aperture with respect to the robot vision is small. (f) Trajectory followed by the CG method, where the gap aperture with respect to the robot vision is small.

The other scenario (2) demonstrates the advantage of adding the angle α to the motion direction θ_{md} as stated in subsection 3.2.3. This scenario implies moving the robot from its start position to the goal through the unique opening as shown in Figs. 3.4e,f. The region identified by the SND method (from point *A* to *B*) is too narrow as seen by the robot (Fig. 3.4e). The robot moves to the mid of the region which is nearly towards the obstacles identified by point *P*, and then it will be deflected away when it gets close to these obstacles. Adjusting θ_{md} to fit the robot diameter in the CG approach forces the robot to navigate far from the obstacles on point P as shown in Fig. 3.4f. Hence, a safer and smoother trajectory is achieved.

3.3.2 Simulations for Scenario (3, 4)

These two scenarios explain the deficiency of the SND method when there is a large difference in the number of threats on the two sides of the robot. The first one shows a situation where the robot should pass two narrow openings in order to reach its goal. By using the SND method, the robot collides with obstacles on the side (*S*1) of the first opening, coming to a full stop as shown in Fig. 3.5a. This is because side (*S*2) has more obstacles (a high deflection) than side (*S*1). The CG method overcomes this problem causing a safe and smooth navigation passing the two openings towards the specified goal (see Fig. 3.5b).

The route chosen for scenario (4) contains tight corridors, as explained in Figs. 3.5c, d, where the objective is to pass them safely and smoothly. The robot moved very close to obstacles on the corridors labeled *A*, *B* and *C* when we implemented the SND algorithm as shown in Fig. 3.5c. This refers to the same problem of the high deflection towards the side containing fewer obstacles mentioned previously. Adjusting the difference between threats on the two sides and providing stricter deflection from the nearest obstacles solve this drawback as stated in subsection 3.2.4. This can be noticed in the CG behavior shown in Fig. 3.4d where the robot navigates from the mid of the tight corridors. No actual differences in smoothness are noticed between the SND and the CG methods except on some locations where the CG did better. This can be sensed from the points labeled 1, 2 and 3 on their corresponding figures. We support that by plotting the angular velocity *w* against time for the SND and the CG methods as shown in Fig. 3.5e-1 and Fig. 3.5e-2. Moreover, the goal was reached in 140 *sec* using the SND method while it took 125 *sec* in the case of the CG.

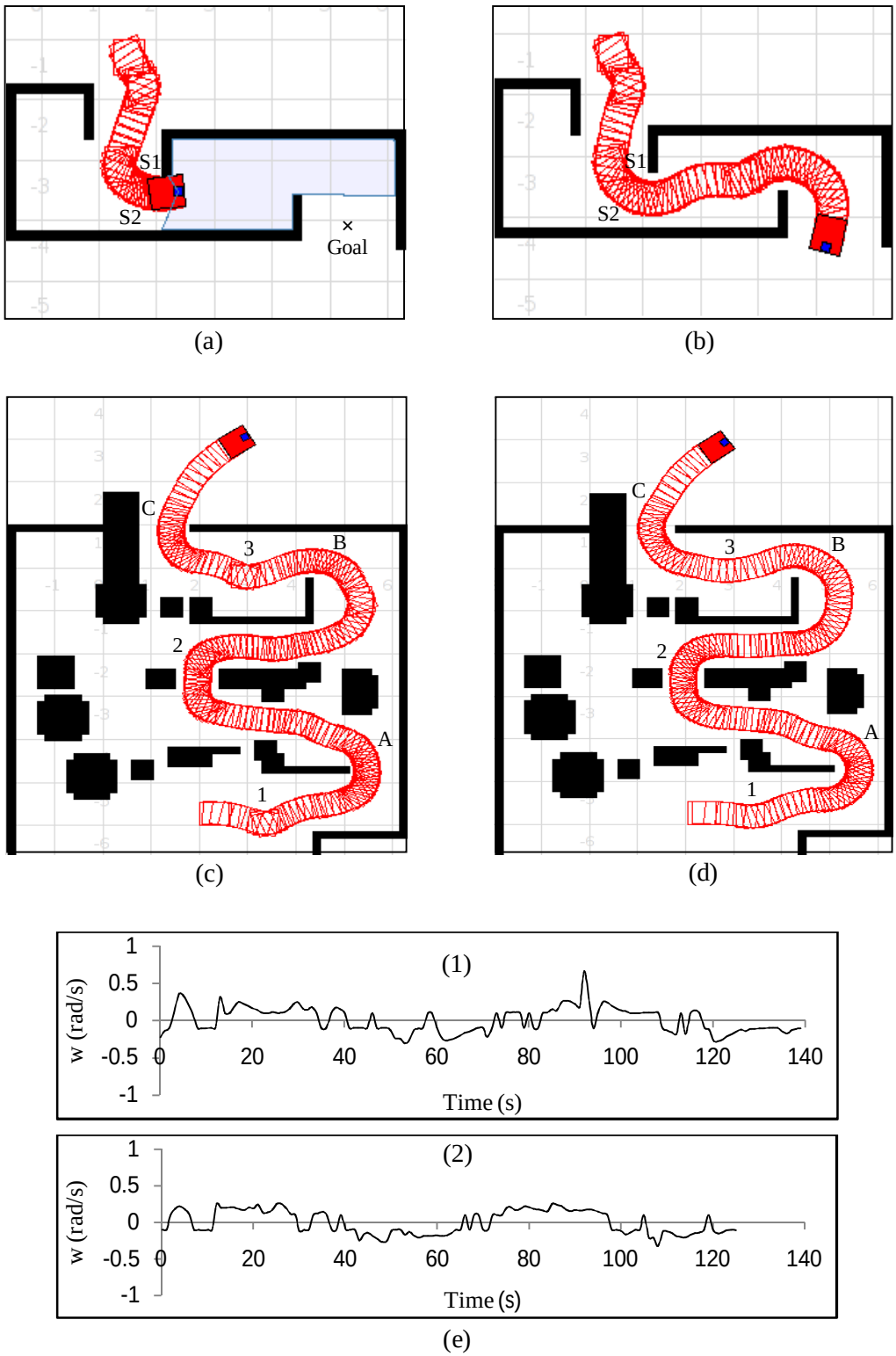


Figure 3.5: CG simulation results part 2. (a) Deadlock occurs in SND, with numerous threats on one side and fewer threats on the other. (b) Safe path with the CG method, with numerous threats on one side and fewer threats on the other. (c) Trajectory followed by the SND. (d) Trajectory followed with the CG. (e) Angular speed versus time.

3.4 Experimental Results

The simulated results have been confirmed using a real robot, a differential drive Pioneer 3-AT mobile robot equipped with a Hokuyo URG-04LX laser scanner and an on-board 2 GHz Pentium M computer. The robot platform is rectangular (0.53 × 0.49 *meters*) with non-holonomic constraints. The maximum translational velocity of the robot is 0.7 *m*/*s* and the maximum rotational one is 140 *deg*/*s*. In doing the experiments, these velocities were limited to (0.5 *m*/*s*, 1.0 *rad*/*s*). The experiments were carried out in collaboration with the Get Lap research group at the University of Paderborn, Germany.

We adapted the algorithm introduced in this thesis to accommodate the rectangular shape of the real robot (the reader can see Appendix B for more information about this adaptation). Furthermore, the inaccuracy of sensor readings is alleviated by using a median filter with radius equals to 5.

The non-holonomic constraints are not taken into consideration in this algorithm. Moreover, the experiments were carried out without moving (dynamic) obstacles.

Fig. 3.6a shows one of our experiments. The only information provided to the robot in advance was the goal location. The experiment was carried out using SND and CG. While travelling through the first openings, e.g. passage 1, the two methods behave fairly similar but differences become clearer when looking at passage 2. Using the SND algorithm, the robot moved close to side 1 of the opening (Fig. 3.6b) and came to a full stop while nearly touching the obstacle (Fig. 3.6c). Using our new CG method the robot safely moved through this passage and reached the goal (Fig. 3.6d,e). The recorded angular speeds are shown in Fig. 3.6f-1 and Fig. 3.6f-2.

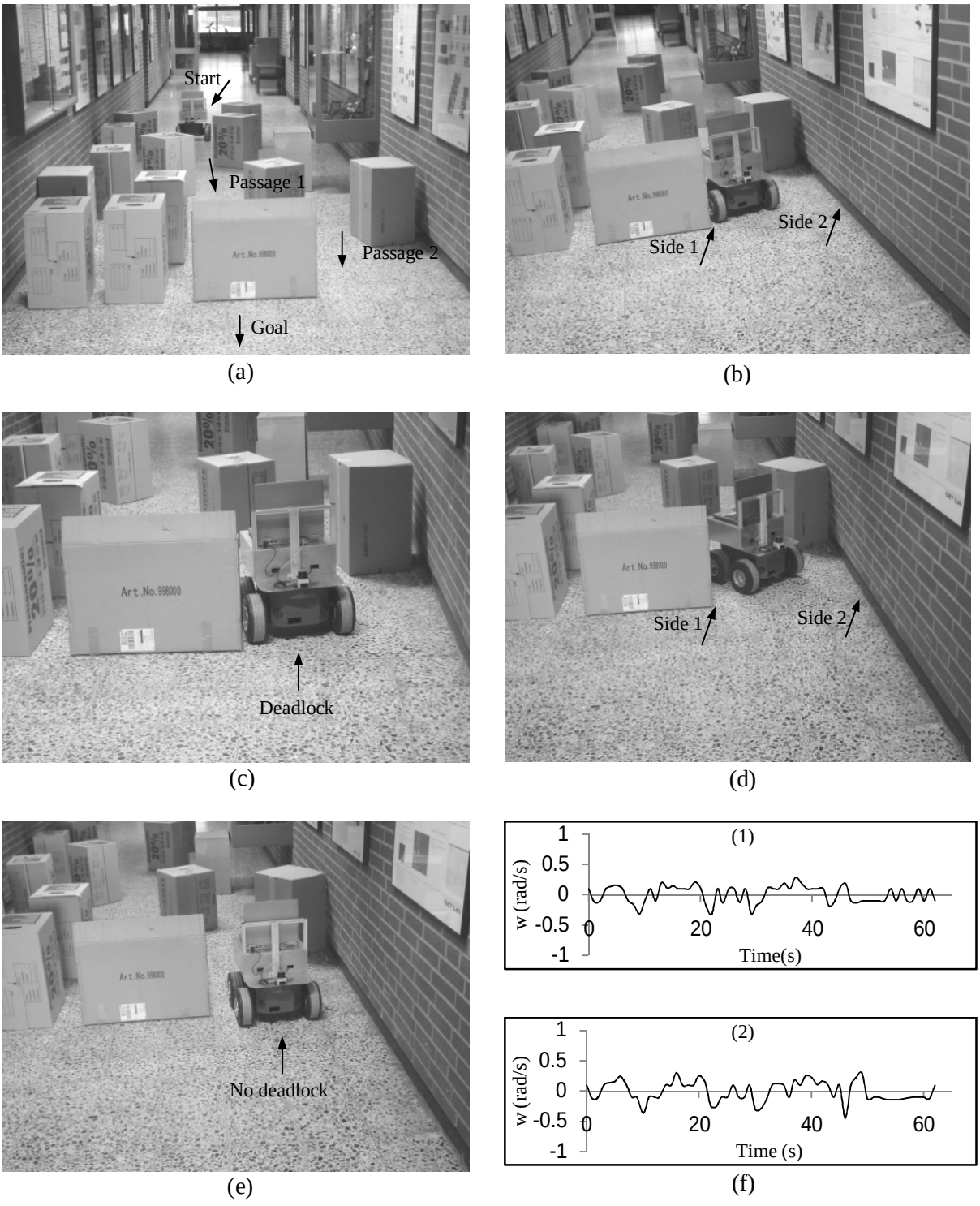


Figure 3.6: CG experimental results (a) Experimental setup. (b) The robot moves close to side 1 using the SND method. (c) A deadlock occurs with the SND method, with numerous threats on one side and fewer threats on the other. (d) The robot navigates safely using the CG method. (e) Passing a gap with the CG method, with numerous threats on one side and fewer threats on the other. (f) Angular speed versus time for the SND (1) and the CG methods (2).

3.5 Discussion

We present in this subsection a discussion showing the advantages of the CG method as compared with the SND approach on the basis of the drawbacks and limitations mentioned in section 3.1.

The **computational complexity** is reduced with this method. This is due to the fact that the CG only detects the needed gaps that are directly in front of the robot. If a certain gap leads to other ones, only this gap is considered. Therefore, it cancels useless gaps and as a result their number is highly reduced as compared with the SND method. Now, assume that the useless gaps are the closest ones to the goal and they are non-navigable. In this case, the algorithm will waste time in checking the navigability of these gaps. Fig. 3.2 depicts this situation, where the SND method detects 12 gaps (A-Z). The SND algorithm checks the navigability of the gaps from A to I before determining that gap I is the navigable one closest to the goal. The CG algorithm only detects gap 2 which leads to all of these useless gaps. Therefore, checking navigability is only done for this gap.

Increasing the number of gaps detected raises the possibility of **oscillations**. In addition, it increases the possibility of **trapping the robot** in a situation where the gap closest to the goal changes frequently between two gaps. Fig. 3.4(a-d) explains such situation. Therefore, SND is prone to such problems than CG.

With this method, considering the angular width of the closest gap with respect to the robot vision in selecting the motion direction (the best heading) gives a **smoother and safer behavior**. This appears clearly when the closest gap is very narrow (Fig. 3.3). The direction of motion in the SND is through the mid of the gap. There is no actual difference between the mid and the two sides of the gap. Therefore, the chosen direction guides the robot towards the obstacles on the side closest to the robot till they enter the security zone of the boundary of the robot, in which case, the robot is deflected away from these obstacles. It is obvious that this situation reduces the smoothness and safety of the robot paths. Using the CG, the direction of motion guides the robot far from the obstacles on the side closest to the robot by a distance equals to the robot diameter at least. This results in smoother and safer robot trajectories.

The CG method avoids **the deadlock problem** occurring in narrow corridors, with high threats on one side and low threats on the other. This is achieved by adjusting the difference

in the number of threats on the two sides of the robot and modifying the weight. Adjusting the number of threats will nearly equate the number of obstacles on the two sides of the robot, and as a result the deflection from the two sides will be nearly equal and the robot will move to the mid of the corridor. The function that calculates the weight (Eq. 3.15) maps the value of the weight from 1 to ∞ as the threat changes from 0 to 1. As the threat gets closer to 1, the output increases more sharply (Fig. 3.7a). This is required to ensure a stricter deflection from the closest obstacles. The range of the output will be only from 0 to 1 (Fig 3.7b), if the weight equals the square of the threat (as in SND). In this case, as the threat gets closer to 1, its square increases less sharply than the previous case which decreases safety (Fig. 3.5 and Fig. 3.6).

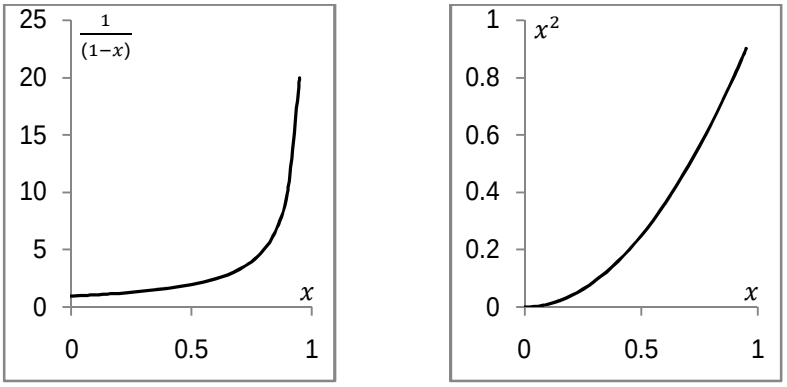


Figure 3.7: (a) Obstacle weight used in CG. (b) Obstacle weight used in SND.

3.6 Conclusions

We have addressed the Closest Gap Navigation (CG) method for local reactive collision avoidance. CG alleviates oscillations and computational complexity by designing a new scheme for fetching gaps which reduces their number and eliminates unnecessary ones. The robot vision of the opening angle of the gap is taken into account also in order to provide a smoother behavior. Moreover, it improves the safety of paths generated by the Smooth Nearness-Diagram (SND) method through considering the ratio of threats on the two sides of the robot and applying stricter deviation against an obstacle as it gets closer to the robot. As a consequence, a robust navigation in very dense and cluttered scenarios is achieved.

Chapter 4

Tangential Closest Gap Navigation

4.1 Introduction

In the previous chapter, a new obstacle avoidance method called Closest Gap (CG) was developed. This method can safely drive the robot in highly cluttered environments like the well known approaches designed to work in these environments (e.g. the Nearness-Diagram Navigation (ND) method [15]). Over the past years researchers have distinguished between these approaches that focus on their applicability for cluttered and complex environments and approaches that focus on fast reaction against obstacle detection (e.g. the Tangential Escape (TE) method [17]). The former have a slow behavior, particularly in avoiding dynamic obstacles, and they are likely to produce oscillatory robot trajectories. The latter cease to function in slightly complex scenarios and could be trapped in local minima.

In this chapter, a new real-time obstacle avoidance algorithm for complex environments is developed, entitled Tangential Closest Gap (TCG) method, where fast reaction against dynamic obstacles and a less oscillatory behavior are achieved. Therefore, the limitations of the well known techniques designed for dense environments, e.g. the Nearness Diagram Navigation methods [15, 16, 78], are avoided. The power of this approach lies in the integration of two concepts: The Closest Gap (CG) method (proposed in chapter 3) for fetching and analyzing openings surrounding the robot and the Tangential Escape (TE) approach [17] for reactive obstacle avoidance navigation.

The Tangential Closest Gap (TCG) navigation approach forces the robot to move towards the goal in the absence of obstacles along the direct path to the goal. If not, the trajectory is switched to the navigable gap closest to the goal by rotating the goal position temporarily till passing the gap. If the distance to an obstacle gets less than a predefined safe

distance during navigation, the position of the goal is switched again to another point that keeps the robot moving in parallel to the tangent of the closest obstacle. Motion commands are derived with proven stability for the whole control system ensuring that the robot reaches any reachable goal.

The chapter is organized as follows: A brief description of the Tangential Escape (TE) approach is introduced in Section 4.2, whereas Section 4.3 presents the reactive navigation method design. In section 4.4, dynamic motion enhancements for real robots are discussed. Simulation and experimental results are presented in Sections 4.5 and 4.6 showing the effectiveness and the advantages of the proposed approach over previous techniques. Finally, in Section 4.7, we draw some conclusion remarks.

4.2 The Tangential Escape (TE) Method

The Tangential Escape (TE) method, which is presented in [17] and implemented in [84], uses a very simple criterion to reach the goal; as soon as an obstacle enters the security zone of the boundary of the robot, the current robot orientation is changed forcing the robot to follow a path that is parallel to the tangent of this obstacle. This is done by changing the goal position temporarily till passing the risk. The process is repeated each time a new obstacle appears until the robot reaches its destination.

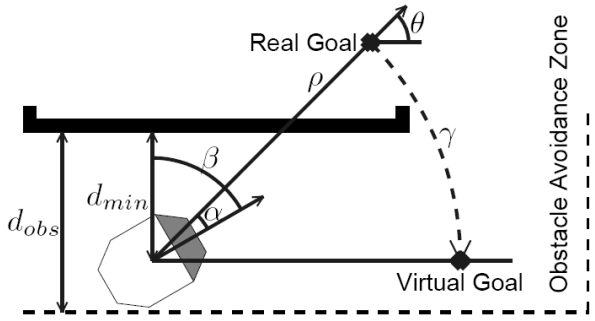


Figure 4.1: Obtaining the tangential deviation [17].

Motion commands are derived from the kinematic model of the robot where the stability of the system is proved through a Layabunov function. In the absence of obstacles, these motion commands drive the robot towards the goal. For an existence of obstacles within the security zone around the robot, the goal position is rotated by an angle γ (Fig. 4.1). This angle, named tangential deviation, is calculated as follows [17].

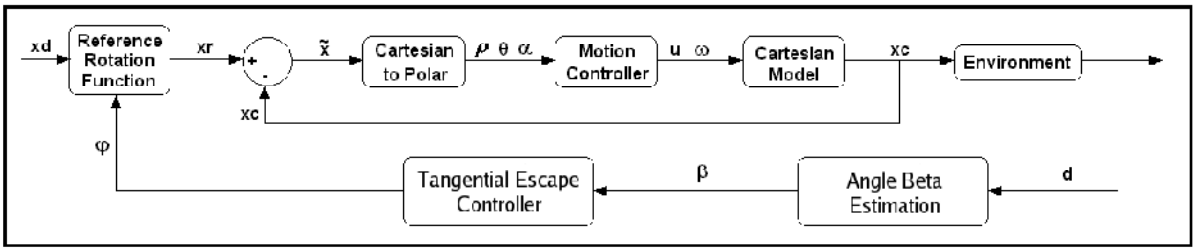


Figure 4.2: Block diagram of the control system based on the tangential escape approach [17].

$$\gamma = -(\frac{\pi}{2} - |\beta|)\text{Sign}(\beta) - \alpha \tag{4.1}$$

where β and α are the angle towards the obstacle point closest to the robot and the angle towards the goal relative to the current robot orientation, respectively.

This angle γ is used to create a rotation matrix that is applied to the coordinates of the real target, so that it is rotated to a new position (the virtual target) [84]. The full Tangential Escape control system is shown in Fig. 4.2. The new robot orientation will always consider the coordinates of the virtual goal to navigate in parallel to the tangent of the obstacles. Notice that in the absence of obstacles the tangential deviation γ is zero, the virtual goal is the real one, and the robot will seek the real goal.

This approach is very simple to implement, achieves fast reaction against dynamic obstacles and a smooth behavior. However, it is only suitable for very simple scenarios. Also, no planning before moving is done (only go and avoid). The robot can get stuck in many situations. Moreover, the robot follows the contour of obstacles without a leaving condition that resumes the progress towards the goal when the risk is passed. More details regarding these issues and how they are avoided in the TCG approach are explained in the next sections.

4.3 The Reactive Navigation Method Design

In this section, the Tangential Closest Gap Navigation (TCG) method for collision avoidance is presented, assuming a circular and differential drive non-holonomic mobile robot. The TCG technique works as follows: first, the sensory information is periodically analyzed to determine the structure of obstacles and find openings surrounding the robot as will be explained in subsection 4.3.1. Based on the current situation identified from these analyses, the robot heading angle is changed to move through the gap closest to the goal and/or avoid

nearby obstacles, through rotating the goal position temporarily till passing the risk as will be described in subsection 4.3.2. In subsection 4.3.3, we illustrate how the motion commands that drive the robot to a given goal, in a space free of obstacles, are derived. The details of calculating the rotation angles are the subjects of subsections 4.3.4 and 4.3.5.

4.3.1 Fetching and Analyzing Gaps

A gap can be defined as a potential free path wide enough for the robot to navigate through. In this subsection, we briefly summarize the algorithm used for analyzing gaps seen by a robot navigating in the world model which is proposed in chapter 3.

First, a forward search for gaps from scan 0 to $N-1$ is done (supposing that the final scan is N). The first side of a gap occurs at scan number i when two successive depth measurements are either have a positive difference (i.e. a rising discontinuity) more than the robot diameter $2R$ or the second measurement returns the maximum sensor range $d_{\max}$ while the first is less. In the first case, the second side of the gap will be at scan number $j > i + 1$ which satisfies the shortest distance to the first side. The angle δ between the first and second side of the gap must be less than π. In the other case, the second side occurs at scan number $j > i + 1$ if a descending discontinuity is fetched or the first scan of two consecutive scans returns $d_{\max}$ while the second is less. The forward search is then resumed from scan $j + 1$ in order to get other gaps if exist.

Second, another search for gaps is done but backward this time from scan $N-1$ to scan 0. The second side of a gap occurs at scan number j when either a descending discontinuity is fetched or the first scan of two consecutive scans returns $d_{\max}$ while the second is less. In the first case, the first side of the gap will be at scan number $i < j-1$ which satisfies the shortest distance to the second side. Again the angle δ between the first and second side of the gap must be less than π. In the other case, the first side occurs at scan number $i < j-1$, whenever a rising discontinuity is detected. The backward search is resumed from scan $i-1$ in order to get other gaps if exist.

Finally, unnecessary gaps are omitted as follows. Every gap that is contained inside another gap is deleted. Then, the gap that has a width less than the robot diameter is also removed. The navigable gap closest to the goal is then chosen from among the remaining

gaps. This gap has two sides; it is referred to the closer one to the goal as θ_{cs} and the other as θ_{os}.

4.3.2 Situations and Associated Actions

Navigating in a space full of obstacles requires changing the current robot heading angle to avoid collisions, through rotating the goal position to another point temporarily till passing the risk. Rotating the goal location is dependent on two criteria:

Criterion 1: Way to goal criterion. There are two situations concerning this criterion, depending on whether the way from the robot to the goal is free from obstacles (Free-Way) or not (Dangerous-Way). For an existence of Free-Way (see appendix A for description of the algorithm responsible for checking this situation), no change to the real goal position is done (Fig. 4.3a). Else, the goal location is switched to another point temporarily through rotating it by an angle φ_{sg} relative to the robot heading direction. We refer to the new position as a *sub-goal*, which drives the robot toward the navigable gap closest to the real goal (Fig. 4.3b). Calculating the rotation angle φ_{sg} depends on the angles θ_{cs} and θ_{os} of the closest gap as will be explained in subsection 4.3.4.

Criterion 2: Safety criterion. The existence of obstacles within a predefined safe distance D_s around the robot (Low-Safety) or not (High-Safety) determines the two situations of this criterion, which are checked after checking the situations of criterion 1. If the area inside D_s is free from obstacles (High-Safety), then no change to the actions associated with criterion 1 situations explained above (Figs. 4.3a,b). Else, the goal/sub-goal position is switched to another point temporarily through rotating it by an angle φ_{vg} relative to the robot heading direction. We refer to the new position as a *virtual-goal*, which keeps the robot navigating in parallel to the tangent of the closest obstacle (Fig. 4.3c). The rotation angle φ_{vg} is dependent on the angle β towards the nearest obstacle to the robot as will be described in subsection 4.3.5.

With this, we can now define the net rotation angle φ as the sum of the two angles:

$$\varphi = \varphi_{sg} + \varphi_{vg} \tag{4.2}$$

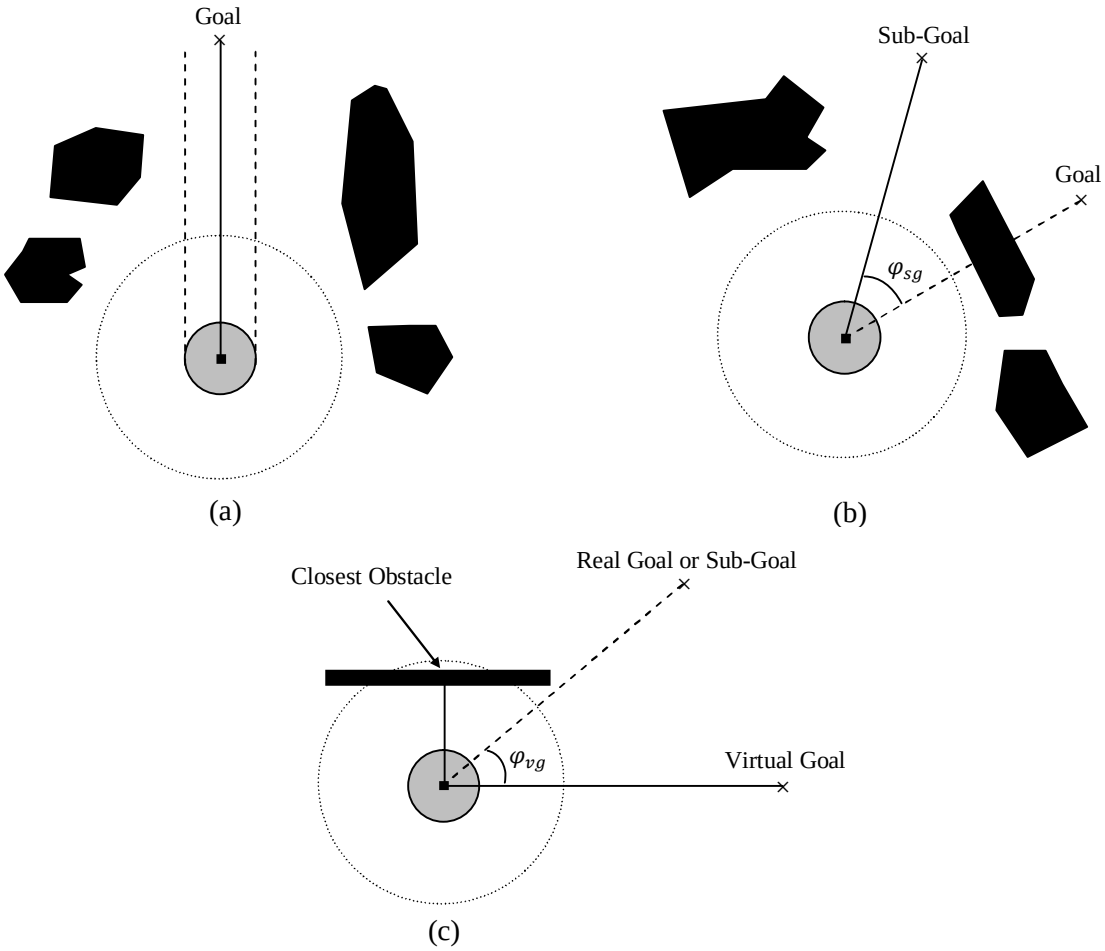


Figure 4.3: Situations and their actions. (a) Example showing Free-Way and High-Safety situations. (b) Dangerous-Way and High-Safety situations, so the goal position is rotated towards the gap closest to the goal (sub-goal). (c) Low-Safety situation, so the goal/sub-goal position is switched temporarily to a virtual goal. This forces the robot to navigate in parallel to the tangent of the closest obstacle.

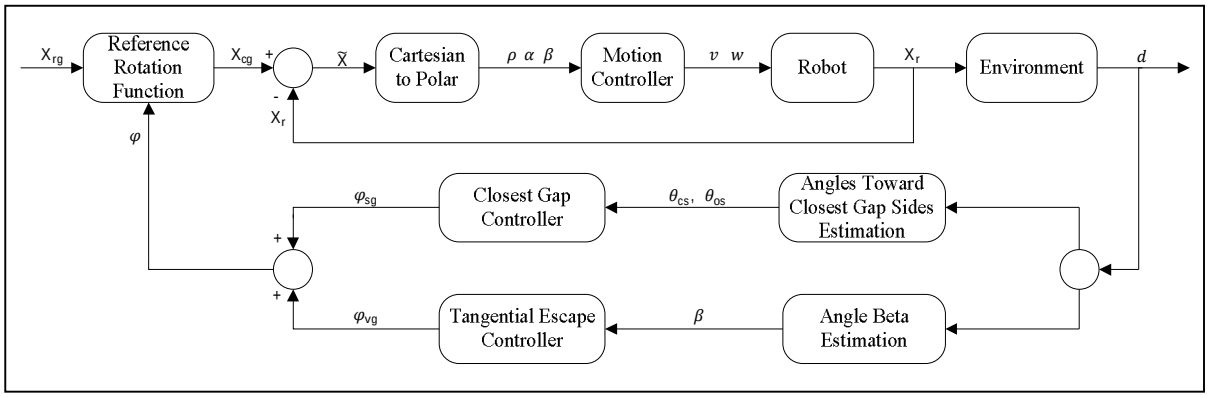


Figure 4.4: The block diagram of the closed control loop for the TCG approach.

The Dangerous-Way situation results in a rotation angle φ_{sg}. Otherwise (Free-Way), we set $\varphi_{sg} = 0$. Also, the Low-Safety situation results in a rotation angle φ_{vg}. Else (High-Safety), we let $\varphi_{vg} = 0$.

The block diagram for the control system responsible for guiding the robot to reach the target following this approach is shown in Fig. 4.4. One can notice that the inner loop of this control system, which is used to guide the robot to the goal with Free-Way and High-Safety situations, is the same as the one introduced in [17] (Fig. 4.2). The outer loop differs from the one developed in [17] by adding the angle φ_{sg} responsible for switching the goal position to the sub-goal to move through the gap closest to the goal (Dangerous-Way situation) as mentioned above. We refer to the position of the robot and the real goal as X_r and X_{rg}. Each sensor update, the current goal position X_{cg} is calculated through rotating the real goal by the rotation angle φ.

$$X_{cg} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} X_{rg} \tag{4.3}$$

If φ is positive, the goal position is rotated clockwise relative to the robot heading angle. Otherwise, it will rotate counterclockwise.

4.3.3 Determining Motion Commands

In this subsection, the motion commands (velocities) derived from the kinematic model of the robot is presented. These velocities ensure that a robot moving in a space reaches any reachable goal in the absence of obstacles. This can be achieved through choosing velocities that make the system stable in the Lyapunov sense. We use the motion commands defined in [17] after adapting them to be suitable for this approach as shown below.

The kinematic model for a differential drive mobile robot considering two actuator wheels and non-holonomic constraints can be found in many references in the literature (e.g. [85]). Fig. 4.5 describes a robot location in the world model showing all necessary parameters. The mathematical model for this robot in the inertial frame is [85]:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4.4}$$

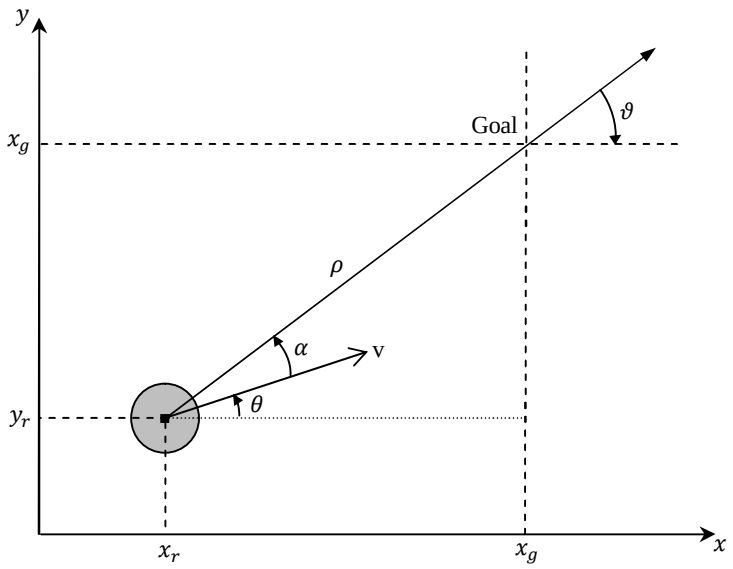


Figure 4.5: Position of a mobile robot in a plane.

where (x_r, y_r) are the coordinates of the current robot position, θ the angle between the robot heading direction and the horizontal axis, v the linear velocity and w the angular velocity. (x_r, y_r) and θ are taken from the robot odometry.

The position error can be written in polar coordinates as:

$$\rho = \sqrt{\Delta x_r^2 + \Delta y_r^2} \tag{4.5}$$

$$\alpha = -\theta + \text{atan2}(\Delta y_r, \Delta x_r) \tag{4.6}$$

$$\vartheta = -\theta - \alpha \tag{4.7}$$

where ρ is the distance between the center of the robot and the goal, α the angle between the robot heading and the goal (angular error) and ϑ the angle between the goal and the horizontal axis with respect to the current robot location.

The above equations can be rephrased on the following matrix:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\vartheta} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & 0 \\ \frac{\sin(\alpha)}{\rho} & -1 \\ -\frac{\sin(\alpha)}{\rho} & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4.8}$$

Rewriting the above matrix by the following three equations, we get [85, 17]:

$$\dot{\rho} = -v\cos(\alpha) \tag{4.9}$$

$$\dot{\alpha} = -w + v\frac{\sin(\alpha)}{\rho} \tag{4.10}$$

$$\dot{\vartheta} = -v\frac{\sin(\alpha)}{\rho} \tag{4.11}$$

It is clear that the value of ρ should not equal zero. Otherwise, the values of $\dot{\alpha}$ and $\dot{\vartheta}$ will be indefinite. As a result, we assume that the goal is reached when ρ is less than ε, where ε is a small value.

The objective is to reach the goal without oscillations. Hence, motion commands should be chosen such that the state variables ρ and α go asymptotically to zero. We consider the Lyapunov function candidate proposed in [17].

$$V = \frac{1}{2}\rho^2 + \frac{1}{2}\alpha^2 \tag{4.12}$$

In order to ensure stability, the derivative of the above Lyapunov function should be negative definite. This can be achieved if the motion commands (v, w) are defined as:

$$v = k_{\mathrm{b}} v_{\mathrm{limit}}\cos(\alpha) \tag{4.13}$$

$$w = \mathrm{sat}_{[-w_{\max}, w_{\max}]}\left(k_{\mathrm{w}}\alpha + \frac{v\sin(\alpha)}{\rho}\right), \quad k_{\mathrm{w}} > 0 \tag{4.14}$$

The above motion commands differ from the ones proposed in [17] in the following. The variable k_b is used to ensure smooth breaking at the final step (reaching the real goal). We put $k_b = \tanh(\rho)$ if Free-Way and High-Safety are the current situations. Else, it is set to 1. In the TE approach, k_b is always set to $\tanh(\rho)$ since it always seeks for the real goal if no obstacles exist in the vicinity of the robot (do not seek for sub-goals as in the TCG). The *sat* function, which does not exist in the TE method, is used to cap the value of the angular velocity between the maximum negative and positive values obtained from the data sheet of the robot (w gets greater than $w_{\max}$ when $\rho < 1$). Moreover, the speed of the robot must be controlled to avoid collisions, particularly in cluttered environments. Therefore, the v_{limit} variable in Eq. (4.13) is used which is defined in the CG approach, proposed in the previous chapter, as:

$$v_{\mathrm{limit}} = \mathrm{sqrt}\left(1 - \mathrm{sat}_{[0,1]}\left(\frac{D_{\mathrm{vs}} - d_{\min}}{D_{\mathrm{vs}}}\right)\right) . v_{\max} \tag{4.15}$$

where $v_{\max}$ is the maximum linear speed of the robot, D_{vs} the velocity safe distance and $d_{\min}$ the distance towards the obstacle closest to the boundary of the robot. Increasing D_{vs} decreases the speed.

One can notice from Eq. (4.13) that the linear velocity v is negative if the angle towards the goal α gets greater than $-\pi/2$ or less than $\pi/2$, forcing the robot to move backwards. However, for mobile robots that are restricted to move only forward, as the real robot used in our experiments[7], the following is done: If the absolute value of α is greater than $\pi/2$, we set the value of v to zero and force the robot to rotate by setting $w = w_{\max}$ until the absolute value of α gets less than $\pi/2$. If so, we resume using the above motion commands.

Regarding the constant k_{w}, the TE approach considers that the maximum value of w occurs when $\alpha = \pi/4$ and then performs calculations to find the value of k_{w}. There is no real justification for this assumption. As α increases, the value of w is required to increase. The maximum possible value for the angle α is $\pi/2$. Therefore, considering that the maximum value of w occurs when $\alpha = \pi/2$ will be more reasonable. With this we can substitute in Eq. (4.14) and find that $k_{\text{w}} = w_{\max}/(\pi/2)$.

Remark 4.1 (system stability): The Lyapunov function introduced in this subsection proves that the state variables ρ and α are asymptotically go to zero and the system is stable in the Layapunov sense. In this regard, the robot reaches any reachable goal in case of Free-way to goal and High-Safety situations (Fig. 4.3a). If the goal position is switched temporarily to X_{cg} as a result of *Dangerous-Way* to goal (Fig. 4.3b) and/or *Low-Safety* (Fig. 4.3c), the control variables v and w will guide the robot to the new goal position X_{cg} keeping the stability of the system. This process will be repeated till passing all gaps and circumnavigating all obstacles. As a consequence, the overall navigation process towards the goal is stable and the goal will be reached (supposed to be reachable).

4.3.4 Calculating Sub-Goal Rotation Angle

The robot should change the current orientation to navigate through the gap closest to the goal, whenever a situation of Dangerous-Way to a goal occurs. This can be achieved by

[7] The real robot is equipped with a laser scanner covering 240 degrees. The rear of the robot is not seen in this case. Therefore, we cancelled moving backwards.

rotating the goal position by an angle φ_{sg} as stated in subsection 4.3.2. Setting the rotation angle depends on two parameters: the location of the goal and the width of the gap. If the angle towards the goal falls within the closest gap, then no need to rotate the real goal and the angle of rotation φ_{sg} is set to 0. Else, the sub-goal rotation angle is set where a safe navigation towards the gap should be assured dependent on its width. The direction of safe motion towards the gap is stated in chapter 3 (Eq. 3.8) as:

$$\theta_{md} = \begin{cases} \theta_{mid}, & \text{if } \mathrm{dist}(\theta_{cs}, \theta_{mid}) < \mathrm{dist}(\theta_{cs}, \theta_{scs}), \\ \theta_{scs}, & \text{otherwise.} \end{cases} \quad (4.16)$$

where θ_{mid} is the angle of motion towards the mid of the gap if it is narrow, θ_{scs} is the angle of safe motion towards the gap if it is wide and $\mathrm{dist}(x, y)$ returns the angular width between the two angles x and y. Chapter 3 describes the calculations of the angles θ_{mid} and θ_{scs}, and modifying θ_{md} dependent on the angular width of the gap as seen by the robot sensors.

After getting θ_{md}, the rotation angle to switch the real goal to a sub-goal temporarily till passing the gap is:

$$\varphi_{sg} = \alpha - \theta_{md}. \quad (4.17)$$

where α is the angle towards the real goal relative to the robot heading direction.

4.3.5 Calculating Virtual Goal Rotation Angle

Whenever a situation of Low-Safety occurs during navigation, the current goal (real or sub goal) is rotated by an angle φ_{vg} forcing the robot to navigate in parallel to the tangent of the closest obstacle as stated in subsection 4.3.2. This subsection describes how to calculate the angle of this rotation.

Note: When we mention the word *goal* in the remaining of this section, we mean either the real goal or the sub-goal based on checking the situations of the way to goal criterion.

The Tangential Escape approach [17] proposed an angle γ (Eq. 4.1) that forces the robot to move in parallel to the tangent of the closest obstacle. This angle does not work properly if the range of the laser scanner is more than 180 degrees (from $-\pi/2$ to $\pi/2$). Moreover, when obstacles are detected inside the safe distance, the robot keeps moving following the contour of these obstacles without a leaving condition. This condition is necessary to resume the

progress towards the goal, when the risk is passed. Because the range of the laser scanner covers only half a circle and the trail of the robot is not seen, the robot leaves the contour of obstacles at the edges (–π/2 or π/2) with a considerable abruptly transition. Our approach solves this limitation through improving the rotation angle defined in [17] to be suitable for all the cases. The leaving condition is set also, with the help of dividing to sub-goals.

Let α be the angle towards the goal and β the angle towards the obstacle closest to the robot. The algorithm for calculating the proposed angle is described below:

If $(\text{sign}(\alpha) \; ! = \text{sign}(\beta))$

$$\varphi_{\text{vg}} = \begin{cases} \text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha), & \text{if } |\beta - \alpha| < \pi, \\ -\text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha), & \text{if } |\beta - \alpha| > \pi, \end{cases}$$

Else

$$\varphi_{\text{vg}} = \begin{cases} \text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha), & \text{if } |\beta| > |\alpha|, \\ -\text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha), & \text{if } |\beta| < |\alpha|, \end{cases}$$

where the function $\text{sign}(x)$ returns the sign of the angle x. Positive and negative angles are to the left and right sides of the robot, respectively.

As explained in the above algorithm, setting the rotation angle φ_{vg} depends mainly on the sign of the angle towards the goal α and the closest obstacle β relative to the robot (if they are on the same side or in different sides).

1) For different signs: The angle depends on the path length from α to β travelling clockwise if α is positive and counterclockwise if it is negative.

   (a) For a path length less than π: We set the rotation angle that assures moving in parallel to the tangent of the nearest obstacle, In this case, the direction of motion should be towards the positive x axis relative to the robot (region labeled *F* in Fig. 4.6) which can be achieved by choosing the following angle:

$$\varphi_{vg} = \text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha) \tag{4.18}$$

   Figs. 4.6a,b show the virtual goal caused from rotating the goal position by this angle.

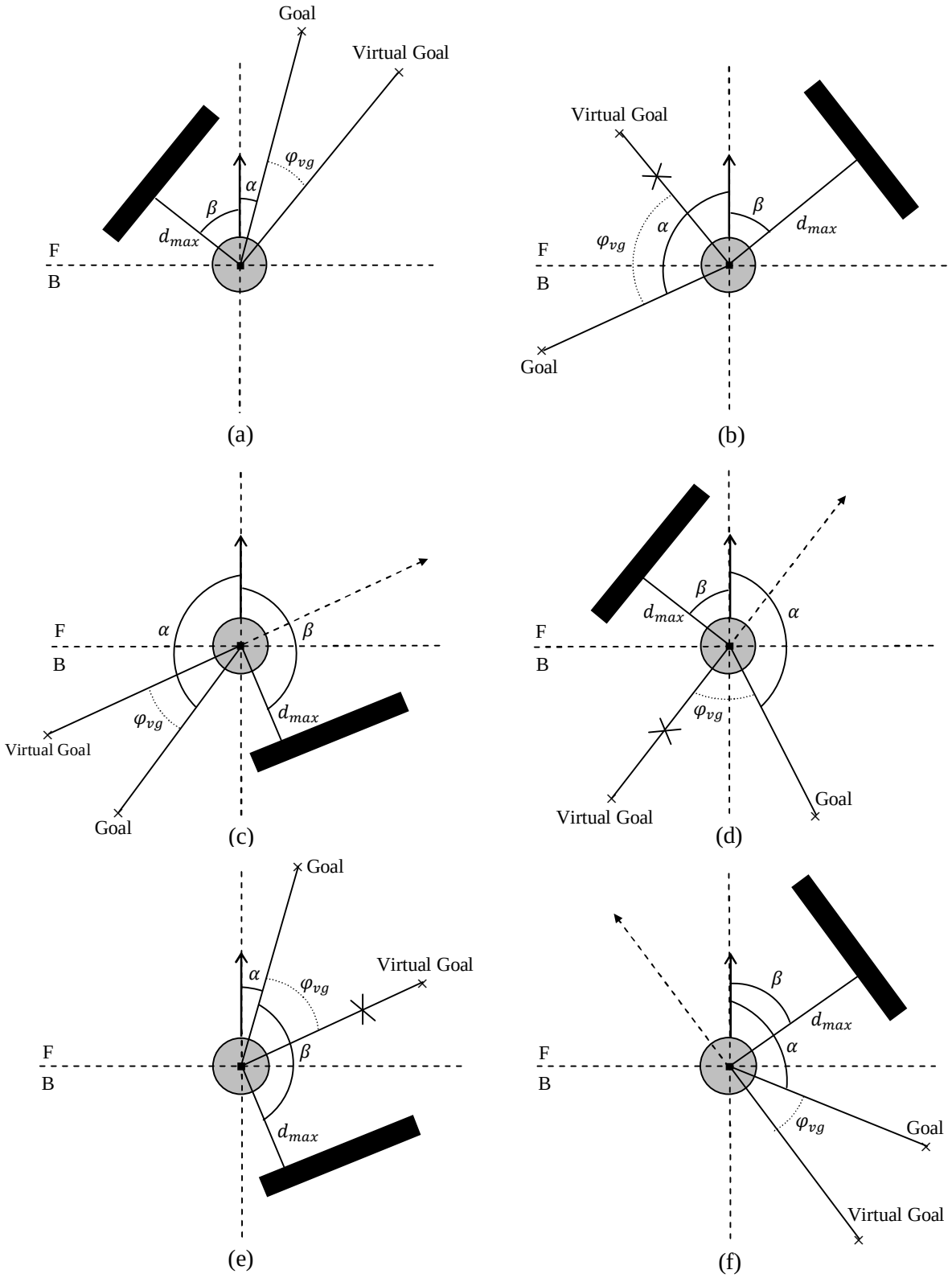


Figure 4.6: Different locations for the goal α and the closest obstacle β relative to the current robot position. (a - d): $\text{sign}(\alpha) \; != \; \text{sign}(\beta)$ where $|\beta - \alpha| < \pi$ for (a) and (b), $|\beta - \alpha| > \pi$ for (c) and (d). (e, f): $\text{sign}(\alpha) = \text{sign}(\beta)$ where $|\beta| > |\alpha|$ for (e) and $|\beta| < |\alpha|$ for (f). The leaving condition occurs in (b, d and e) where $\text{dist}(\alpha, \beta) > \frac{\pi}{2}$.

(b) For a path length more than π: The direction of motion should be towards the negative x axis relative to the robot (region labeled *B* in Fig. 4.6), in parallel to the tangent of the closest obstacle. This is to assure that the robot moves to the direction close to the goal. The angle in this case will be:

$$\varphi_{vg} = -\text{sign}(\beta)\frac{\pi}{2} - (\beta - \alpha) \tag{4.19}$$

The virtual goal caused from rotating the goal position by this angle is shown in Figs. 4.6c,d. If we use Eq. (4.18) for calculating the angle, the robot will move in the direction explained by the dashed arrows in Figs 4.6c,d. Therefore, it will depart away from the goal.

2) For equal signs: Deriving the angle in this situation is dependent on the absolute value of the angle towards the goal α, if it is greater or less than the absolute value of the angle towards the closest obstacle β, with respect to the current robot location.

(a) If $|\alpha|$ is less than $|\beta|$: The angle will be the same as in Eq. (4.18). Fig. 4.6e shows the virtual goal after rotating the goal position by this angle, which forces the robot to move towards the direction close to the goal in parallel to the tangent of the closest obstacle.

(b) If $|\alpha|$ is greater than $|\beta|$: We have the same situation as case 1 (b). The direction of motion should be towards the region labeled B, which ensures movement in the right way closer to the goal. The rotation angle is the same as in Eq. (4.19). Fig. 4.6f shows the virtual goal after rotating the goal position with the angle specified.

With this we can now define the leaving condition which forces the robot to resume the progress towards the goal (real or sub goal). Before that, we address the following definition:

Definition: let θ_1 and θ_2 be any two angles relative to the robot heading direction, we define the angular distance between θ_1 and θ_2 as follows.

$$\text{dist}(\theta_1, \theta_2) = \begin{cases} 2\pi + (\theta_1 - \theta_2), & \text{if}(\theta_1 - \theta_2) < -\pi, \\ 2\pi - (\theta_1 - \theta_2), & \text{if}(\theta_1 - \theta_2) > \pi, \\ |\theta_1 - \theta_2|, & \text{otherwise}, \end{cases} \tag{4.20}$$

The leaving condition occurs when the angular distance between the angle towards the goal and the closest obstacle exceeds 90 degrees.

$$\text{dist}(\alpha, \beta) > \frac{\pi}{2} \tag{4.21}$$

If the leaving condition is satisfied, we set the rotation angle φ_{vg} = 0. Back to Figs. 4.6(b, d and e), one can notice that moving directly towards the goal is more reactive and pushes the robot away from obstacles than navigating in parallel to the tangent of the closest obstacle. Moreover, the robot resumes moving towards the goal when the risk is passed instead of keep following the contour of obstacles. So, we reset the rotation angle calculated above for these situations to zero and, as a result, the virtual goal is canceled as explained in Figs. 4.6(b, d and e).

Remark 4.2 (comparison to earlier works): Avoiding nearby obstacles in ND, SND and even CG navigation methods is done through deflecting the robot heading direction by an angle that is a factor of 180 degrees. This factor depends on the proximity of the obstacle from the robot. In the TCG method, the robot changes the heading angle to navigate in parallel to the tangent of the closest obstacle when an obstacle is detected close to the robot. The behavior will be much faster and decreases oscillations. Increasing the safe distance in SND and CG will affect smoothness and speed whereas in TCG no change will occur. Hence, we can increase the safe distance in the TCG to achieve faster reaction against dynamic obstacles. Moreover, the problem of the deadlock mentioned in chapter 3, which is alleviated in the CG approach, does not appear in our technique at all.

4.4 Motion Enhancements for Real Robots

Sometimes, undesirable motion behavior can appear in the real implementation, particularly when moving at high speeds. This refers to many reasons such as: the uncertainty caused from sensors, the computational power and the dynamic properties of the mobile robot. The TCG depends a lot on measuring the angle towards the obstacle closest to the robot β and the angle towards the goal α. Assuming that the laser scanner returns the precise angles and that the robot responds instantaneously to situations (ideal case) may cause unwanted behavior

(e.g. oscillatory behavior). This section presents the improvements that have been integrated in the TCG in order to overcome these limitations.

4.4.1 Determining the Angle towards the Closest Obstacle

In implementing the TCG approach, considering only one laser scan point to estimate β, which is the angle towards the closest obstacle, causes oscillations in behavior. This is due to the fact that the closest obstacle changes frequently (between closely separated ones) while the robot is moving. The problem becomes more obvious if the shape of the robot is rectangular as the one used in our experiments. To overcome this problem, we propose the following algorithm for calculating the angle β.

The *inputs* of the algorithm are:

1) The robot location (X_{robot}).
2) A list (L) of obstacle points, where an obstacle point is (O_i^L) and the angle towards it is (θ_i^L).

The *output* of the algorithm is the angle β.

First, we find the distance to the obstacle point closest to the robot. Then, this distance is compared with the distance to each obstacle point in the list L. Any one that has a very small difference is chosen. From among the chosen obstacle points, we take only the ones that lie on the side of the robot where the closest obstacle exists (assuming that the robot has two sides, one is to the left and the other to the right of the robot heading direction). The angle β will be the average of the angles toward these obstacle points. In other words, the algorithm can be stated as follows.

Consider the list (L) of the obstacle points. The following is done:

1) Find the obstacle point O_c^L from L where:

   $d(O_c^L, X_{robot}) \leq d(O_i^L, X_{robot})$, for any point $i \in L$.

   where $d(x, y)$ returns the distance between the two points x and y.

2) Find the list $N \in L$ of obstacle points where:

a) $\text{sign}(\theta_i^N) = \text{sign}(\theta_c^L)$,

b) $\left[d(O_i^N, X_{robot}) - d(O_c^L, X_{robot})\right] < \varepsilon$,

where ε is a small value. We used a value of 5 cm in the real implementation.

3) The angle β is now defined as:

$$\beta = \sum_{i=1}^{N} \frac{\theta_i^N}{N}$$

4.4.2 Adding a Digital Low-Pass Filter

Satisfying the leaving condition in the TCG switches the direction of motion from navigating in parallel to the tangent of the closest obstacle to moving towards the goal. In some cases, this results in a drastic change in the steering direction (the direction towards the current goal[8] α) which reduces smoothness. To overcome this problem, we add a simple filter defined as follows:

$$\beta_i = s\alpha_i + (1 - s)\beta_{i-1} \tag{4.22}$$

where β_i is the resultant steering direction, α_i the steering direction before filtering, β_{i-1} the previous output of the filter, and s the smoothing factor which is defined as follows:

$$s = \frac{T}{\tau + T} \tag{4.23}$$

where T is the sampling interval and τ a time constant.

The laser scanner used in the experiments has an update rate of 10 Hz. Therefore, the sampling interval T was set to 0.1 sec, Choosing a high value for the time constant causes a considerable delay in response. However, doing various tests showed that choosing $s = 0.5$ gives us a well-balanced behavior (i.e. $\tau = T$).

[8] We mean by the current goal: the real goal, sub-goal or virtual goal depending on checking the situations mentioned in subsection 4.3.2.

4.5 Simulations

This section shows the differences in the execution of TE, SND, CG and TCG navigation methods. They were implemented in the open-source Player/Stage robot software system version 2.1.1. We used a rectangular differential drive robot with a length of 0.53 m and a width of 0.49 m. The maximum translational speed was set to $v_{\max} = 0.5\ \text{m/s}$, the maximum rotational one to $w_{\max} = 1.0\ \text{rad/s}$ and the safe distance to $D_s = 1\ \text{m}$. The simulated laser rangefinder scans $n = 683$ points over 240° with a maximum range of $d_{\max} = 5.6\ \text{m}$ (for the TE approach, we limit the range to cover only 180° in order to work properly). We set the velocity safe distance to $D_{vs} = 1\ \text{m}$ in the CG and TCG methods.

For these simulations we created a map with narrow corridors where the robot should do sharp turns passing all gaps till reaching the goal. Fig. 4.7 shows this map where the start and goal locations are identified also. The progress of the robot is shown by square bounding boxes left behind on the map during navigation. The density of these boxes on a specific location determines the relative speed for the robot at this position.

4.5.1 TE Simulation

The trajectory followed using the TE method is shown in Fig. 4.7a. The robot navigates in parallel to the tangent of the closest obstacle without any planning (only go and avoid because openings are not analyzed in this approach). When the robot gets close from the obstacles on side labeled *A*, it moves parallel to their tangent following the contour of these obstacles. The robot should get closer to the obstacles on side *B* before changing its direction to follow their contour. As a result, the robot gets very close to these obstacles as shown in the figure. After leaving the obstacles on point *C*, the closest obstacle switches between the right (point *D*) and left (point *F*) sides of the robot causing sharp changes for the trajectory of the robot, and finally it comes to a full stop after 75 sec when colliding obstacles on the side labeled *E*. But if the robot sees the gap labeled *Gap* on the figure as a step towards the goal, it will leave following the contour of obstacles and directly move towards this gap (sub-goal) whenever the leaving condition is satisfied. The recorded motion commands (v and w) versus time are explained in Fig. 4.8a.

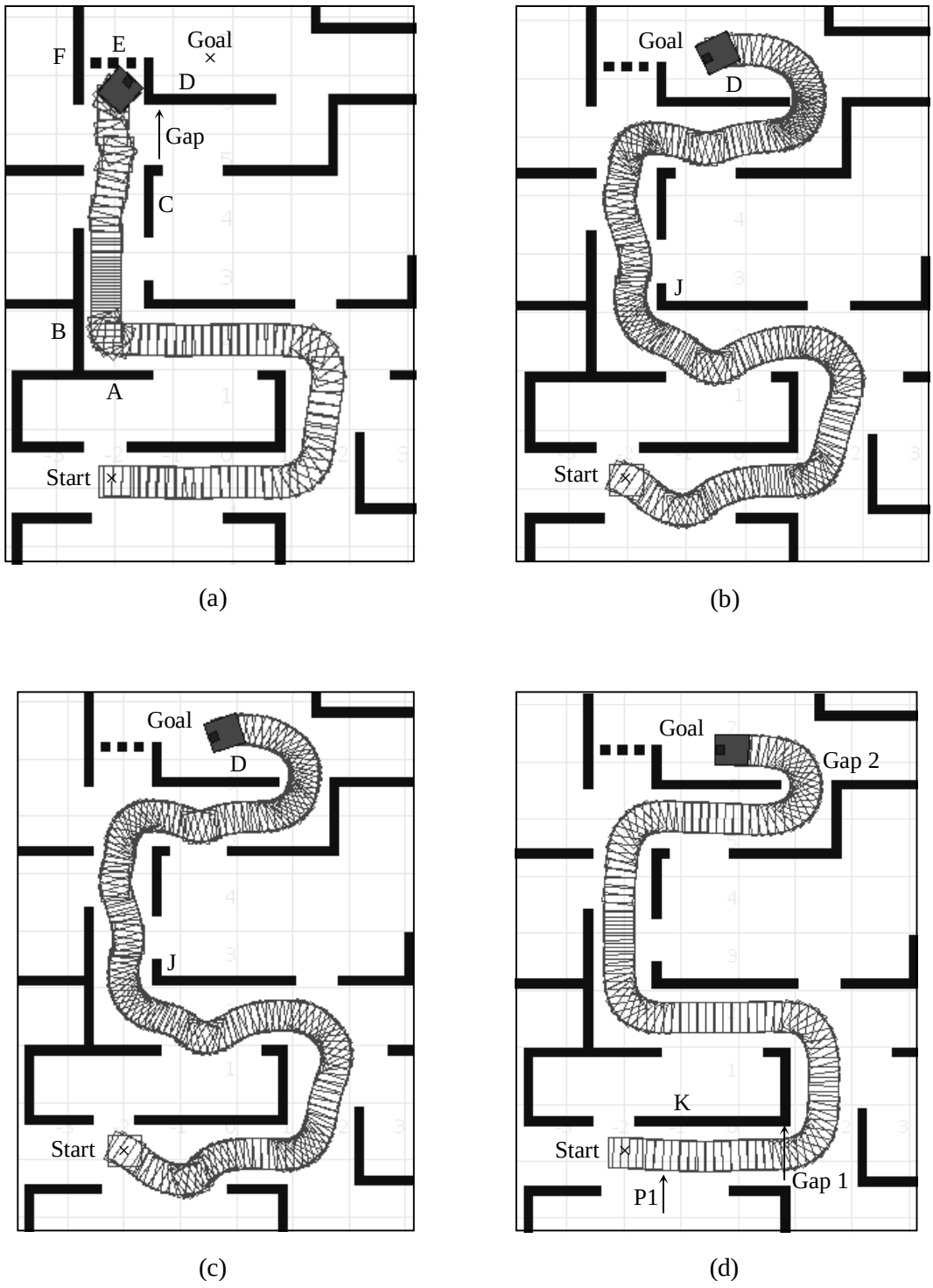


Figure 4.7: TCG simulation trajectories. (a) Trajectory followed by TE where it touches obstacles and stops before reaching the goal. (b) Trajectory followed by SND. (c) Trajectory followed by CG. (e) The best trajectory which is followed by TCG.

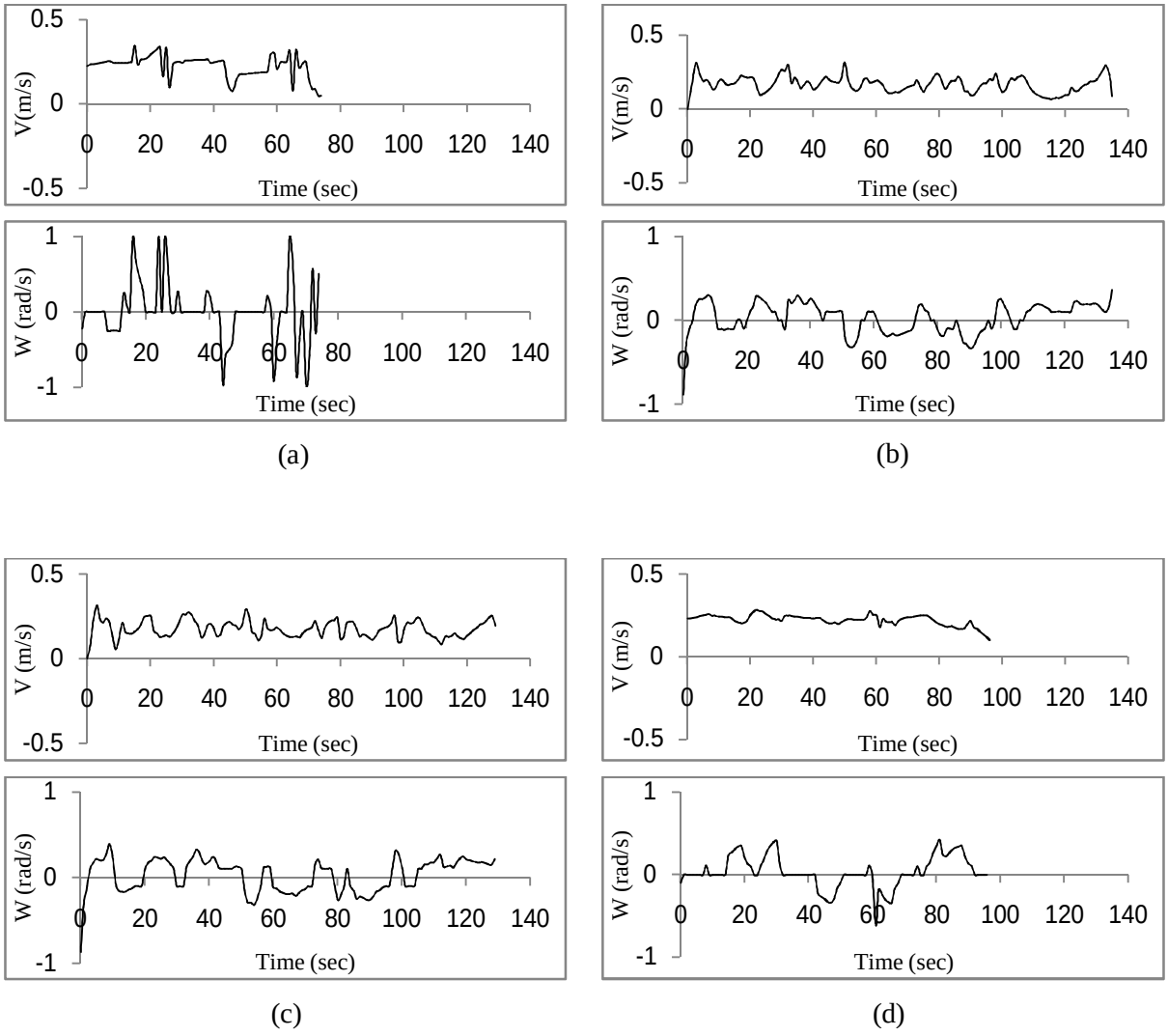


Figure 4.8: TCG simulation analyses. (a) Translational and rotational velocities versus time for Tangential Escape (TE) approach where high oscillations and deadlock appear. (b) Translational and rotational velocities versus time for Smooth Nearness-Diagram (SND) method, where better results are achieved as compared with the TE. (c) Translational and rotational velocities versus time for Closest Gap (CG) approach where the resultant curve are similar to SND method. (d) Less oscillatory behavior appears from plotting translational and rotational velocities versus time for Tangential Closest Gap (TCG) approach.

4.5.2 SND and CG Simulations

The path followed using SND and CG methods are shown in Fig. 4.7b and Fig. 4.7c. No actual differences are observed in the trajectories of the two approaches except on two points where the robot gets very close to obstacles on the sides labeled *D* and *J* running the SND algorithm. This is because sides *D* and *J* have low threats as compared with their opposite sides relative to the robot heading direction. Moreover, the goal reached in 135 sec using the SND method and 129 sec using the CG. Fig. 4.8b and Fig. 4.8c show the recorded motion commands (v and w) versus time for the SND and CG methods, where it can be noticed that there is no considerable difference in smoothness.

4.5.3 TCG Simulation

Fig. 4.7d shows the trajectory obtained by the TCG method. On the starting point the closest obstacle will be on the side labeled *K* and the sub-goal on the mid of Gap 1. The angular distance between them $\mathrm{dist}(\alpha, \beta)$ is less than $\pi/2$. Therefore, the robot navigates in parallel to the tangent of side *K* obstacles. When the robot reaches point P1, the angular difference will get around $\pi/2$. So the leaving condition is satisfied and the progress will be directly towards the sub-goal. This process will be repeated till passing all gaps and finally reaching the real goal. The right rotation angle in the direction close to the goal/sub-goal is always determined. One can notice from the trajectory near Gap 2 that the problem of the deadlock does not appear in this approach. The goal is reached in 95 sec only. The TCG method drives the robot much faster with a better trajectory than the previous methods. This can be noticed from the density of bounding boxes on the map and from plotting the recorded motion commands versus time as shown in Fig. 4.8d.

4.6 Experimental Results

We tested the TCG method on a differential drive Pioneer 3-AT mobile robot equipped with an on-board 2 GHZ Pentium M computer and a Hokuyo URG-04LX laser scanner. The specifications of the laser scanner are the same as the simulated one introduced in section

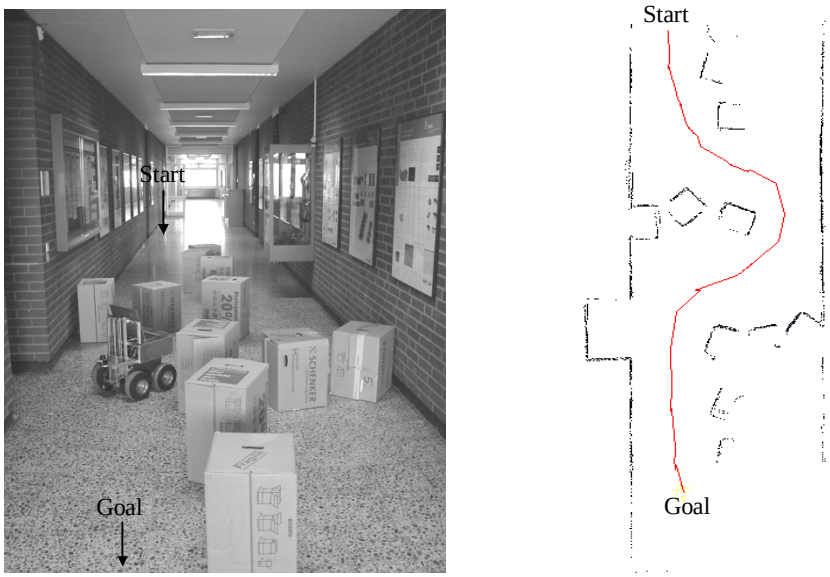
4.5. The mobile robot has a rectangular shape (0.53 × 0.49 *meters*) with two actuator wheels and non- holonomic constraints. This mobile robot has a maximum travel speed of 0.7 *m*/*s* and a maximum rotational velocity of 140 *deg*/*s*. These velocities were limited to (0.3 *m*/*s*, 0.6 *rad*/*s*) while carrying out the experiments. The safe distance D_s and velocity safe distance D_{vs} were set to 1 m. The sampling interval and the time constant of the low-pass filter were set to 0.1 sec. The experiments were carried out in collaboration with the Get Lap research group at the University of Paderborn, Germany.

The non-holonomic constraints are not taken into consideration in this algorithm. Moreover, the experiments were carried out without moving (dynamic) obstacles.

We ran the TCG-controlled Pioneer 3-AT through difficult courses. The obstacles were boxes, tables, chairs and even dynamic obstacles (such as people). Fig. 4.9a shows one of these experiments where none of the obstacle locations were known to the robot in advance. In this scenario, the obstacles were boxes randomly distributed along a corridor as shown in the figure. The start and goal locations are also shown.

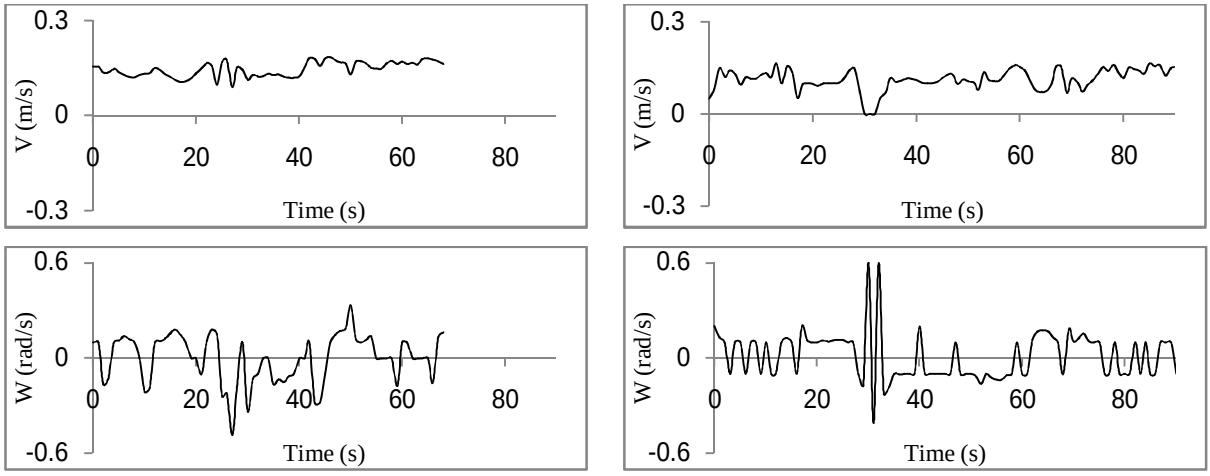
The resultant trajectory that the robot followed to reach its goal is shown in Fig. 4.9b. We recorded the linear and angular velocities (v and w) over the course of this experiment. Fig. 4.10a shows v and w for the TCG method and are plotted against the time elapsed by the robot. The experiment was carried out again using the CG method this time. The recorded linear and angular velocities versus time for the CG method are shown in Fig. 4.10b. The goal was reached in 68 *sec* using the TCG method while it took 90 *sec* in the case of the CG.

One can notice from these results (Fig. 4.10) that the TCG generates less oscillatory robot trajectories. Moreover, it reaches the goal faster than the predecessor techniques such as ND, SND and CG methods.



(a) (b)

Figure 4.9: TCG experimental results. (a) Experimental setup. (b) Trajectory followed by the TCG.



(a) (b)

Figure 4.10: TCG experimental analyses. (a) Translational and rotational velocities versus time for the Tangential Closest Gap (TCG) approach. (b) More oscillatory behavior appears from plotting translational and rotational velocities versus time for the Closest Gap (CG) approach.

4.7 Discussion

We present in this subsection a discussion showing the advantages of the TCG method as compared with the CG approach on the basis of the drawbacks and limitations mentioned in section 4.1.

With this method, the robot generates **less oscillatory trajectories**. This is due to the fact that avoiding obstacles falling within the safe distance in the CG approach is done by deflecting the robot heading angle by an angle that is a factor of 180 degrees (depending on the proximity of the obstacle from the robot boundary), whereas the TCG method forces the robot to navigate in parallel to the tangent of the closest obstacle. Fig. 4.11 shows two narrow gaps that the robot should pass through in order to reach its goal. Using the CG method, the following occurs. On the starting point, the obstacles on the right side of the robot (labeled C) deflect the robot away towards the free space labeled F1. When the robots gets closer to the obstacles on side A, it is sharply deflected to move nearly to the mid of the gap between sides A and B. After passing the gap, the robot then moves towards the free space labeled F2. The same is done when passing the gap between sides B and C. Using the TCG method, the robot navigates in parallel to side C obstacles at first. When passing the gap between sides A and B, the robot either moves in parallel to side A or side B obstacles. This depends on the side closest to the goal (in this scenario, it moves in parallel to side C obstacles). It is clear that the TCG method produces less oscillatory robot paths than the CG approach.

One can notice that the oscillatory path produced by the CG method requires longer time for the robot to reach its goal. Therefore the TCG approach can drive the robot much **faster** than the CG method.

Increasing the safe distance in the CG approach will deflect the robot more towards the free spaces (e.g. F1 and F2 in Fig. 4.11). This increases oscillations and decreases the speed. In the TCG method, increasing the safe distance does not affect its behavior since it always move in parallel to the tangent of the closest obstacles (no deflection).

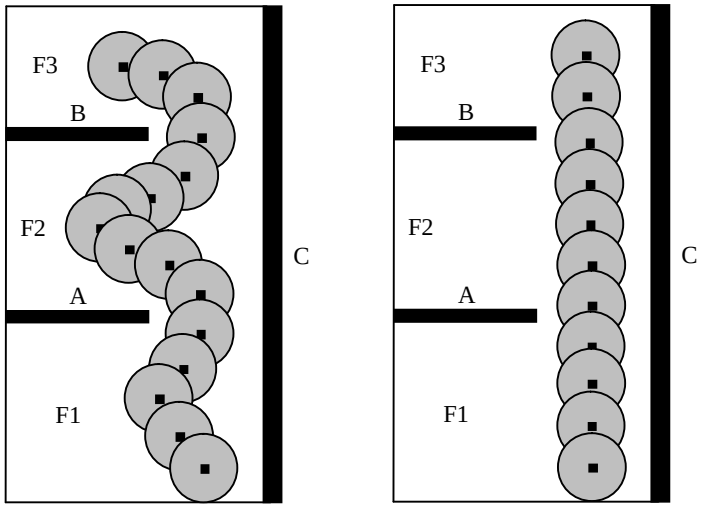


Figure 4.11: (a) Navigation in narrow corridors by CG. (b) Navigation in narrow corridors by TCG where faster behavior and less oscillatory robot trajectories are achieved.

4.8 Conclusions

We have presented in this chapter the Tangential Closest Gap (TCG) Navigation method for complex environments. This approach drives the robot much faster with less oscillatory motion than the well known approaches designed to operate in such environments. This is achieved by integrating two methods: the Closest Gap (CG) and the Tangential Escape (TE) navigation methods. The whole system is controlled by choosing motion commands that are stable in the Lyapunov sense. Comparisons between previous works and the TCG in simulations and experiments demonstrate the advantages of this combination.

Chapter 5

Smooth Tangential Closest Gap Navigation

5.1 Introduction

As stated previously, safe navigation in highly cluttered environments, where robots are always required to move, is an important research area. We have addressed in chapter 3 the Closest Gap Navigation (CG) method to deal with such environments. The CG method overcomes significant problems in the well known Smoothness Nearness-Diagram Navigation (SND) approach [16] as shown previously. This method is then integrated with the Tangential Escape scheme leading to the Tangential Closest Gap (TCG) approach, introduced in chapter 4. It has been shown that the TCG method produces faster and less oscillatory robot trajectories. However, deflecting the robot heading angle in the TCG approach, when obstacles fall within the security zone around the robot, depends only on the obstacle closest to the robot. If the shape of this obstacle is polygonal, this method achieves good smoothness except on the points where the leave condition is satisfied. If not, the smoothness of the resultant trajectory is reduced.

This chapter presents further enhancements of the TCG obstacle avoidance approach, entitled Smooth Tangential Closet Gap (STCG) method. As compared with the TCG technique, the key difference in this approach is that calculating the rotation angle is based on all nearby obstacles, not just the closest one. This removes the abrupt transitions in behavior, particularly when the shape of the closest obstacles is not polygonal. The oscillatory behavior caused from switching between navigating in parallel to the tangent of the closest obstacle and resuming the progress towards the goal (satisfying the leaving condition) are also removed. Therefore, a very smooth path is generated as will be shown in section 5.3. Moreover, we adjust the algorithm of calculating the rotation angle to be more reactive,

particularly for dynamic obstacles that make an angle around 90 degrees with the robot heading and get closer to the robot as they move.

This chapter is organized as follows: The proposed reactive navigation method is introduced in Section 5.2. In Section 5.3, the simulation results are described, and in Section 5.4 we present some conclusions.

5.2 The Reactive Navigation Method Design

This section presents the Smooth Tangential Closest Gap (STCG) navigation method for obstacle avoidance that works as follows: first, each sensor update the laser rangefinder data is checked to find opening surrounding the robot as will be described in subsection 5.2.1. The goal position is rotated by an angle forcing the robot to move inside the navigable gap closest to the goal. In subsection 5.2.2, we explain how to calculate this rotation angle which we call the *gap rotation angle* φ_g. The existence of obstacles inside a predefined safe distance D_s causes another rotation for the goal. We refer to the angle of this rotation by the *collision avoidance rotation angle* φ_c which is computed as will be illustrated in subsection 5.2.3.

5.2.1 Finding Gaps

The method for finding gaps is the same as in the CG approach explained in chapter 3. In this section, a brief description of this method for finding gaps around the robot is presented. Before that, consider the following two types of discontinuities introduced in chapter 3.

Type 1 discontinuity: Occurs when the difference between the depth measurements of scans i and j exceeds the robot diameter.

Type 2 discontinuity: Occurs if one of the two measurements returns the maximum sensor range.

if $j > i$, a *rising* discontinuity occurs at scan i; else, it will be a *descending* discontinuity at scan j. Type 1 discontinuity has a higher priority than type 2.

Suppose that the first laser scan point is 0 and the final one is N. The scanning for gaps is done twice; the first is called a *forward scan* from scan 0 to $N-1$ and the other is a

backward scan from scan $N - 1$ to 0. In the forward scan, a search for rising discontinuities is done. The scan point i at which the discontinuity occurs determines the first side of the gap (e.g. points E and A in Fig. 5.1). Determining the second side depends on the discontinuity type. If it is type 1, the second side will be at scan number $j > i + 1$ where the distance from the obstacle point it returns and the first side is the minimum and the angle between them relative to the current robot location should not exceed π (see point J, Fig. 5.1). If the discontinuity is type 2, the second side occurs at scan number $j > i + 1$ when a descending continuity is fetched at this point (point B, Fig. 5.1). After that, the forward scan is resumed starting from scan $j + 1$. This scan produces gaps 1, 3, 5 and 7 in Fig. 5.1.

In the backward scan, the search is done for descending discontinuities which determine the second side j of the gap (e.g. points D and H in Fig. 5.1). The first side depends on the discontinuity type also. For a type 1 discontinuity, the first side will be at scan number $i < j - 1$ where the distance from the obstacle point it returns and the second side is the minimum and the angle between them should not exceed π also (see point C, Fig. 5.1). But if the discontinuity is type 2, the first side occurs at scan number $i < j - 1$ where a rising discontinuity should be fetched at this point (point G in Fig. 5.1). The backward scan is resumed from scan $i - 1$. This scan produces gaps 2, 4, 6 and 8 in Fig. 5.1.

Once the list of gaps is assembled, eliminate every gap that exists inside another gap (e.g. gaps 1, 3, 6 and 8, Fig. 5.1) and then remove from the remaining gaps every one with a width less than the robot diameter.

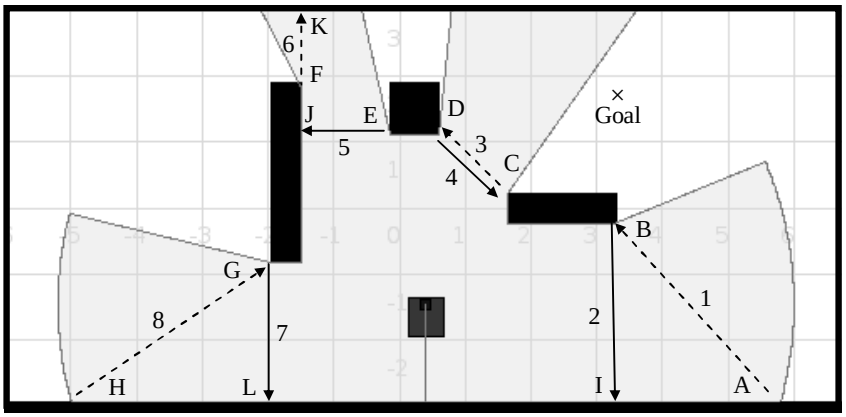


Figure 5.1: Finding Gaps by the GG method.

5.2.2 Gap Rotation Angle

After finding the list of gaps identified by their sides (first and second sides), the angles toward the two sides of each gap are compared to the angle towards the goal θ_{goal}. The navigable gap that has the side closest to the goal is then chosen from among these gaps (gap 4 in Fig. 5.1). We refer to this side as θ_{cs} (point C) and the other as θ_{os} (point D).

The closest gap is considered as the best step towards the goal. Navigating through this gap requires switching the goal position by an angle (which we call the gap rotation angle) that ensures safe motion towards the gap. The direction of safe motion is identified in the CG approach (refer to chapter 3, Eq. 3.8) as:

$$\theta_{\text{md}} = \begin{cases} \theta_{\text{goal}}, & \text{if } \theta_{\text{goal}} \in \text{closest gap}, \\ \theta_{\text{mid}}, & \text{if } \text{dist}(\theta_{\text{cs}}, \theta_{\text{mid}}) < \text{dist}(\theta_{\text{cs}}, \theta_{\text{scs}}), \\ \theta_{\text{scs}}, & \text{otherwise}, \end{cases} \tag{5.1}$$

where θ_{scs} ensures that the obstacles creating the side closest to the goal will not enter the safe distance of the boundary of the robot as the robot moves towards this side. θ_{mid} is the angle towards the mid of the gap (between θ_{cs} and θ_{os}) which is selected if the gap is narrow. $\text{dist}(\alpha, \beta)$ returns the angular distance between α and β.

This direction is then modified to take into consideration the angular width of the gap relative to the robot vision. One can refer to chapter 3 for more information about setting the values of θ_{mid} and θ_{scs}, defining the function $\text{dist}(\alpha, \beta)$ and modifying θ_{md}.

With this, the gap rotation angle is now defined as follows:

$$\varphi_{\text{g}} = \theta_{\text{goal}} - \theta_{\text{md}}. \tag{5.2}$$

One exception occurs when a direct free-way to the goal exists. In this case, we set $\varphi_{\text{g}} = 0$ without analyzing gaps at all. The algorithm for checking the free-way to a goal condition is presented in Appendix A.

One can notice that the mechanism for analyzing gaps and choosing the gap rotation angle is the same as in the TCG method stated in chapter 4; the difference comes with calculating the collision avoidance rotation angle.

We refer to the new position after rotating the goal location by φ_{g} as the *sub-goal* position X_{sg} which is calculated as follows:

$$X_{sg} = \begin{bmatrix} \cos(\varphi_g) & \sin(\varphi_g) \\ -\sin(\varphi_g) & \cos(\varphi_g) \end{bmatrix} X_{goal} \tag{5.3}$$

If φ_g is positive, the goal position is rotated clockwise relative to the robot. Otherwise, it will rotate counterclockwise.

5.2.3 Collision Avoidance Rotation Angle

The STCG method will consider switching the sub-goal position (that results from rotating the goal position by the gap rotation angle φ_g identified in subsection 5.2.2) based on the structure of obstacles surrounding the robot. The TCG method (introduced in chapter 4) forces the robot to move in parallel to the tangent of the closest obstacle. This achieves good smoothness in case of uniform or polygonal obstacle shapes. However, for most applications in mobile robotics the environment has a random distribution of obstacles, which reduces smoothness in this method. Moreover, the switching to the leaving condition causes an abruptly transition which produces an oscillatory behavior. The STCG achieves better smoothness and decreases oscillations by considering all of the obstacles around the robot, not just the closest one. In addition, it modifies the angle of rotation to be more reactive for dynamic obstacles. Before explaining the method, consider the following two definitions presented in chapters 4 and 3, respectively:

Definition 1: let θ_1 and θ_2 any two angles relative to the robot location, the angular distance between θ_1 and θ_2 is defined as follows.

$$\text{dist}(\theta_1, \theta_2) = \begin{cases} 2\pi + (\theta_1 - \theta_2), & \text{if}(\theta_1 - \theta_2) < -\pi, \\ 2\pi - (\theta_1 - \theta_2), & \text{if}(\theta_1 - \theta_2) > \pi, \\ |\theta_1 - \theta_2|, & \text{otherwise}, \end{cases} \tag{5.4}$$

Definition 2: Assume that a < b, the saturation function, which is used to limit a value between two boundaries, is defined as follows:

$$\text{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x \le a, \\ x, & \text{if } a < x < b, \\ b, & \text{if } x \ge b. \end{cases} \tag{5.5}$$

Suppose that N obstacles fall within the safety distance D_s of the boundary of the robot. Each of these N obstacles causes a rotation angle φ_i for the sub-goal position. The value of φ_i depends on the *angular distance* between the angle towards the sub-goal and the angle towards this obstacle. If it is more than 90 degrees, we set $\varphi_i = 0$ since no need to move the robot away from obstacles in this way (going directly to the goal is more reactive than following the contour); else, the value of φ_i is set to make an angular distance of 90 degrees between the robot heading direction and the angle towards the obstacle as follows:

$$\text{If } (\text{sign}(\theta_{sg}) \; != \text{sign}(\theta_i))$$

$$\varphi_i = \begin{cases} \text{sign}(\theta_i)\frac{\pi}{2} - (\theta_i - \theta_{sg}), & \text{if } |\theta_i - \theta_{sg}| < \frac{\pi}{2}, \\ -\text{sign}(\theta_i)\frac{\pi}{2} - (\theta_i - \theta_{sg}), & \text{if } |\theta_i - \theta_{sg}| > \frac{3\pi}{2}, \end{cases}$$

Else

$$\varphi_i = \begin{cases} \text{sign}(\theta_i)\frac{\pi}{2} - (\theta_i - \theta_{sg}), & \text{if } |\theta_i| > |\theta_{sg}|, \\ -\text{sign}(\theta_i)\frac{\pi}{2} - (\theta_i - \theta_{sg}), & \text{if } |\theta_i| < |\theta_{sg}|, \end{cases}$$

where θ_i is the angle towards the i^{th} obstacle point and θ_{sg} the angle towards the sub-goal (inside the navigable gap closest to the goal). Positive angles are to the left of the robot whereas negative angles are to the right. Setting φ_i should force the robot to navigate, making an angle of $\frac{\pi}{2}$ with the obstacle point i, in the direction of the sub-goal. To achieve this: In case of unequal signs for the angles θ_{sg} and θ_i, setting φ_i depends on the path length from θ_{sg} to θ_i travelling clockwise if θ_{sg} is positive and counterclockwise if it is negative. The direction of motion will be towards the positive x axis relative to the robot if the path length is less than $\frac{\pi}{2}$ or towards the negative x axis if it is more than $\frac{3\pi}{2}$. In case of equal signs, the direction will be towards the positive x axis if the absolute value of the angle θ_i is more than the absolute value of θ_{sg} or towards the negative x axis if it is not.

We adjust this angle to better deal with dynamic obstacles near the sides of the robot, particularly for those making an angle around 90 degrees relative to the robot heading angle. Fig. 5.2 shows a situation where obstacles are very close to the right side of the robot (the side labeled RS). Navigating in parallel to the tangent of these obstacles (following the contour of obstacles on this side) is dangerous for the robot especially if they are dynamic

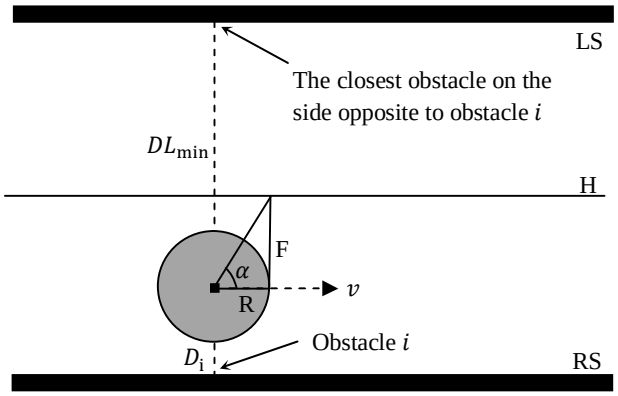


Figure 5.2: Adding the angle α in calculating the collision avoidance rotation angle for each obstacle to achieve safer navigation for obstacles making an angle around 90 degrees with the robot heading.

ones and gets closer to the robot as they move. This problem can be solved by forcing the robot to move far from obstacles with a specified safe distance (which we call the reactive safe distance D_{rs}). If the obstacles on the opposite side (the side labeled LS in Fig. 5.2) are near the robot, the robot should move to the mid of the distance between the obstacles on the two sides (along the line labeled H in Fig. 5.2). To achieve this, we modify φ_{i} defined in the above algorithm for each obstacle by adding an angle α that is calculated as follows:

$$\alpha = \text{sign}(\theta_{\text{i}}) \arctan\left(\frac{F}{R}\right) \tag{5.6}$$

where F is the distance from the current robot location to the point that is the mid of the distance from the obstacle point i to the obstacle point closest to the robot on the side opposite to obstacle i. We multiply by the sign of the angle towards obstacle i to move to the right direction. The value of F is defined below:

$$F = \text{sat}_{[0,(D_{\text{rs}}-D_{\text{i}})]}\left(\frac{DL_{\min}-D_{\text{i}}}{2}\right) \tag{5.7}$$

where D_{i} is the distance towards obstacle i, $DL_{\min}$ the distance towards the obstacle point closest to the robot on the side opposite to obstacle i (the left side in Fig. 3) and D_{rs} the reactive safe distance. The sat function is used to cap the value of F at 0 when $D_{\text{i}} > DL_{\min}$, at $D_{\text{rs}} - D_{\text{i}}$ (to reach the full reactive safe distance) when the distance F exceeds the difference between the reactive safe distance and the distance to obstacle i.

To identify the relative weight of each obstacle rotation angle φ_{i} we use the following function, which depends on the proximity of the obstacle from the robot boundary:

$$w_{\mathrm{i}} = \frac{1}{[1-\mathrm{sat}_{[0,1]}\left(\frac{D_{\mathrm{S}}-D_{\mathrm{i}}}{D_{\mathrm{S}}}\right)]^{k}} \tag{5.8}$$

where D_{i} is the distance to the i^{th} obstacle point from the robot boundary and k defines the strength of the weight. We use the weight in this manner to strictly differentiate between close and far obstacles especially for the closest ones (see the CG approach, chapter 3). This is to achieve safer navigation.

Now, suppose that N_{pos} and N_{neg} are the number of obstacles within D_{S} that cause *positive* and *negative* rotation angles, respectively. We compute the total positive and negative rotation angles caused from their corresponding obstacles individually. Then, the net rotation angle caused from all obstacles is calculated after adjusting the difference in the number of obstacles that cause positive and negative rotation angles. This is to avoid the problem of the deadlock that occurs in the SND approach, when one side of the robot has a large number of obstacles as compared with the other one.

The total weights for obstacles causing positive rotation angles can be defined as:

$$W_{\mathrm{pos}} = \sum_{\mathrm{i}=1}^{N_{\mathrm{pos}}} w_{\mathrm{i}} \tag{5.9}$$

We can now define the total positive rotation angles as the sum of all positive φ_{i} multiplied by its relative weight.

$$\varphi_{\mathrm{pos}} = \sum_{i=1}^{N_{\mathrm{pos}}} \frac{w_{\mathrm{i}}}{W_{\mathrm{pos}}} \varphi_{\mathrm{i}} \tag{5.10}$$

For negative rotation angles, W_{neg} and φ_{neg} are also found in the same manner as Eq. (5.9) and Eq. (5.10).

Let us call the smaller of the two numbers: N_{pos} and N_{neg} as min and the other as max. The ratio between them is defined as: $r = min/max$ (we set $r = 1$ if either min or max equals zero). Before calculating the net rotation angle, we multiply either the total positive or negative rotation angles (φ_{pos} or φ_{neg}) by the ratio r, in order to adjust the difference between N_{pos} and N_{neg}, as follows:

$$\text{If } (N_{\text{pos}} > N_{\text{neg}})$$

$$\varphi_{\text{pos}} = r . \varphi_{\text{pos}}$$

$$\text{Else}$$

$$\varphi_{\text{neg}} = r . \varphi_{\text{neg}}$$

With this we can now define the net collision avoidance rotation angle as the weighted sum of φ_{pos} and φ_{neg}.

$$\varphi_{\text{c}} = \frac{1}{r} . \frac{W_{\text{pos}} . \varphi_{\text{pos}} + W_{\text{neg}} . \varphi_{\text{neg}}}{W_{\text{pos}} + W_{\text{neg}}} \tag{5.11}$$

where φ_{c} is multiplied by $1/r$ to re-adjust its value.

Remark 5.1: Considering the ratio r as min/max equates the number of obstacles that results in positive and negative rotation angles exactly. It is safer and smoother to keep a little bit difference between them especially if the ratio is very low. Hence, we propose to make the ratio $r = sqrt(min/max)$ in the above equations.

Avoiding collisions with nearby obstacles requires rotating the sub-goal position (that results from rotating the goal position by the angle φ_{g} presented in subsection 5.2.2) by the angle φ_{c}. Each sensor update, the resultant current goal position will be:

$$\mathrm{X}_{\text{cg}} = \begin{bmatrix} \cos(\varphi_{\text{c}}) & \sin(\varphi_{\text{c}}) \\ -\sin(\varphi_{\text{c}}) & \cos(\varphi_{\text{c}}) \end{bmatrix} \mathrm{X}_{\text{sg}} \tag{5.12}$$

where X_{sg} is the sub-goal position.

According to the motion commands that drive the robot to the specified current goal position, we use the same equations of the TCG approach proposed in chapter 4:

$$v = k_b v_{\text{limit}} \cos(\theta_{\text{cg}}) \tag{5.13}$$

$$w = \text{sat}_{[-w_{\text{max}}, w_{\text{max}}]}\left(k_{\text{w}}\theta_{\text{cg}} + \frac{v \sin(\theta_{\text{cg}})}{\rho}\right), \quad k_{\text{w}} > 0 \tag{5.14}$$

where θ_{cg} is the angle towards the current goal location, w_{max} the maximum rotational velocity obtained from the data sheet of the robot and v_{limit} defined in Eq. (4.15).

5.3 Simulations

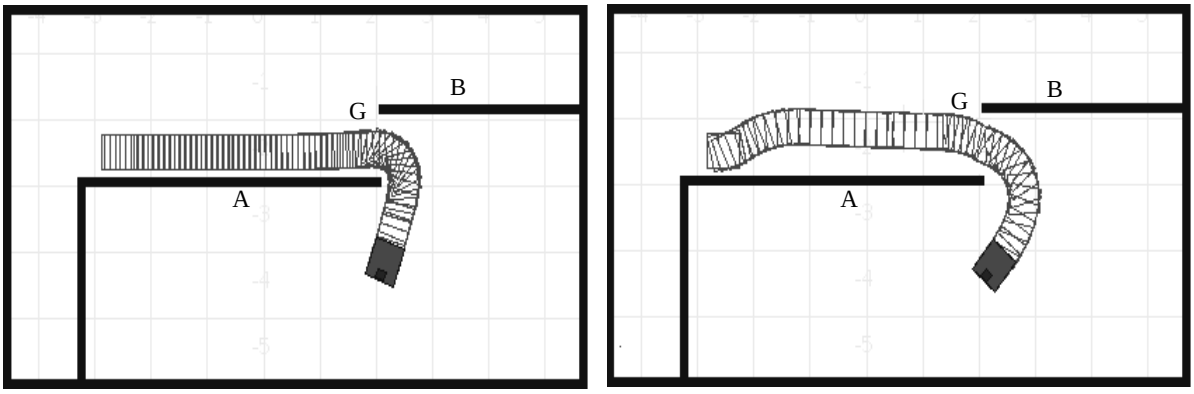
The main contribution of the TCG approach appears in driving the robot much smoother and even faster than the predecessor techniques such as the CG and the TCG methods. To demonstrate the differences in the execution of these methods, we implemented them in the Player/Stage robot software system version 2.1.1. The simulated robot is a rectangular one ($0.59 \times 0.53\ meters$) that works in a differential-driven mode. The sensing system adopted is a laser scanner installed with a distance of $0.11\ meters$ from the front of the robot towards its center, which delivers 683 measurements over 240° covering a range of 5.6 m at maximum. The maximum operational velocities were set to ($0.5\ m/s, 1\ rad/s$) while the safe distance was set to 1 m. According to the velocity safe distance, it was set to 1 m.

We outline next two scenarios, the first one shows how adding the angle α to the collision avoidance rotation angle φ_{i} proposed in section 5.2.3 achieves safer trajectories, particularly in avoiding dynamic obstacles and the other demonstrates the increased smoothness of the STCG method. In the two scenarios, the reactive safe distance D_{rs}, in the STCG approach, was set to $0.4\ m$.

5.3.1 Scenario (1) Simulations

For this scenario a simple map was created where obstacles are chosen to be very close from the right side of the robot at the starting point. The map can be seen in Fig. 5.3, where the block regions are obstacles. The simulated scenario was carried out twice, one using the TCG method and the other using the STCG approach.

For TCG simulation, the robot navigates keeping nearly the same distance from the obstacles labeled A on the right side till reaching the gap G as shown in Fig. 5.3a. This is due to the fact that the path chosen, whenever an obstacle enters the security zone around the robot where the angular distance between the angle towards this obstacle (the closest one) and the angle towards the sub-goal does not exceed 90 degrees, is tangent to the obstacle boundary. If this angular distance gets more than 90 degrees, the leaving condition will be satisfied and the progress is resumed towards the sub-goal which is the point in the mid of the narrow gap G. In the two cases, the robot does not move away from obstacles and the



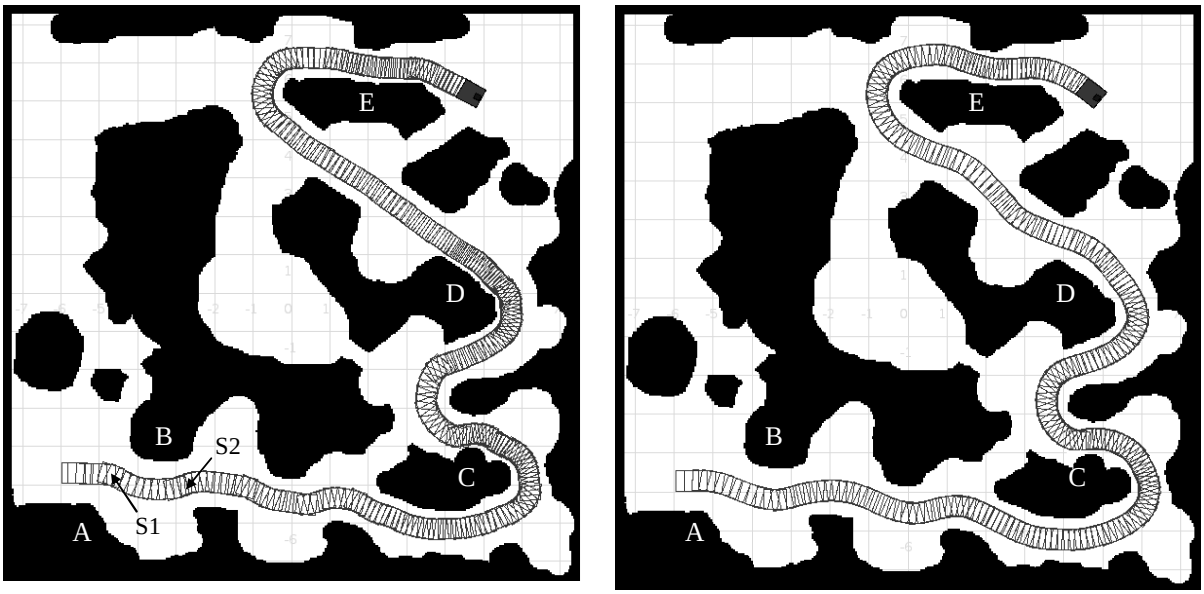
(a) (b)

Figure 5.3: STCG scenario 1 simulation results. (a) Trajectory followed implementing the TCG method where the robot navigates very close from the obstacles on the right side. (b) Trajectory followed by the STCG where the robot moves away from the obstacles at first to keep the robot far from them with a safe distance (D_{rs}). Then, the robot keeps navigating following the contour of obstacles till reaching the gap G where the robot moves towards the mid of the distance between obstacles on the two sides.

route chosen is dangerous which may lead to collisions, particularly if the obstacles on this side are dynamic ones and gets closer to the robot as they move. Adding the angle α when calculating the rotation angle, in the STCG method, pushes the robot to be deflected and navigate far from obstacles with a predefined safe distance (the reactive safe distance D_{rs}). Fig. 5.3b shows how the robot is repelled from the obstacles on side A and then moves following their contour, while keeping the reactive safe distance between them, till reaching gap G. Entering gap G will force the robot to move towards the mid of the distance between sides A and B, which is safer than deflecting by the full reactive safe distance.

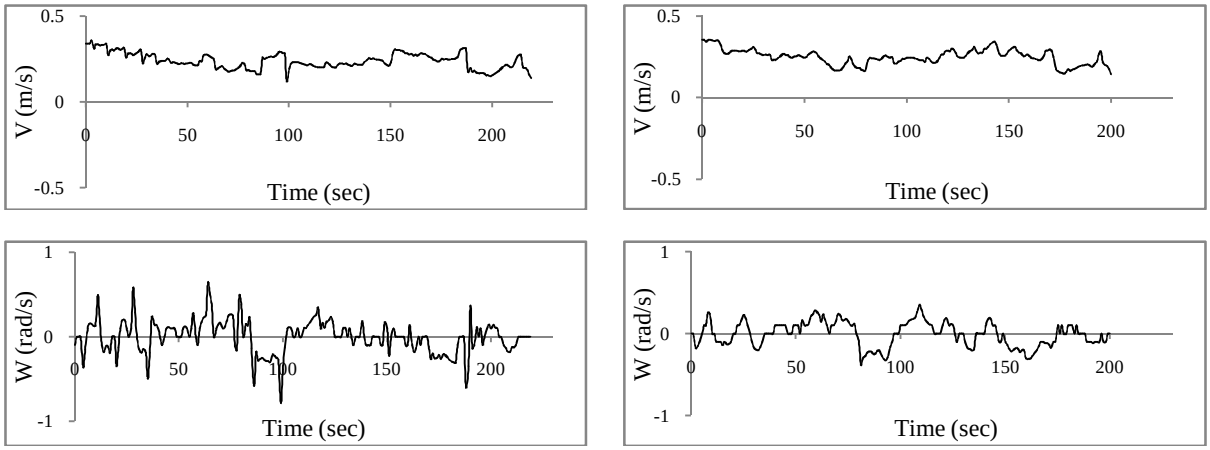
5.3.2 Scenario (2) Simulations

In order to show how the motion commands generated by the STCG approach provides smoother and faster trajectories than the TCG method, we created a map with many tight and narrow corridors where obstacles are randomly distributed on the map (look at Fig. 5.4). For the sake of comparison, this scenario was executed using the implementation of each of the two methods; TCG and STCG.



(a) (b)

Figure 5.4: STCG scenario 2 simulation results. (a) Path followed by TCG method through tight corridors where the robot either navigates in parallel to the tangent of the closest obstacle or towards the sub-goal directly (satisfying the leaving condition). The robot navigates very close from obstacles on sides C, D and E. (b) Trajectory followed by STCG method using the same conditions as Fig. 5.4a where a smoother and safer behavior are achieved by using a weighted sum of all obstacle points and keeping D_{rs} far from obstacles while following their contour.



(a) (b)

Figure 5.5: Linear and angular velocities versus time for (a) TCG and (b) STCG.

Applying the TCG algorithm produces the trajectory shown in Fig. 5.4a. On the starting point the leaving condition is satisfied because the angular distance between the angle towards the sub-goal θ_{sg} (labeled S1 on the figure) and the angle towards the closest obstacle β (on the side labeled A) exceeds 90 degrees. The robot navigates directly towards this sub-goal till the distance towards the obstacles on side B gets less than those on side A. In this case, the angular distance between θ_{sg} (the sub-goal S2 this time) and β (on side B) gets less than 90 degrees forcing the robot to move in tangent to the obstacle boundary on side B. After that, the leaving condition is satisfied again and the robot moves towards the sub-goal S2. These processes will be repeated till reaching the goal. It is clear that the switching to the leaving condition causes sharp changes in the trajectory. Moreover, the direction towards the closest obstacle point will change frequently as the robot moves, particularly in tight corridors with many obstacle points, which reduces smoothness. The robot moves very close from obstacles on the sides labeled C, D and E, which results from following the contour of obstacles without keeping a safe distance between the robot and the obstacles, as explained in the previous scenario. By using a weighted sum over all the obstacle points, STCG avoids the sharp changes in the path generated and produces a smoother behavior, which can be sensed from the square bounding boxes left behind on the map during navigation as shown in Fig. 5.4b. Furthermore, adding the angle α in calculating the rotation angle achieves a safer trajectory along the entire path. The points labeled C, D and E in Fig. 5.4b demonstrate how the robot is deflected away from obstacles while following their contour, and how it avoided the limitations of the TCG method shown on Fig. 5.4a on the same points.

We recorded the translational and rotational velocities over the course of scenario 2 simulations. In order to prove the increased smoothness of the STCG method, the translational and rotational velocities are plotted against the time elapsed by the robot along the entire path. Fig. 5.5a shows the resultant curves for the TCG while Fig. 5.5b shows the curves of the STCG. One can notice that the STCG method drives the robot much smoother than the TCG approach. The goal was reached in 220 sec using the TCG approach and 200 sec implementing the STCG method.

5.4 Conclusion

This chapter presented the Smooth Tangential Closest Gap (STCG) local reactive navigation method for mobile robots, moving in very cluttered and complex environments. The STCG method improves the smoothness of the paths generated by the TCG method through considering all obstacle points surrounding the robot, which fall inside the safe distance, in generating the motion law, not just the closest one. It also adjusts the motion law to keep a safe distance between the robot and the obstacles while circumnavigating them. If the obstacles on the two sides are very close from the robot, the navigation will be through the mid of the distance between them. Therefore, a safer trajectory is achieved.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

This work contributes to the reactive collision avoidance (sensor-based navigation) field. A new approach, entitled Closest Gap (CG) Navigation, for mobile robots navigating in a very cluttered, dense and complex environment has been developed and implemented. This approach is designed depending on two previous works; the Nearness-Diagram (ND) and the Smooth Nearness-Diagram (SND) navigation methods.

The CG approach, addressed in chapter 3, proposes a new procedure for fetching gaps surrounding the robot. This procedure alleviates oscillations and reduces computational complexity by eliminating unnecessary gaps. The CG method increases the smoothness of the paths generated by considering the angular width of the chosen gap with respect to the robot vision. Furthermore, it generates safer paths than the Smooth Nearness-Diagram (SND) Navigation method through considering the ratio of threat counts on the two sides of the robot and applying stricter deviation against an obstacle as it gets closer to the robot. Therefore, a robust navigation in very dense and cluttered scenarios is achieved.

The CG approach is then integrated with the Tangential Escape method. The new method is called the Tangential Closest Gap Navigation (TCG) technique. This approach is proposed in chapter 4. It increases the speed of the robot, particularly for avoiding dynamic

obstacles, while reducing oscillations. The motion commands are chosen in this approach where the whole control system is asymptotically stable in the Layabunov sense.

The TCG scheme is further developed in chapter 5, and named the Smooth Tangential Closest Gap Navigation (STCG) method, by considering all obstacle points surrounding the robot in generating the motion laws, not just the closest one (as in TCG). Hence, it can drive the robot much smoother than the TCG method. Moreover, it improves the safety of paths generated by the TCG by keeping a safe distance between the robot and obstacles while following their contour, in case of moving through a wide corridor. But if the corridor is narrow, the robot is forced to move in the mid of the distance between the obstacles on the two sides of the corridor.

6.2 Future Works

Future work will focus on the following subjects:

- Finding an analytical fully proofed solution that ensures safety in all cases and a random distribution of obstacles.
- Calculating an optimal speed that ensures global convergence towards the goal taking into consideration the shape, kinematics and dynamics of the robot.
- Studying factors that make a path smooth and finding the relation between optimal and smooth paths.
- Finding a procedure for measuring the computational complexity of the proposed method for analyzing gaps to demonstrate its advantage over the one proposed in the ND method.
- It is also of interest to define and prove smoothness by an analytical solution to get a precise comparison between the proposed approaches and the previous methods.

Appendix A

Checking Free-Way to Goal Criterion

This appendix presents an algorithm to check whether a Free-Way to a goal exists or not assuming a circular robot.

Suppose that the plane is divided into four regions as shown in Fig. A.1. These regions are formed from the x and y axes relative to the current robot location.

The *inputs* of the algorithm are as follows:

1) The robot location $(\mathrm{X}_{\mathrm{robot}})$ and robot radius R.
2) The goal position $(\mathrm{X}_{\mathrm{goal}})$.
3) A list (L) of obstacle points, where an obstacle is $(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}})$.

The *output* of the algorithm is a Boolean variable; either (yes) if there is a Free-Way to the goal or (no) if no such way exists.

We first exclude all obstacle points that are behind the robot (i.e. lie on regions R_3 and R_4). Then, we exclude obstacle points that are outside the specified rectangle (shown in Fig. A.1), with length equals to the line connecting the center of the robot with the goal plus the radius of the robot and width equals to the diameter of the robot centered at the X axis relative to the robot. Finally, the algorithm checks if there are obstacles on the specified rectangle or not. The existence of obstacles returns (no) which means no Free-Way to the goal. But, if the area within this rectangle is free from obstacles, then there is a Free-Way to the goal and the output will be (yes). We explain the algorithm in other words as follows.

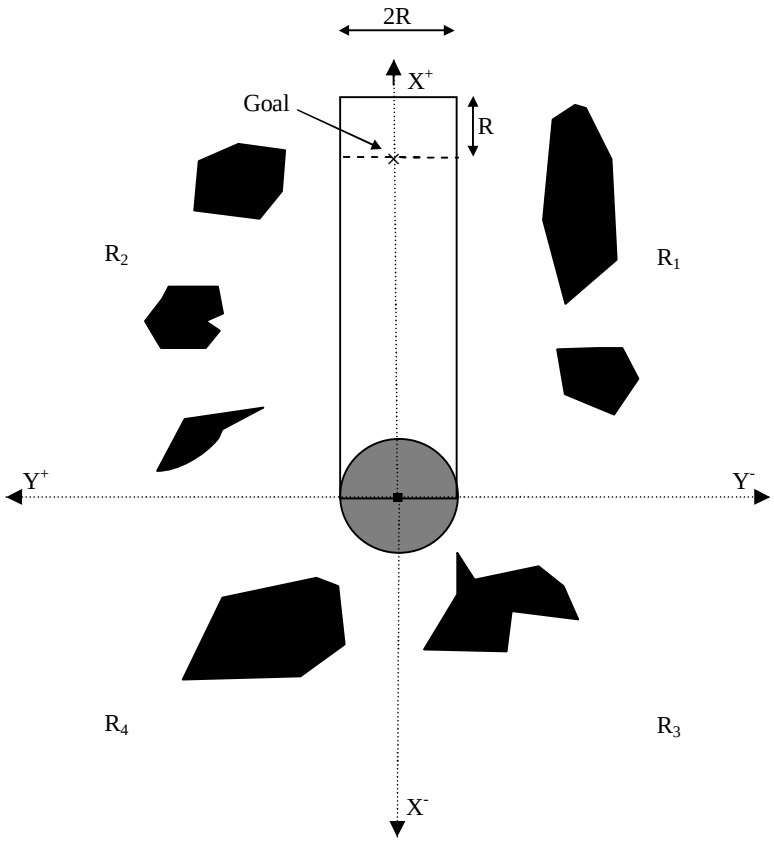


Figure A.1: Checking if there exists a Free-Way to a goal.

Consider the list (L) of the obstacle points. The following is done:

1) Eliminate from (L) every point (i) where:

   (a) $(O_i^L \in R_3)$ or $(O_i^L \in R_4)$;

   (b) $d(O_i^L, X) > R$.

   Where X is the x-axis relative to the robot (see Fig. A.1) and $d(A, B)$ returns the distance between the two points A and B.

2) If $\exists j / d(O_j^L, X_{robot}) < [d(X_{goal}, X_{robot}) + R]$ then return (no), else return (yes).

Appendix B

Accommodating the Rectangular Shape

For the sake of simplicity, the methods developed in this thesis compute collision-free motion commands considering a circular robot shape. In addition, the laser scanner is assumed to be on the center of the robot. But the real robot which we used in carrying out our experiments, as most of the real robots, has a rectangular shape and a laser scanner situated at the front of the robot. The goal of this appendix is to discuss how we adapt the algorithms addressed in this work to deal with this problem.

The shape problem can be noticed in two major aspects; in checking whether a gap is navigable or not (this includes checking discontinuities), and calculating the distance to an obstacle. The following two subsections explain how we solve these two problems.

B.1 Checking Navigability

For checking discontinuities and navigability of a gap, we easily consider that a virtual circle is drawn around the robot where it touches the extreme points on the robot (see Fig B.1). It is then assumed that the width of the robot equals to the diameter of this circle. The gap is navigable if its width is greater than this diameter. If we consider the real width of the robot for checking navigability, the robot may collide with obstacles if it does not move straight forward while entering the gap. In most of the cases, particularly for cluttered environments, this is not ensured and the robot may enter the gap while turning to the left or to the right.

B.2 The Distance to an Obstacle

Calculating the distance to an obstacle in this appendix means finding the distance between this obstacle and the robot boundary. For a circular robot with a laser scanner placed on its

center, the distance can easily be computed by subtracting the robot radius from the distance towards the obstacle that the laser scanner returns. For rectangular robot shapes, the case is more difficult. We developed the following algorithm to deal with these robot shapes.

The *inputs* of the algorithm are:

1. The detected obstacle (O), which has the coordinates (x_o, y_o).
2. The robot width (W_r) and robot length (L_r).
3. The distance from the laser scanner position to the front of the robot (D_s).

The *output* of the algorithm is the distance to the obstacle (D_o).

We first divide the plane into nine regions (1 – 9) as shown in Fig. B.2. This aims to find a single formula for all obstacles positioned in the same region. The algorithm can be summarized in the following two steps:

Step 1: The region that the obstacle belongs to is identified. The following pseudo code shows how we can specify the obstacle region (see Fig. B.2).

Algorithm B.1 Region Identification

```
 1: if (x_o > D_s)  // in front of the robot
 2:     if (y_o > (W_r/2)
 3:        region = 8;
 4:     else if (y_o < (−W_r/2))
 5:        region = 2;
 6:     else
 7:        region = 1;
 8: else if (x_o < −W_r + D_s)  // behind the robot
 9:     if (y_o > (W_r/2)
10:        region = 6;
11:     else if (y_o < (−W_r/2))
12:        region = 4;
13:     else
14:        region = 5;
15: else  // left or right of the robot, or in the robot
16:     if (y_o > (W_r/2)
17:        region = 7;
18:     else if (y_o < (−W_r/2))
19:        region = 3;
20:     else
21:        region = 9;  // in the robot
```

Step 2: The equation that calculates the distance to an obstacle depends on the region specified in step 1. The algorithm for calculating the distance is shown below.

Algorithm B.2 Calculating the Distance to the Obstacle

```
1:  switch (region)
2:  {
3:      case 1:
4:          Do = xo − Ds;
5:          breake;
6:      case 2:
7:      case 8:
8:          Do = sqrt((xo − Ds)² + (|yo| − Wr/2)²);
9:          breake;
10:     case 3:
11:     case 7:
12:         Do = |yo| − Wr/2;
13:         breake;
14:     case 4:
15:     case 6:
16:         Do = sqrt((xo + (Lr − Ds))² + (|yo| − Wr/2)²);
17:         breake;
18:     case 5:
19:         Do = |xo + (Lr − Ds)|;
20:         breake;
21:     default:  // in the robot (case 9)
22:         Do = 0;
23: }
```

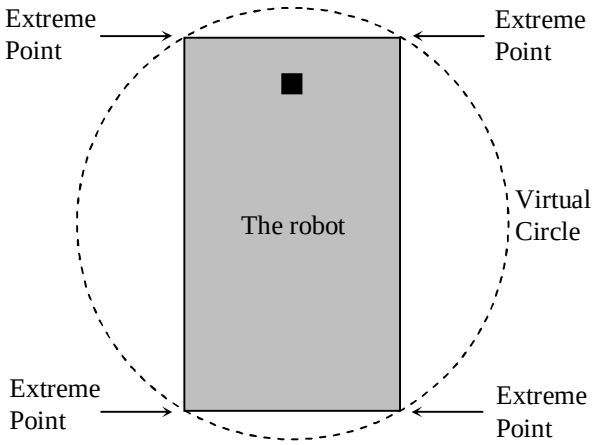


Figure B.1: Drawing a circle around the robot which touches its extreme points. The width of the gap is considered to be the diameter of this circle when checking the navigability of a gap.

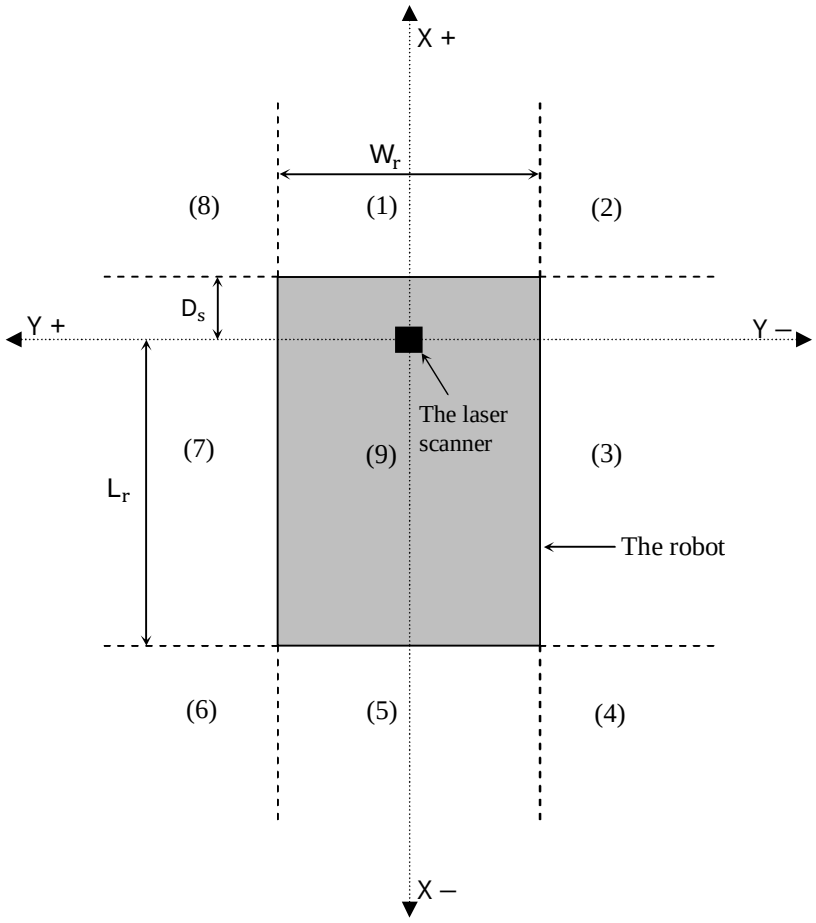


Figure B.2: Dividing the plane around the robot into 9 regions. Region 1 is to the front of the robot, region 2 to the front right corner of the robot, region 3 to the right of the robot, region 4 to the right back corner of the robot, region 5 behind the robot, region 6 to the back left corner of the robot, region 7 to the left of the robot, region 8 to the left front corner of the robot, and region 9 in the robot.

Bibliography

[1] Saurabh Sarkar, “Path planning and obstacle avoidance in mobile robots”, Master Thesis, Division of Research and Advanced Studies of the University of Cincinnati, 2007.

[2] P. Cao, X. Liao, and E. L. Hall, "Motion Control Design of the Bearcat II Mobile Robot," *SPIE, Intelligent Robots and Computer Vision XVIII: Algorithms, Techniques, and Active Vision*, vol. 3837, pp. 118-126, 1999.

[3] J. L. Fuller, “Robotics: Introduction, Programming, and Projects”, *Prentice Hall*, 1999.

[4] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Stanley: The robot that won the DARPA Grand Challenge,” *Journal of Robotic Systems*, vol. 23, no. 9, pp. 661–692, 2006.

[5] S. Hirose and E. Fukushima, “Snakes and Strings: New Robotic Components for Rescue Operations,” *Springer Tracts in Advanced Robotics*, Experimental Robotics VIII, volume 5, pp. 48-61, 2003.

[6] B. A. Maxwell, W. Smart, A. Jacoff, J. Casper, B. Weiss, J. Scholtz, H. Yanco, M. Micire, A. Stroupe, D. Stormont, and T. Lauwers, “2003 AAAI robot competition and exhibition,” *AI Magazine*, vol. 25, no. 2, pp. 68–80, 2004.

[7] A. Sanchez, X. Hernandez, O. Torres and Alfredo Toriz P., “Mobile robots navigation in industrial environments”, *2009 Mexican International Conference on Computer Science*, pp.155-166, 2009.

[8] Erwin Prassler, Arno Ritter, Christoph Schaeffer and Paolo Fiorini, “A Short History of Cleaning Robots”, Autonomous Robots, *Volume 9, Number 3, pp. 211-226, Springer Netherlands*, December, 2000.

[9] "Robot assisted surgery: da Vinci Surgical System", *Brown University Division of Biology and Medicine*, http://biomed.brown.edu/Courses/BI108/BI108_2005_Groups/04/davinci.html, 25 July 2009.

[10] Graham, Stephen, "America's robot army", *New Statesman*, http://www.newstatesman.com/2006a06120018, 12 June 2006.

[11] "Battlefield Robots: to Iraq, and Beyond", *Defense Industry Daily*, http://www.defenseindustrydaily.com/battlefield-robots-to-iraq-and-beyond-0727, 20 June 2005.

[12] D. Caltabiano & G. Muscato, “A Robotic System for Volcano Exploration”, *Cutting Edge Robotics*, ISBN 3-86611-038-3, pp. 784, Germany, July 2005.

[13] “Robots - our helpers in space”, *European Space Agency*, http://www.esa.int/esaCP/SEM8XXWJD1E_index_0.html, 29 November 2004.

[14] Steven M. LaValle, “Planning Algorithms,” *Cambridge University Press*, 2006.

[15] J. Minguez and L. Montano, “Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios,” *IEEE Transactions on Robotics and Automation*, Vol. 20, Issue 1, pp. 45-59, Feb. 2004.

[16] Joseph W. Durham and Francesco Bullo, “Smooth Nearness-Diagram Navigation,” *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pp. 690-695, 22-26 Sept. 2008.

[17] A. Ferreira, M. Sarcinelli-Filho, and T. F. Bastos-Filho, "A new approach to avoid obstacles in mobile robot navigation: Tangential escape," in *Proc. of the Second International Conference on Informatics in Control*, Automation and Robotics, ICINCO'05, vol. 3, pp. 341-346, Barcelona, Spain, September 2005.

[18] E. D. Dickmanns and N. Muller, “Scene Recognition and Navigation Capabilities for Lane Changes and Turns in Vision-Based Vehicle Guidance”, *Elsevier Science Ltd*, Vol. 4, No. 5, pp. 589-599, Neubiberg, Germany, 1996.

[19] Amit Singhal, “Issues in Autonomous Mobile Robot Navigation”, *Computer Science Department of Rochester*, May, 1997.

[20] J. Leanard and H.F. Durrant-Whyte, “Mobile Robot Localization by Tracking Geometric Beacons”, *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3,1991, pp. 376-382.

[21] W. Burgard, D. Fox and S. Thrun, “Active Mobile Robot Localization”, in *15th International Joint Conference on Artificial Intelligence*, pp. 1346-1352, Nagoya, Japan, 1997.

[22] Roland Siegwart and Illah R. Nourbakhsh “Introduction to Autonomous Mobile Robots”, *the MIT Press*, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2004.

[23] Anders Oreback, “A Component Framework for Autonomous Mobile Robots”, Doctoral Thesis, Stockholm, Sweden, 2004.

[24] Robert Holmberg and Oussama Khatib, “Development of a Holonomic Mobile Robot for Mobile Manipulation Tasks”, in *FSR’99 International Conference on Field and Service Robotics*, Stanford University, Stanford, USA, 1999.

[25] J. Minguez, F. Lamiraux and J. P. Laumond, “Motion Planning and Obstacle Avoidance/Springer Handbook of Robotics”, *Springer Berlin Heidelberg*, 2008.

[26] “Non-holonomic Constraints and Lie brackets”, *School of Computer Science*, Carnegie Mellon University, California.
http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/ppp/Non_holonomic.pdf, 10 March 2010.

[27] Muhammet Bastan, “Visula Servoing Of Mobile Robots Using Potential Fields”, Master thesis, Sabanci University, Spring 2004.

[28] J. Lobo, L. Marques, J. Dias, U. Nunes and A. T. de Almeida, “Sensors for Mobile Robot Navigation/Autonomous Robotic Systems”, *Springer Berlin / Heidelberg*, pp. 50-81, Berlin, 1998.

[29] “Dead Reckoning”, *Bowditch -The American Practical Navigator*, http://www.irbs.com/bowditch/pdf/chapt07.pdf, 30 February 2010.

[30] “Laser Range Finder Overview”, *Acroname Robotics*, http://www.acroname.com/robotics/info/articles/laser/laser.html, 5 October 2009.

[31] Mehmet Serdar Guzel, “Mobile Robot Navigation using a Vision Based Approach”, *School of Mechanical and Systems Engineering*, Newcastle University.

[32] G. N. DeSouza and A. C. Kak, “Vision for Mobile Robot Navigation: A Survey”, in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, No. 2, pp.237-267, February 2002.

[33] B.P. Gerkey, et al., “Most Valuable Player*:* A Robot Device Server for Distributed Control”, *Proceedings of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2001: p. 1226-1231.

[34] P. Nattharith, “Mobile Robot Navigation using a Behavioral Strategy”, *School of Mechanical and Systems Engineering*, Newcastle University, Newcastle, 2008.

[35] T. H.J. Collett, B.A. MacDonald, and B. Gerkey, “Player 2.0: Toward a Practical Robot Programming Framework”, *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, 2005.

[36] J. C. Latombe, “Robot Motion Planning”, *Norwell*, MA: Kluwer, 1991.

[37] Maria Isabel Ribeiro, “Obstacle Avoidance”, Navigation/Collision Avoidance, November 2005.

[38] Daniele Calisi, “Motion Planning, Reactive Methods, and Learning Techniques for Mobile Robot Navigation”, Sapienza, Roma, September 2008

[39] V. Kunchev, L. Jain and V. Ivancevic, “Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review”, University of South Australia, Australia, *Springer-Verlag Berlin Heidelberg*, 2006.

[40] J. J.A.M. Keij, “Obstacle Avoidance for Wheeled Mobile Robotic Systems (Literature Exploration)”, University of Technology, Eindhoven, February 2003.

[41] Javier Minguez and Luis Montano, “Robot Navigation In Very Complex, Dense, and Cluttered Indoor/Outdoor Environments”, in *the 15th Triennial World Congress*, Barcelona, Spain, 2002.

[42] SICK AG, “Proximity Laser Scanner (PLS)”, *Technical Description*, Germany, 2000.

[43] Javier M. Zafra, “Robot Shape, Kinematics, and Dynamics in Sensor-Based Motion Planning”, Ph.D. Dissertation, University of Zaragoza, Spain, July 2002.

[44] V. J. Lumelsky and A. A. Stepanov, “Dynamic path planning for a mobile automaton with limited information on the environment,” *IEEE Trans. Automatic Control,* 31(11): 1058-1063, 1986.

[45] V. J. Lumelsky and T. Skewis, “Incorporating range sensing in the robot navigation function,” *IEEE Trans. Sys., Man Cybernet,* 20(5): 1058-1069, 1990.

[46] V. Lumelsky and Stepanov, “Path-planning srategies for a point mobile automaton amidst unknown obstacles of arbitrary shape,” *in Autonomous Robots Vehicles, I.J. Cox, G.T. Wilfong (Eds), New York, Springer*, pp. 1058 – 1068, 1990.

[47] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, “Principles of Robot Motion: Theory, Algorithms, and Implementation”, September, 2007.

[48] A. Yufka and O. Parlaktun, "Performance Comparison of Bug Algorithms for Mobile Robots", *5th International Advanced Technologies Symposium (IATS'09),* Karabuk, Turkey, May 2009.

[49] Kamon, I., Rivlin, E., Rimon, E., "A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots," *in Proceedings of the IEEE International Conference on Robotics and Automation,* Minneapolis, April 1996.

[50] O. Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", *he Intl. Journal of Robotics Research*, vol. 5, pp. 90- 98, 1986.

[51] Ganesh Swaminathan, "Robot Motion Planning", *School of Engineering Science*, Simon Fraser University, Canada, February 27, 2006.

[52] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1398–1404, Sacramento, CA, 1991.

[53] J. Barraquand and J. C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom", In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1712-1717, 1990.

[54] R. B. Tilove, "Local obstacle avoidance for mobile robots based on the method of artificial potentials," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, Cincinnati, OH, 1990, pp. 566–571.

[55] M. Khatib, "Sensor-based motion control for mobile robots", Ph.D. dissertation, LAAS-CNRS, Toulouse, France, 1996.

[56] M. Khatib and R. Chatila, "An Extended Potential Field Approach for Mobile Robot Sensor-based Motions", *In Intl. Conf. On Intelligent Autonomous Systems IAS'4*, (1995).

[57] B. H. Krogh and C. E. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, 1986, pp. 1664–1669.

[58] L. Montano and J. Asensio, “Real-time robot navigation in unstructured environments using a 3Dlaser rangefinder,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, vol. 2, Grenoble, France, 1997, pp. 526–532.

[59] J. Borenstein, Y. Koren, “Real-time obstacle avoidance for fast mobile robots in cluttered environments”**,** *Proceedings of IEEE International Conference on Robotics and Automation*, 1990, Pages: 572 – 577, Vol.1.

[60] J. Borenstein and Y. Koren, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1187, May 1989.

[61] J. Borenstein and Y. Koren, “The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots”, *IEEE Transactions on Robotics and Automation*, pp.278-288, 7(3), June (1991).

[62] Elfes, A., "A Sonar-Based Mapping and Navigation System." *Carnegie-Mellon University*, The Robotics Institute, Technical Report, 1985, pp. 25-30.

[63] Moravec, H. P. and Elfes, A., "High Resolution Maps from Wide Angle Sonar." *IEEE Conference on Robotics and Automation*, Washington, D.C., 1985, pp. 116-121.

[64] Manz, A., Liscano, R., and Green, D.A., "A Comparison of Realtime Obstacle Avoidance Methods for Mobile Robots", *Experimental Robotics*, Toulouse, France, June 1991.

[65] I. Ulrich and J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots." *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1572-1577, Leuven, Belgium, May 1998.

[66] I. Ulrich and J. Borenstein, "VFH*: Local Obstacle Avoidance with Look-Ahead Verification." *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2505-2511, San Francisco, California, Apr. 2000.

[67] Khatib, O., Quinlan, S., “Elastic Bands: Connecting, Path Planning and Control,” *in Proceedings of IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[68] Ashish B. Shah, “an Obstacle Avoidance Strategy for the 2007 Darpa Urban Challenge”, Master Thesis, the Ohio State University, 2008.

[69] Khatib, M., Jaouni, H., Chatila, R., Laumod, J.P., “Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots,” *in Proceedings of IEEE International Conference on Robotics and Automation*, Albuquerque, NM, April 1997.

[70] R. Simmons, “The Curvature-Velocity Method for Local Obstacle Avoidance”, In *IEEE Intl. Conf. on Robotics and Automation ICRA’96*, pp. 2275-2282, Minneapolis, April 1996.

[71] Roland Philippsen, “Motion Planning and Obstacle Avoidance for Mobile Robots in Highly Cluttered Dynamic Environments”, Ph.D. Thesis, Lausanne, EPFL, 2004.

[72] Stephane R. Petti, “Safe Navigation within Dynamic Environments: A Partial Motion Planning Approach”, Ph.D. thesis, Paris, July 2007.

[73] Simmons R., Nak Young Ko, "The Lane-Curvature Method for Local Obstacle Avoidance," *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, Victoria B.C., Canada, 1998.

[74] D. Fox, W. Burgard. and S. Thrun, “The Dynamic Window Approach to Collision Avoidance”, *IEEE Robotics and Automation Magazine*, 4(1), pp. 23-33, March 1997.

[75] J. Minguez, L. Montano, T. Simeon, and R. Alami, “Global Nearness Diagram Navigation (GND)”,in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[76] O. Brock and O. Khatib, “High-Speed Navigation Using the Global Window Approach”, *in IEEE Intl. Conf. on Robotics and Automation ICRA’99*, Detroid, Michigan, pp. 341-346, May (1999).

[77] R. C. Arkin, “Behavior-Based Robotics”, Cambridge, *MA: MIT Press*, 1999.

[78] J. Minguez, J. Osuna, and L. Montano, “A ”divide and conquer” strategy based on situations to achieve reactive collision avoidance in troublesome scenarios,” in *IEEE Int. Conf. on Robotics and Automation*, (New Orleans, LA), pp. 3855–3862, Apr. 2004.

[79] Minguez, J., Montano, L., “Robot Navigation in Very Complex, Dense, and Cluttered Indoor / Outdoor Environments,” *in Proceeding of International Federation of Automatic Control (IFAC2002)*, Barcelona, April 2002.

[80] Thierry Fraichard, “A Short Report about Motion Safety”, *INRIA*, No. 5987, September 27, 2006.

[81] Y. Lin, C. Chou and F. Lian, “Indoor Robot Navigation Based on DWA*: Velocity Space Approach with Region Analysis,” in *ICROS-SICE International Joint Conference 2009*, pp. 700-705, 18-21 Aug. 2009.

[82] R. Siegwart and I. Nourbakhsh, “Autonomous Mobile Robots, Chapter 6”, http://ai.cs.umbc.edu/~oates/classes/2010/Robotics/Slides6.pdf, 19 January 2010.

[83] B. Gerkey and contributors, "Player/Stage/Gazebo Project,” http://playerstage.sourceforge.net/doc/Player-2.1.0/player, Sept. 2005, Version 2.03, 15 November 2009.

[84] A. Ferreira, F. G. Pereira, T. F. B.-Filho, M. S.-Filho and R. Carellit, “Avoiding Obstacles in Mobile Robot Navigation: Implementing the Tangential Escape Approach”, in *IEEE ISIE 2006*, Montreal, Canada, July 2006.

[85] H. I. Christensen, “Robot Kinematics,” Centre for Autonomous Systems, Kungl Tekniska Högskolan, 2006.

[86] Sven Bottcher, “Principles of robot locomotion”, Seminar, Department of Computer Science **,** Southern Illinois University at Carbondale, 2009.