NTNU

Norwegian University of
Science and Technology

Reactive Collision Avoidance

Safe navigation in a dynamic cluttered
environment

Andreas L. Aarvold

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Kristin Ytterstad Pettersen, ITK
Co-supervisor: Martin Syre Wiig, FFI

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem description

Autonomous vehicles are increasingly used in both scientific and commercial applications. During autonomous or semi-autonomous operations, the capability to avoid static and dynamic obstacles without human intervention is crucial for mission success and vehicle safety. In complex environments with dynamic obstacles the vehicle has to react quickly to obstacles, which can make the time consumption of motion planning algorithms unacceptable. Hence, there is a need for reactive collision avoidance algorithm for avoiding moving obstacles.

Many vehicles, such as cars, can be modeled as a vehicle with nonholonomic constraints, i.e. they can move forwards and turn, but not move sideways. Such a model can also be used as a simplified model of a ship or a fixed wing aircraft, however these vehicle have a limited speed envelope and may have significant constraints on the forward acceleration due to a high mass.

The goal of this master assignment is to develop a reactive collision avoidance algorithm for a vehicle with nonholonomic constraints and limited forward acceleration. The algorithm should be combined with a classical guidance law for target reaching or path following, and the system should make the vehicle able to safely traverse a crowded environment with multiple obstacles. The performance of the algorithm should be analyzed, and it should be tested in simulations.

This master assignment builds on a project assignment where the following subtasks were achieved:

- Perform a literature study on reactive collision avoidance algorithms for mobile robots
- Develop a reactive collision avoidance algorithm, or modify an existing algorithm, suitable for a nonholonomic vehicle with velocity constraints and limited forward acceleration
- Perform an analysis of the system when applied to such a vehicle in an environment with at least one obstacle
- Implement the algorithm in a simulation environment
- Verify the performance of the collision avoidance algorithm in the presence of multiple obstacles

The main topic of the master assignment will be to extend the algorithm from the project assignment to a multi-agent scenario. Each agent should be modeled like a unicycle, and should employ the same collision avoidance algorithm. The following subtasks are proposed:

- Perform a literature survey on collision avoidance algorithm in multiagent scenarios
- Modify the algorithm from the project assignment to work in such a scenario, if necessary
- Analyze the resulting algorithm to find conditions under which safety are guaranteed
- Verity the analysis in simulations
- Implement the Reciprocal Velocity Obstacles algorithm and compare simulation results

Abstract

Collision-free navigation in unknown environments is an essential quality for any autonomous vehicle. In this thesis, a reactive collision avoidance algorithm is presented for vehicles constrained by a unicycle nonholonomic model in a multi-agent environment. The agents navigate independently in a decentralized manner, without explicit communication. Restricted forward speed makes the model suitable for vehicles with heavy linear constraints such as marine vessels and unmanned aircraft. The sensor model is given by an integrated representation of the environment where only limited sensing is required. A new *braking rule* is created to cope with typical multi-agent challenges such as oscillation and deadlocks.

Through rigorous mathematical analysis, sufficient conditions for collision-free navigation is derived by reducing the number of agents. Tests, simulating thousands of cluttered environments, is presented including scenarios with both multiple agents and passive obstacles. The simulations prove that agents safely navigates the environment even when ignoring the strict conditions made in the mathematical analysis. Furthermore, the algorithm shows promising results when compared to other well known multi-agent reactive algorithms, such as the Reciprocal Velocity Obstacles.

The main contribution of this thesis is a computational efficient reactive algorithm suited for a wide range of vehicles. By merging two existing algorithms and adding a new breaking rule, the result is a fast and safe multi-agent algorithm. In addition, a literature review is carried out to investigate alternative approaches to collision-free navigation and present the most relevant prior research in the field.

Sammendrag

Kollisjonsfri navigering i ukjente miljøer er essensielt for autonome kjøretøy. Denne avhandlingen presenterer en reaktiv antikollisjons-algoritme for agenter modellert som ett-hjuls kjøretøy med ikke-holonomiske begrensninger. Agentene er desentralisert og styres individuelt av samme algoritme, uten kommunikasjon. Begrenset lineær fart gjør modellen egnet for kjøretøy som biler, ubemannede fly og skip. Sensormodellen er gitt av en integrert representasjon av miljøet rundt en agent og krever begrenset målingsdata. En ny *bremseregel* tar hånd om typiske utfordringer relatert til fler-agent situasjoner, som oscillering og deadlocks.

Gjennom robust matematisk analyse er det lagt til grunn tilstrekkelige betingelser som garanterer kollisjonsfri navigasjon. Det er presentert tester med mange agenter og passive hindringer i tusenvis av tette simuleringer. Testene viser at agenter trygt navigerer simuleringene selv uten å ta hensyn til betingelsene slått fast i den matematiske analysen. Algoritmen viser lovende resultater når den sammenlignes med andre kjente reaktive algoritmer, som Reciprocal Velocity Obstacles.

Hovedbidraget til denne avhandlingen er en reaktiv algoritme som er beregningsmessig effektiv og aktuell for en stor mengde kjøretøy. Resultatet av å sammenslå to eksisterende algoritmer og legge til en ny bremse-lov, er en rask og trygg algoritme tilegnet miljøer med mange agenter. Det er ytterligere utført et litteraturstudie som undersøker tidligere forskning på området og alternative antikollisjonsalgoritmer.

Preface

This thesis concludes my Master of Science degree from the Department of Engineering Cybernetic at the Norwegian University of Science and Technology. Researching autonomous vehicles has been an inspiring undertaking. As a technology enthusiast, autonomous vehicles intrigues me and relies on technology that I find truly fascinating. I would like to use the opportunity thank my supervisors, PhD candidate Martin Syre Wiig and Professor Kristin Ytterstad Pettersen. Their support throughout this project has guided me on the right path and steered me away from possible collisions. The background material provided prior to writing this thesis is described fully in Section 1.4. Finally, I would like to thank my friends and fellow students for the time together at NTNU. It has been five years with a lot of fun and invaluable experiences.

Andreas L. Aarvold
Trondheim 04-06-2018

Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

IEA	Integrated Environment algorithm
PA	Proposed Algorithm
RVO	Reciprocal Velocity Obstacles

Variables

α	Angle of emitted sensor rays, defining where obstacles are detected
$\hat{E}(t)$	Set of impenetrable obstacles expanded with a safety distance $d_{safe,i}$
$\mathcal{D}_i(t)$	Sensor disk in front of an agent
$\psi_i(t)$	Agent heading
$\psi_{\tau,i}(t)$	Angle to agent's target
$\psi_{d,i}(t)$	Desired agent heading in guidance mode
$a_i(t)$	Agent linear acceleration bounded by $[-a_{max,i}, a_{max,i}]$
$C_i(t)$	Desired direction in collision avoidance mode
$E(t)$	Set of impenetrable obstacles

$r_i(t)$ Agent heading rate, bounded by $[-r_{max,i}, r_{max,i}]$

$v_i(t)$ Vehicle forward speed, bounded by $[v_{min,i}, v_{max,i}]$

$v_{\hat{E}_i}(t)$ Obstacle forward speed

$x_i(t),\ y_i(t)$ Agent Cartesian coordinates, with position $\boldsymbol{p}_i = [x_i(t),\ y_i(t)]$

Constants

δ Discrete time step

τ_i Agent target with radius $R_{\tau,i}$ and position $\boldsymbol{p}_{\tau,i} = [x_{\tau,i},\ y_{\tau,i}]$

$a_{max,i}$ Maximum agent linear acceleration

$d_{safe,i}$ Safety distance to account for agent size and other safety measures

$r_{max,i}$ Maximum agent heading rate

$R_{r,i}$ Agent radius

$R_{turn,i}$ Agent minimum turning radius

T_{brake} Minimum amount of time the braking rule is active

$v_{max,i}$ Agent maximum forward speed

$v_{min,i}$ Agent minimum forward speed

Functions

$M_i(\alpha, t)$ Binary function representing obstacles inside the sensor disk $\mathcal{D}_i(t)$

$\hat{M}_i(\hat{\alpha}, t)$ Analog to $M_i(\alpha, t)$ with obstacle velocity compensation

$\hat{m}_i(t)$ Binary function indicating if one or more obstacles are detected

$b_i(t)$ Binary function indicating that the braking rule is active

Chapter 1

Introduction

1.1 Motivation

In today's society autonomy can be found in self-driving cars, unmanned aerial vehicles(UAV), unmanned maritime vessels and much more. It is fair to say that autonomous vehicles are increasingly important in everyday life. The upsides of autonomous vehicles are the opportunities to operate with increased efficiency and safety while reducing cost and exposure. In environments unsuitable to humans, for example in radiation-exposed areas, minefields or on the moons of Saturn, the advantage of unmanned vehicles are unprecedented. More practical examples are given in [1], [2], [3] and [4].

To ensure safe navigation and application of autonomous vehicles, the ability to successfully perform collision avoidance is of essential importance. Autonomous collision avoidance has been, and still is, a major field of research in the scientific community. There are numerous different approaches to collision avoidance, normally they are divided into two categories: *motion planning* and *reactive* collision avoidance. A motion planning algorithm use information about the environment to calculate a desired path and a reactive algorithm maps sensor input directly to control output,

making reactive methods less computationally complex.

This thesis focuses on the problem of collision avoidance in a decentralized homogeneous multi-agent system. The algorithm proposed is local and reactive, where limited computation effort and sensing are of interest. There is no communication between agents and all computations are done locally. This kind of algorithm is well suited for real-time application for vehicles with limited computational capacity or operating in rapidly changing environments.

The improvement in sensing and computation abilities has in recent years led to reactive collision avoidance being especially interesting. With improved accuracy, sampling rate and computational power, vehicles have a better understanding of their surroundings and are capable of finding safe control inputs within a reasonable time. As the literature review shows, previous methods are often restricted both by available computationally power and sensor data. Reactive collision avoidance is an important part of a safe navigation system and can be combined with a more computationally expensive path planning algorithm to form a robust strategy. By doing so the complete navigation system becomes both safer and more efficient compared to using either method alone.

Rigorous mathematical analysis is important to classify safety and reliability of collision avoidance algorithms. In order to mathematically prove safe navigation, several assumptions and restrictions have to be made. These restrictions are often strict and thus simulations and experimental testing are important additions to present a wider view on the algorithms' performance.

1.2 Literature review

Safe navigation of autonomous vehicles remains an active research field. The primary objectives are related to collision avoidance, target reachability, multi-vehicle coordination or a combination of the three. In order to achieve these goals, numerous methods

have been suggested. Although the main focus of this thesis is on local multi-agent reactive navigation, for completeness the review also considers alternative methods and prior research. A survey focusing on local navigation for mobile robots in cluttered environments is given in [5].

The alternative to reactive collision avoidance is path planning algorithms. In a path planning approach, the vehicle uses knowledge about its environment to calculate the most desirable path. Most commonly, such algorithms require more information about the environment than a reactive method, but that need not be the case. In [6], [7] and [8] optimal paths are found through a static environment assuming it is a priori known. Unfortunately, the path calculation tends to be computationally complex. In fact, the more general motion planning problem for multiple agents with bounded velocities has been shown by [9] to be NP-hard.

Another interesting and more recent approach in this category is the *model predictive control* (MPC) [10]. A path for the next N time steps is calculated at each time instant, taking the vehicle model into account. According to [5], MPC has several favorable properties compared to the commonly used *artificial potential fields* and *velocity object* based methods, which are generally more conservative when extended to higher order vehicle models. MPC has mostly been applied to scenarios where the environment is known [11], but can also be used in a priori unknown environments as in [12] and [13]. In general, MPC has the advantage of considering a more optimal path, but on the other hand, tend to be more computationally complex and lacking rigorous mathematical analysis. Without higher order vehicle models, the advantages of using MPC abates.

A reactive approach can be seen as a memory-less mapping between sensor input and control output, and thus tend to be computationally effective. A popular reactive approach is the *artificial potential field* [14], which works by creating artificial fields where the target has an attracting ability and obstacles are repulsive. It has some stability problems [15], that have been solved by [16] and [17]. A polar histogram

of merged sensor measurements serves as the solution in [17]. By choosing a new desired velocity using the polar histogram, the stability problem vanishes. Only a single, holonomic vehicle and static obstacles are considered. The *dynamic window* [18] can be regarded as an MPC algorithm with a prediction horizon of a single time step and handles nonholonomic constraints by choosing a safe direction among a set of valid vehicle trajectories, also only for static environments. The *collision cone* [19] and *velocity obstacle* [20] algorithms on the other hand deals with dynamic obstacles, but without considering nonholonomic constraints.

The above-mentioned algorithms suffer from different undesirable limitations such as stability, and many do not consider dynamic obstacles or nonholonomic constraints. Some of the older papers focus on noisy and inaccurate sensor measurements as a major limitation. However, with improved sensors and pre-processing of sensor data that need not be the main focus. Thus there is room for new and smarter ways to achieve reactive collision avoidance.

The algorithm proposed in this thesis builds on the algorithm presented in [21]. Resembling the *vector field histogram* from [17], the algorithm is reactive and local. Additionally, it accounts for both moving and deformable objects in an unknown environment as well as a nonholonomic vehicle with constant forward speed. An integrated representation of the local environment together with bearing to the target provides the only required measurements. The algorithm works by steering the vehicle towards a collision-free zone on a sensor disk in front of the vehicle. According to the simulations, the algorithm safely navigates cluttered environments with multiple moving obstacles. For this reason, along with the interesting use of an integrated sensor representation, the algorithm from [21] serves as a starting point for the algorithm proposed in this thesis. Nevertheless, by not accounting for obstacle velocity, the algorithm can be conservative and less effective dealing with moving obstacles.

Another interesting approach to reactive collision avoidance can be found by inspiration from the world of biology. The algorithm presented in [22] is one example.

Here, dynamic obstacles are overcome by providing a navigation strategy that switches between moving to the target along straight lines, and a sliding-mode obstacle avoidance law. The sliding-mode controller aims at maintaining a constant avoidance angle between the vehicle's heading and the tangent from the vehicle to the obstacle. The paper does, however, allow varying vehicle forward speed without hard constraints, resulting in a behavior where in some scenarios the vehicle completely stops in order to maintain safe navigation. Working with aircraft or marine vessels, having large inertia which restricts linear acceleration, such behavior may not be feasible. The algorithm proposed by [23] builds on [22] and incorporates constant forward speed and a nonholonomic vehicle model. Decoupled from the vehicle speed, the heading is the only control input. In addition [23] contributes with a new method to compensate for obstacle velocities, which is discussed in detail in later chapters.

In recent years multi-agent systems have become a growing research topic, both due to relevant applications such as highway maneuvering, aircraft traffic control and modeling pedestrians, and to the price reduction in swarm-like vehicles such as UAVs. There are many challenges related to multi-agent systems and different ways to achieve safe navigation. In this thesis, we consider an arbitrary number of homogeneous agents guided by the same collision avoidance algorithm. This creates an environment well suited for a local reactive collision avoidance algorithm. A well-known paper on the topic is *Reciprocal Velocity Obstacles* [24], which expands [20] to decentralized multi-agent control. The basic idea is to share the responsibility of the collision avoidance maneuver between the conflicting agents. This concept has been used in several other multi-agent papers such as [25], [26] and [27]. A challenge occurs when agents encounter passive moving obstacles that are not guided by the same collision avoidance algorithm. One solution to the challenge is to prioritize agents and obstacles, such that agents take full responsibility on an encounter with passive obstacles. An obvious requirement is that agents are able to distinguish between obstacles and other agents. To avoid this challenge, the algorithm presented in this thesis treats all conflicts equal, agents or obstacles. Another interesting method is given in [28], where the algorithm requires initial maneuvering to ensure no agents

are in conflict, then guarantees that all agents avoid a state of conflict.

Deadlocks and oscillations have proven to be considerable problems in multi-agent systems. "*Rules of the Road*" is a common concept in multi-agent control to overcome deadlocks. A set of rules are defined that apply to all agents. When a conflict arises between agents the rules decide which agent has to yield and which to pass. An interesting example is given in [29], here the International Regulations for Preventing Collisions at Sea, known as COLREGS, are applied to maritime autonomous navigation guided by *Velocity Obstacles*. Building on the Rule of the Roads concept, the algorithm presented in this thesis applies a conditioned yield-pass *braking rule*.

In an alternative approach to multi-agent control, vehicles share state information to provide safe navigation. Sharing information can be done through inter-agent communication or by a centralized entity. Examples are given in [30], [31] and [32]. Unfortunately in a real-world scenario, one can not always assume communication with other vehicles and thus making this kind of algorithms unsuitable for a general reactive approach.

In summary, there exist numerous different approaches to collision avoidance. The methods differ in constraints, assumptions and environments. Even though many of the above methods have important contributions, there is still room for improvements. The main contribution of this thesis is a computational effective reactive algorithm for a multi-agent environment where the agents are restricted by nonholonomic constraints with bounded forward speed. The algorithm deals with moving obstacles, without being too conservative, in a cluttered environment. Building on the work done in [21], the integrated environment representation is used. Nevertheless, agents are not restricted to constant forward speed, providing more flexibility without the risk of a complete stop. Further, the algorithm is expanded by the velocity compensation method presented in [23], making it safer in a dynamic environment. A simple *Rule of the road* is implemented as a *braking rule* to avoid deadlocks and increase safety. The thesis presents rigorous mathematical analysis supported by a wide set of simulated scenarios, where Monte Carlo experiments are central.

1.2.1 Reciprocal Velocity Obstacles

This subsection highlights one of the commonly used algorithms for multi-agent environments, namely the classic Reciprocal Velocity Obstacles, RVO, presented in [24]. Due to its widespread usage and the fact that the algorithm is constructed for similar environments as the algorithm presented in this thesis, RVO will be used as a basis for performance comparison in chapter 5. To better understand the reason behind choosing RVO as the benchmark algorithm and the arguments made in the coming chapters, this subsection will have a more in-depth description of RVO.

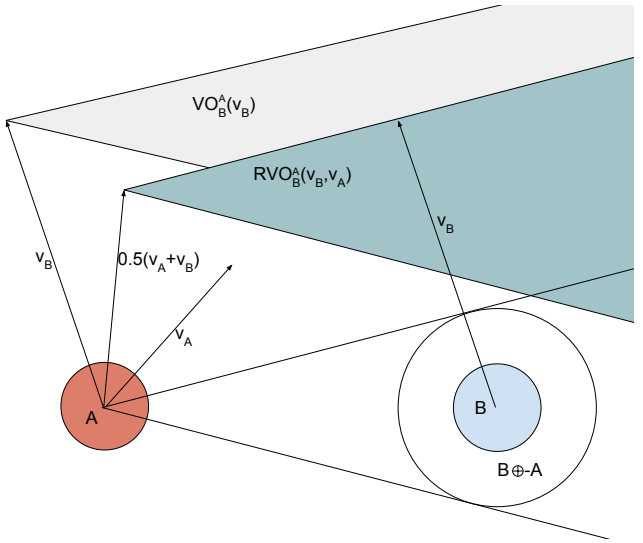


Figure 1.1: Both the Velocity Obstacle $VO^A_B(\boldsymbol{v}_B)$ and the Reciprocal Velocity Obstacle $RVO^A_B(\boldsymbol{v}_A, \boldsymbol{v}_B)$ from agent B to agent A. The figure is inspired by the original RVO paper [24]

RVO is an extension of Velocity Obstacles (VO) introduced in [20]. While RVO inherent the appealing properties of VO, it is extended to resolve the common oscillation problem related to multi-agent navigation. The velocity obstacle is described as follows: Let A be a moving agent in a 2D plane with position $\boldsymbol{p}_A$ and velocity $\boldsymbol{v}_A$. Likewise, let

B be a planar moving agent with position $\boldsymbol{p}_B$ and velocity $\boldsymbol{v}_B$. The velocity obstacle $VO_B^A(\boldsymbol{v}_B)$ from B to A is then described by all velocities $\boldsymbol{v}_A$ that at some point in time will result in a collision between A and B. The velocity obstacle is easier understood geometrically and is illustrated in Figure 1.1, where the light-grayed cone with apex in $\boldsymbol{p}_A + \boldsymbol{v}_B$ is the velocity obstacles from B to A. By choosing a velocity outside the union of all velocity obstacles from all agents to A, agents A is guaranteed collision-free navigation. The oscillation problem occurs when two agents, A and B, both have their current velocity inside the velocity obstacle resulting from the opposing agent. Both change their velocity to avoid a collision, resulting in new velocities outside the velocity obstacles. In the next time instance, both agents will be free to choose new velocities guiding the agents towards their target. If, in the next time instance, the new velocities are inside the velocity obstacle of the opposing agent the problem repeats, resulting in oscillatory behavior.

The extension from VO to RVO is simple. Instead of choosing a new velocity for each agent that is outside the other agent's velocity obstacle, the agent chooses a new velocity that is the weighted average of its current velocity and a velocity that lies outside the other agent's velocity obstacle. The geometrically understanding of RVO can be seen as the green cone in Figure 1.1, where the apex of the cone is moved to $\boldsymbol{p}_A + (1 - \alpha)\boldsymbol{v}_A + \alpha\boldsymbol{v}_B$, where $\alpha \in [0, 1]$ is the weight. For the case in Figure 1.1 the weight is equally shared, $\alpha = 0.5$. If, for every time instance, each agent chooses a new velocity outside the union of all RVOs *closest* to its current velocity, both *collision-free* and *oscillation-free* navigation is guaranteed. For proof, the reader is referred to [24]. In cases where there is no feasible new velocity outside the combined RVOs, a new velocity is chosen by the closeness to the preferred velocity and the minimum time to collision. The RVO algorithm accounts for dynamic constraints by choosing a new velocity in the union of dynamically constrained velocities and velocities outside the RVOs.

The required sensor measurements are the same for RVO and the algorithm presented in this thesis. Both operate in a planar multi-agent environment including

obstacles and are suitable for cluttered navigation. One of the main differences is the nonholonomic constraints that are considered in the proposed algorithm. Although similar preconditions, the resulting algorithms are inherently different and makes for an interesting comparison.

1.3 Assumptions

Throughout this thesis, there are several assumptions. First of all, the environment is a 2D planar surface. Secondly, agents are modeled as unicycles with nonholonomic constraints. The agent's forward speed is upper and lower bounded, and has a maximum rate of change. No assumption is made regarding the shape of the obstacles, moreover, they can be multiple and overlapping. Furthermore, the obstacles are considered passive, meaning they have zero control input and can be either static or moving. The obstacles' linear velocities are however assumed to be lower than the lower bound on agents forward speed. The agents' angular velocity is bounded, resulting in a minimum turning radius. Furthermore, it is assumed that the agents' heading rate and linear acceleration can be controlled directly. All agents operate as individual entities, and there is no communication between agents. The environment is completely a priori unknown, except for information gathered through local sensor measurements. Agents have access to their target bearing at all times. An abstract sensor model is assumed, where measurements are both noise- and error-free.

1.4 Background

In this thesis, the collision avoidance algorithm from [21] serves as a baseline and is presented in Appendix A. More precisely, the integrated environment representation resulting in a binary function for avoidance is used and expanded. The algorithm is chosen for its versatility in regard to size and shape of the obstacles as well as the interesting use of a sensor disk in front of the agents. The velocity compensation calculation provided in [23], serves as a method for incorporating the velocity of other vehicles in the algorithm. It is presented as a stand-alone calculation and is

well suited to extend the integrated sensor representation. See Appendix B for a detailed derivation. A basis for the Matlab simulator used in Chapter 5 is provided by PhD. candidate Martin Syre Wiig. More details about the simulator can be found in Appendix E.

1.4.1 Contributions

The main contribution of this thesis is a computational effective reactive algorithm for multi-agent navigation where the agents are restricted by nonholonomic constraints with bounded forward speed. The algorithm deals with moving obstacles, without being too conservative, in a cluttered environment. It is created by combining the algorithm presented in [21] with the obstacle velocity compensation from [23] and a new *braking rule*. The complete navigation strategy is analyzed by rigorous mathematical analysis to guarantee collision-free navigation in an environment with one agent and one obstacle, and in an environment with two agents. Furthermore, extensive simulation and testing is performed and presented in Chapter 5 and Appendix C. The simulations are implemented without the strict restrictions of the mathematical analysis and with multiple agents and obstacles. Details regarding the work behind building the simulator are described in Appendix E. Besides testing multiple different scenarios for the presented algorithm, this thesis also presents results comparing it to both the algorithm presented in [21] and [24]. By running the algorithms through thousands of simulation in different Monte Carlo experiments, the presented results are based on statistics and are more proportionate. The extensive simulations provide solid support to the analysis presented in Chapter 4, and to the claim that the algorithm works well for less strict conditions.

1.5 Outline

The remainder of this thesis is organized as follows. In Chapter 2 the system description is presented, both agent and obstacle models are provided including respective assumptions. Chapter 3 describes the proposed navigation and collision avoidance

algorithm. Chapter 4 provides the mathematical analysis, including proofs of the stated theorems. Chapter 5 presents several computer-simulated scenarios that analyze performance and provide insight into the behavior of the algorithm. The chapter also provides Monte Carlo experiments where the performance is compared with the algorithm from [21] and [24]. Lastly, Chapter 6 discusses the results, presents a conclusion and states further work. In addition, Appendix A presents the collision avoidance algorithm proposed by [21]. The method used to calculated the velocity compensation angle provided by [23] is shown in Appendix B. Additional Monte Carlo results are presented in Appendix C, testing the system parameters influence on overall performance. Appendix D present and compare Monte Carlo results with an algorithm created by Master of Science student, Erlend Hårstad. Finally, the work behind creating the simulator used in Chapter 5 is described in Appendix E.

Chapter 2

System Description

This chapter presents the mathematical model for both the agents and the obstacles. The main difference being that obstacles are passive; they do not navigate according to the proposed algorithm and in fact have no control input. All stated assumptions in both models are formally declared and listed in the last section of the chapter. The models presented in this chapter are directly implemented in Chapter 5, while some simplifications are made in Chapter 4. Throughout this thesis, all agent and obstacle movement is two dimensional along a planar surface.

2.1 Agent model

The agents are modeled as planar unicycle-type vehicles defined by

$$\dot{x}_i(t) = v_i(t)\,cos(\psi_i(t)) \tag{2.1a}$$

$$\dot{y}_i(t) = v_i(t)\,sin(\psi_i(t)) \tag{2.1b}$$

$$\dot{\psi}_i(t) = r_i(t) \tag{2.1c}$$

$$\dot{v}_i(t) = a_i(t) \tag{2.1d}$$

where i indicate agent number i, $x_i(t)$ and $y_i(t)$ are the agent's Cartesian coordinates. The linear velocity $v_i(t)$ is bonded by $v_i(t) \in [v_{min,i}, v_{max,i}]$, where $v_{max,i} > v_{min,i} > 0$. The heading, $\psi_i(t)$, is measured clockwise from the x-axis[1]. The heading rate $r_i(t)$ is the first control input and is bounded by $r_i(t) \in [-r_{max,i}, r_{max,i}]$, where $r_{max,i} > 0$. The second control input is the linear acceleration $a_i(t)$, bounded by $a_i(t) \in [-a_{max,i}, a_{max,i}]$, where $a_{max,i} > 0$. Agents are shaped as planar disks with radius $R_{r,i}$. This simple model can describe the motion of many vehicles, such as wheeled robots, unmanned aerial vehicles, underwater vehicles and missiles [33]. The agents' position is the center of the disk given by

$$\boldsymbol{p}_i(t) = [x_i(t),\ y_i(t)]^T \tag{2.2}$$

Using this model agents have a minimum turning radius given by

$$R_{turn,i} = \frac{v_{min,i}}{r_{max,i}} \tag{2.3}$$

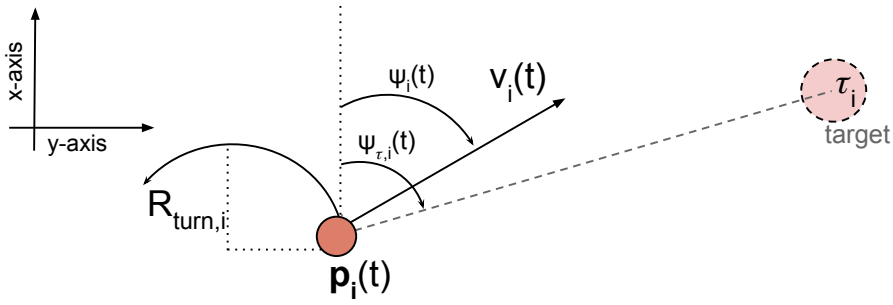


Figure 2.1: Agent model, where $R_{turn,i}$ is the minimum turning radius and the target is indicated by τ_i.

[1]This thesis follows the marine convention of NED-orientated coordinate system

2.2 Passive obstacle model

Obstacles are defined as areas in the environment that are forbidden to the agents. The shape and number of obstacles are arbitrary and they can be both static or dynamic. However, all obstacles are assumed to be passive, having no control input.

Definition 1. *The set of all obstacles are defined as* $E(t) \subset \mathbb{R}^2$ *consisting of all obstacles* $E_i(t)$ *for* $i = 1, 2, ..., n$ *that are forbidden to the agents.*

$E(t)$ is a priori unknown and assumed to have a closed piece-wise analytic boundary. The rate of change is constrained such that any point $\boldsymbol{e}(t) = [x_e(t), y_e(t)] \in E(t)$ has the property $\|\dot{e}(t)\| < v_{min}$. This implies that the $E(t)$ cannot change or move faster than the slowest moving agent. Thus each obstacle, $E_i(t)$, can move and rotate arbitrarily and independently.

Remark. *In the general case, collision-free navigation cannot be guaranteed for obstacles moving faster than an agent.*

Furthermore, a safety margin $d_{safe_i} > 0$ is introduced, that accounts for the size of agent i and contributes as a safety parameter. Hence, $d_{safe,i} \geq R_{r,i}$ for agent i. $E(t)$ is expanded with the safety distance d_{safe_i} to create a new set $\hat{E}(t)$. See Figure 2.2 for an example. Expanding to $\hat{E}(t)$ is only a matter of size, thus the same assumptions apply to $E(t)$ and $\hat{E}(t)$. The remainder of this thesis assume $\hat{E}(t)$ as the default obstacle set.

2.3 Sensing model

This sections follows along the lines of [21], for more details see Appendix A. The sampling period is defined as $\delta > 0$, where measurements and control input are updated at discrete times $t = 0, \delta, 2\delta, ...$ The agents' current heading, $\psi_i(t)$, is measured clockwise from the x-axis. Every agent has a corresponding target, τ_i, with radius $R_{\tau,i}$ and position $\boldsymbol{p}_{\tau,i} = [x_{\tau,i},\ y_{\tau,i}]^T$. The angle to the target center, $\psi_{\tau(t),i}$ is available at every time interval, see Figure 2.1.

Remark. *An agent does not distinguish between other agents and passive obstacles. All objects, obstacles and agents, inside the current agent's sensor range, is recognized as*

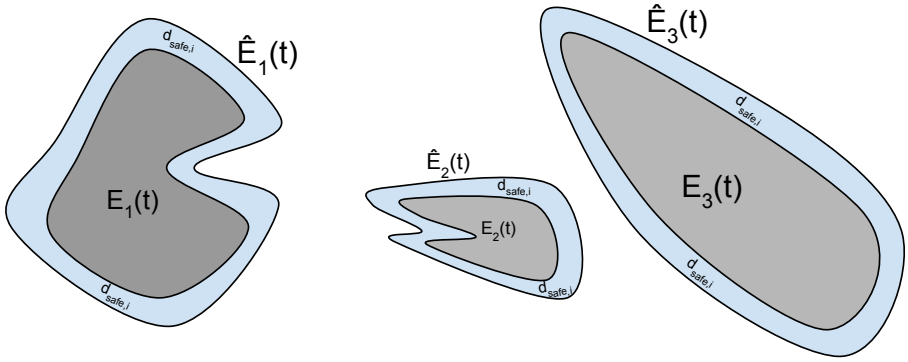


Figure 2.2: Illustrate three obstacles. Original environment $E(t)$ (dark gray) and the expanded environment $\hat{E}(t)$(light blue + dark gray), where $\hat{E}(t) = \hat{E}_1(t) + \hat{E}_2(t) + \hat{E}_3(t)$

obstacles the agent has to avoid. Hence, the remainder of this chapter focuses on agent-obstacles interaction, which does not exclude agent-agent encounters.

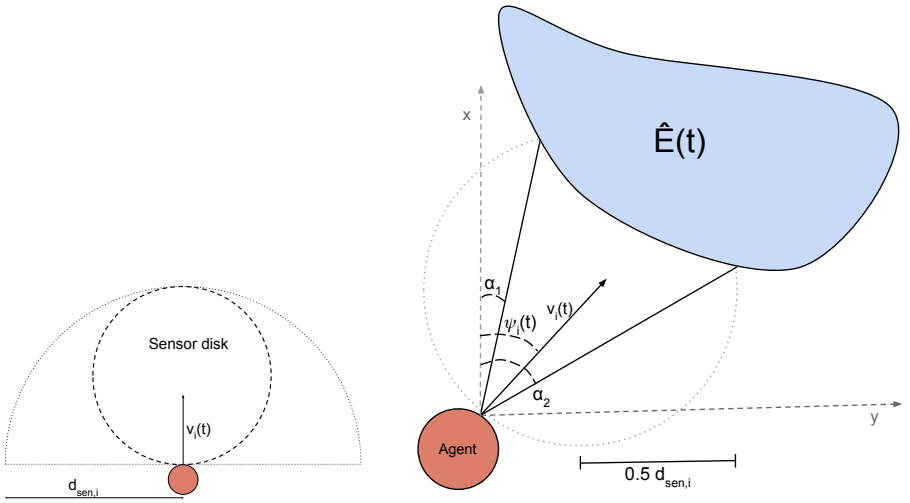
To better understand the integrated sensor model lets first look at a case with only stationary obstacles. A sensor on the agent emit rays in directions denoted α in a half circle defined by $[\psi_i(t) - \frac{\pi}{2},\ \psi_i(t) + \frac{\pi}{2}]$ and range $d_{sen,i}$. The sensor disk $\mathcal{D}_i(t)$ is introduced as a circle in front of the agent with diameter $d_{sen,i}$. The agent detects all obstacles inside the disk

$$\mathcal{D}_i(k\delta) = d_{sen,i}\ cos(\alpha), \quad \forall\ \alpha \in [\psi_i(k\delta) - \frac{\pi}{2},\ \psi_i(k\delta) + \frac{\pi}{2}]$$

at $t = k\delta$. See Figure 2.3a and Figure 2.3b.

Remark. *It is worth noting the assumption that $\hat{E}$ is sensed directly. In a real application, one would detect E and then calculate an angle to compensate for $d_{safe,i}$ to achieve $\hat{E}$. This angle is not necessarily straightforward to calculate and is an area for further work. The same argument goes for detecting other agents.*

Now, the binary function representing the integrated environment can be introduced:



(a) Sensor range and sensor disk

(b) An agent detecting an obstacle

Figure 2.3: Illustration of the sensor disk $\mathcal{D}$ and an agents' sensing ability.

Definition 2. *A binary function* $M_i(\alpha, t) \in \{0, 1\}$ *is defined for all* $t \geq 0$ *and* $\alpha \in [\psi_i(t) - \frac{\pi}{2},\ \psi_i(t) + \frac{\pi}{2}]$ *as*

$$M_i(\alpha, t) = \begin{cases} 1, & if \quad d_{\boldsymbol{p}_E} \leq d_{sen,i}\ cos\alpha \\ 0, & otherwise \end{cases} \tag{2.4}$$

where $d_{\boldsymbol{p}_E} := \|\boldsymbol{p}_i(t) - \boldsymbol{p}_E\|$ is the distance to the point, $\boldsymbol{p}_E$, where the ray emitted from the agent at time t in direction α hits $\hat{E}(t)$ or another agent. $\|\cdot\|$ is the standard Eucledian vector norm. Basic trigonometry prove that $d_{sen,i}\ cos(\alpha)$ is indeed a circle in front of the agent directly from the values of $cos(x)$ for $x \in \left[-\frac{pi}{2}, \frac{pi}{2}\right]$.

Remark. *A typical senor, in this case, could be a Lidar or Radar, where both the time interval* δ *and the sensor resolution of the emitted rays* α *have an impact on the performance of the overall navigational ability. Further sensor and measurement analysis are outside the scope of this thesis.*

One might argue against an agent that only senses obstacles inside a disk in front of itself to be considered safe. In Chapter 4 it is proved that the sensor disk representation is safe under certain assumptions and in Chapter 5 persuasive simulation are presented along with further discussion. The goal of using a sensor disk in this model is that it forces the agents to only avoid obstacles that are in danger of collision. Regarding the bounded agent's forward speed and limited turning radius resulting from the nonholonomic model, it is more efficient to avoid obstacles that the agent is directed towards. With the sensor disk, this is the case. Actively avoiding obstacles within the sensor range, $d_{sen,i}$, and outside the sensor disk, $\mathcal{D}(t)$, is considered conservative and inefficient.

Remark. *Note that the above arguments only hold under the assumption that no passive obstacle moves with a higher speed than the slowest agent and that $d_{safe,i}$ is chosen large enough. The size of the sensor disk is also of critical importance and is discussed in Chapter 4.*

2.3.1 Available measurements

The measurements available to an agent at every time interval is

- Agent heading $\psi_i(t)$
- Bearing to the target $\psi_{\tau,i}(t)$
- Obstacle forward speed $v_{\hat{E}}(t)$
- Obstacle heading $\psi_{\hat{E}}(t)$
- The binary function $M_i(\alpha, t)$

2.4 Velocity compensation

A velocity compensation angle is calculated to expand the binary function $M(\alpha, t)$ with obstacle velocity. By including obstacles' velocity the goal is to increase algorithm

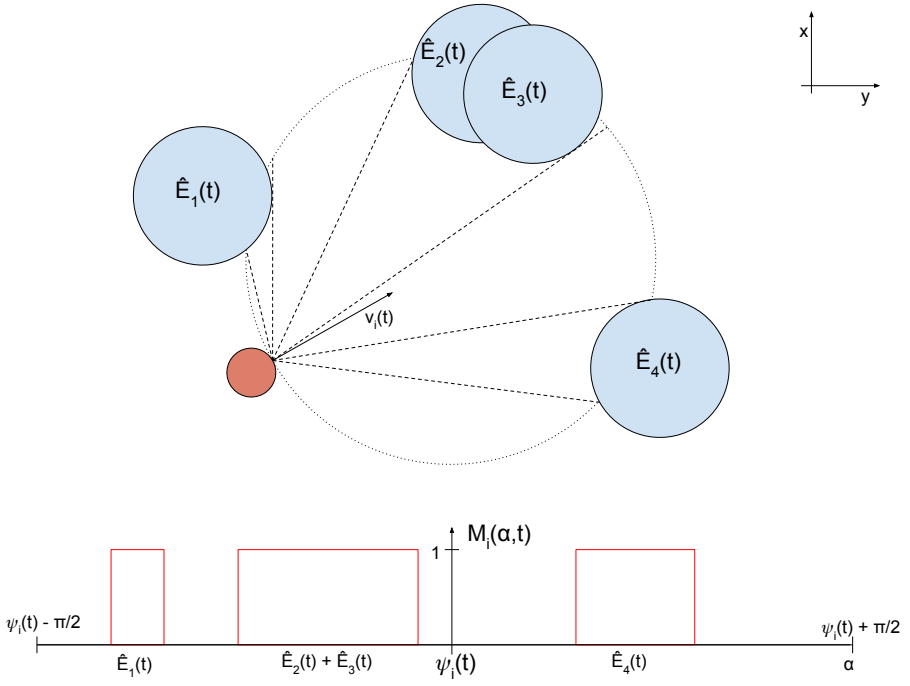


Figure 2.4: Example of $M(\alpha, t)$ showing multiple and overlapping obstacles together with the respective binary values.

safety and efficiency. Examples and discussion are provided in Chapter 5. The velocity compensated binary function is denoted $\hat{M}_i(\hat{\alpha}, t)$, where the collision areas in $\hat{M}_i$ are shifted to account for individual obstacle velocity. See Figure 2.5.

Remark. *Obstacle velocity can be obtained by a rate of change between time intervals and does not require additional sensor components.*

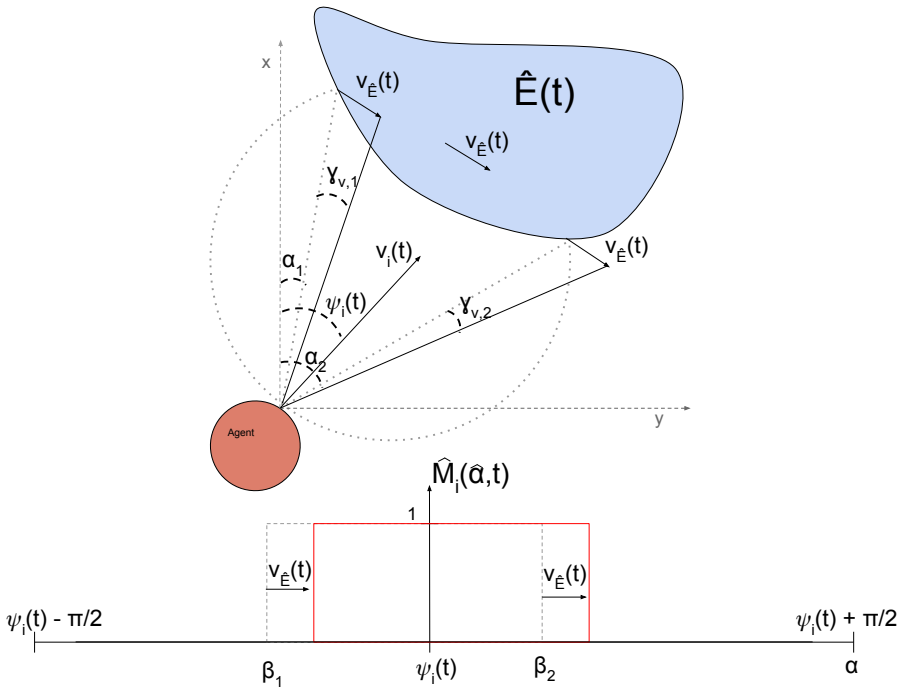


Figure 2.5: Illustration of the sensor disk $\mathcal{D}_i$ and the binary function $\hat{M}_i(\alpha, t)$ with obstacle velocity compensation.

Let's consider the scenario given in Figure 2.5, let α_1 and α_2 be the angles defining the obstacle on the sensor disk before velocity compensation. The desired velocity compensated angles can be expressed as

$$\hat{\alpha}_i = \alpha_i + \gamma_{v,i}, \quad for\ i = 1, 2 \tag{2.5}$$

where $\hat{\alpha}_i$ is bounded by $\hat{\alpha}_i \in [\psi_i(t) - \frac{\pi}{2},\ \psi_i(t) + \frac{\pi}{2}]$, maintaining a valid sensor disk. The compensation angle $\gamma_{v,i}$ is depicted in Figure 2.5. For more details about the derivation of $\gamma_{v,i}$, see Appendix B or the original paper [23].

2.5 Control objective

The control objective is for all agents to reach their target, τ_i, in finite time while keeping a minimum distance d_{safe_i} to any obstacle.

Definition 3. *An algorithm is target reaching if agent i reaches the target τ_i in finite time $t_f > 0$. The algorithm is collision-free if $\boldsymbol{p}_i(t) \notin \hat{E}(t),\ \forall t$, where $\boldsymbol{p}_i(t)$ is the position of agent i.*

In other words, any instance where $d_{\hat{E},min} < d_{safe,i}$ is considered a collision, where $d_{\hat{E},min} := \min_{\boldsymbol{p}_{\hat{E}} \in \hat{E}} \|\boldsymbol{p}_i(t) - \boldsymbol{p}_{\hat{E}}\|$. The navigation strategy is considered successful if it is both *target reaching* and *collision-free* for all agents.

2.6 Assumptions summary

Assumption 1. *All agent and obstacle movement is two dimensional along a planar surface.*

Assumption 2. *The sampling period is defined as $\delta > 0$, where measurements and agents' heading rates are updated at discrete times $0, \delta, 2\delta, \ldots$*

Assumption 3. *Agent movement is bounded by the following constraints:*

$$v_i(t) \in [v_{min,i},\ v_{max,i}], \qquad \textit{where} \quad v_{max,i} > v_{min,i} > 0 \tag{2.6a}$$

$$r_i(t) \in [-r_{max,i},\ r_{max,i}], \qquad \textit{where} \quad r_{max,i} > 0 \tag{2.6b}$$

$$a_i(t) \in [-a_{max,i},\ a_{max,i}], \qquad \textit{where} \quad a_{max,i} > 0 \tag{2.6c}$$

Assumption 4. *Obstacles have no control input.*

Assumption 5. *$\hat{E}(t)$ is a closed set with piece-wise analytic boundary. Any point $\hat{e}(t) = [x_{\hat{e}}, y_{\hat{e}}]^T$ in the set $\hat{E}(t)$ satisfies*

$$\dot{\hat{e}}(t) < \min_i v_{min,i} \tag{2.7}$$

Assumption 6. *The target, τ_i, is assumed to be a stationary circle with radius $R_{\tau,i} > R_{turn,i}$ and position $\boldsymbol{p}_{\tau,i} = [x_{\tau,i}, y_{\tau,i}]^T$. The angle to the target, $\psi_{\tau,i}(t)$, is available to the agent*[2].

Assumption 7. *All measurements are considered perfect in the sense that they are error- and noise-free.*

Assumption 8. *The velocity, both magnitude and direction, of each individual obstacle inside the sensor disk, $\mathcal{D}_i(t)$ is available to the respective agent.*

[2] $R_{turn,i}$ is the agent's minimum turning radius.

Chapter 3

Navigation and Collision Avoidance

The navigation strategy consists of two separate modes with a switching criteria. There are two control inputs, the heading rate and the linear acceleration. In Section 3.1 the *guidance law* is introduced. The guidance law is active when the sensor disk is obstacles-free. The *collision avoidance algorithm* presented in Section 3.2 is active when obstacles are detected inside the sensor disk. The heading rate is controlled by a simple bang-bang controller to ensure fast turning toward the target as well as fast obstacle avoidance

$$r_i(t) := \begin{cases} 0, & \tilde{\psi}_i = 0 \text{ and } \hat{m}_i(t) = 0 \\ \pm r_{max,i}, & \tilde{\psi}_i \neq 0 \text{ or } \hat{m}_i(t) = 1 \end{cases} \tag{3.1}$$

where $\tilde{\psi}_i = \psi_i - \psi_{\tau,i}$ is the heading error and $\hat{m}_i(t) = 1$ when an obstacle is inside the sensor disk and zero otherwise. The function $\hat{m}(t)$ defined in Section 3.2. In situations where the risk of collision is high, an acceleration controller slows the agents forward speed to enhance safety. The results from Chapter 5 show that the combination of heading and speed controller significantly increases the overall safety of the navigation

strategy.

Remark. *The heading controller (3.1) is implemented as a saturated high-gain P controller to avoid chattering.*

3.1 Guidance law

When an agent is not actively avoiding collisions, a simple *pure pursuit* guidance law [10] steers the agent towards the target

$$\psi_{d,i}(t) := atan2(y_{\tau,i} - y_i(t),\ x_{\tau,i} - x_i(t)) \tag{3.2}$$

where $[x_i(t),\ y_i(t)]$ and $[x_{\tau,i},\ y_{\tau,i}]$ are the x- and y-coordinates of the agent and target center respectively. The desired heading, $\psi_{d,i}(t)$, is chosen in the interval $\psi_{d,i} \in (-\pi, \pi]$ to ensure shortest turn toward the desired direction. The function $atan2$ is the four-quadrant inverse tangent, where the signs of the arguments are used to place the output in the right quadrant. In contrast, the normal inverse tangent have an output in the range $\left[-\frac{\pi}{2},\ \frac{\pi}{2}\right]$. An important feature of (3.2) is that $\psi_i(t) = \psi_{d,i}(t)$ results in constant heading while no obstacles are detected.

3.2 Collision avoidance

The introduction of the collision avoidance algorithm is divided into three parts. First, the switching criteria including binary function $m(t)$ is introduced. Then, the braking rule is defined and explained. Finally, in Section 3.2.2, a collision-free heading is chosen. The binary function $\hat{m}_i(t)$ is defined as:

$$\hat{m}_i(t) := \begin{cases} 0, & if\ \hat{M}_i(\hat{\alpha}, t) = 0\ \forall\ \hat{\alpha} \in [\psi_i(t) - \frac{\pi}{2}, \psi_i(t) + \frac{\pi}{2}] \\ 1, & otherwise \end{cases} \tag{3.3}$$

If $\hat{m}_i(t) = 1$, there are one or more obstacles inside the sensor disk at time t and $\hat{m}_i(t) = 0$ if the sensor disk is empty.

Assumption 9. *Initially, at $t = 0$, no obstacles are inside the sensor disk, i.e. $\hat{m}_i(0) = 0$*

3.2.1 The braking rule

Rules of the road are situation dependent rules that apply to all agents. The rules are inherent in the algorithm, thus there is no need for inter-agent communication. The braking rule applied in this thesis resembles a typical yield-pass rule, where agents to the left yield and agents to the right pass, similar to the scenario in Figure 3.1. Let $\Omega \in [0, \frac{\pi}{2}]$ be an angle on the sensor disk defining when the rule applies. See Figure 3.2. There are two different scenarios where the rule applies.

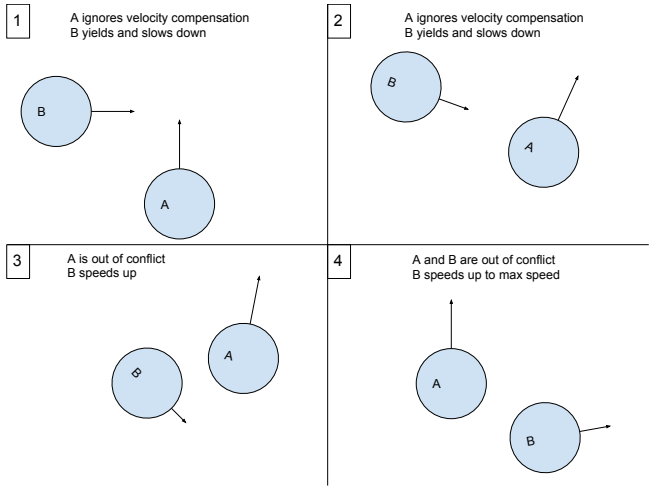


Figure 3.1: Example scenario illustrating the desired yield-pass behavior introduced by braking rule.

First, an obstacle is in the area on the sensor disk for angles *less* than $-\Omega$ and the velocity compensated obstacle is in the right half of the sensor disk, indicated by obstacle A in Figure 3.2, resulting in the agent ignoring the velocity compensation and marks the left half of $\hat{M}_i(\hat{\alpha}, t)$ from the obstacle as occupied. This maneuver lets the agent pass and is similar to the scenario for agent A in Figure 3.1.

Secondly, an obstacle is in the area on the sensor disk for angles *larger* than Ω and the velocity compensated obstacle is in the left half of the sensor disk, indicated by obstacle B in Figure 3.2, results in the agent braking according to

$$a_i(t) = \begin{cases} -a_{max,i}, & v_i(t) > v_{min,i} \\ 0, & otherwise \end{cases} \tag{3.4}$$

The former is a yield maneuver, resembling that of agent B in Figure 3.1. Let the binary function $b_i(t)$ indicate agent i in braking mode

$$b_i(t) = \begin{cases} 1, & \text{agent in braking mode} \\ 0, & otherwise \end{cases} \tag{3.5}$$

To ensure smooth navigation and alleviate churning the agents are forced to brake for a minimum amount of time T_{brake}, even if there are no obstacles on the sensor disk fulfilling the braking criteria in the current time instance. For obstacles C and D in Figure 3.2 the agent operate as normal.

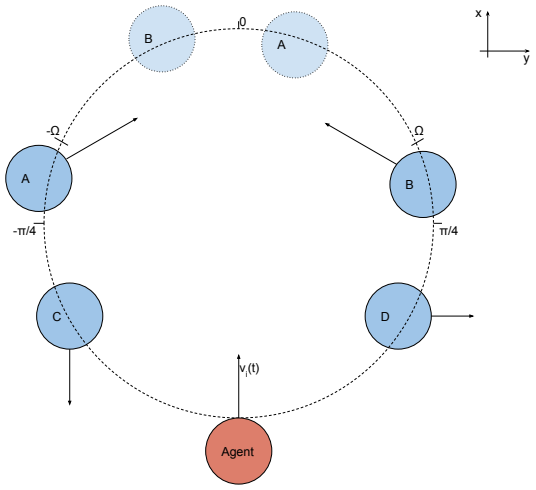


Figure 3.2: Example show scenarios for when the braking rule applies.

Remark. *At first sight, the braking rule might not seem intuitive. However, the rule achieves cooperative behavior in scenarios where velocity compensation is at it most extreme and cause agents to turn in opposite direction of the uncompensated case.*

3.2.2 Choosing a collision-free heading

Consider again Figure 2.4, where an example of $M_i(\alpha, t)$ is given in a scenario with four obstacles. Combine this with the obstacles velocity compensation illustrated in Figure 2.5 to obtain the new Figure 3.3. In this scenario $\hat{m}_i(t) = 1$ and there exist some $\hat{\alpha} \in [\psi_i(t) - \frac{\pi}{2},\ \psi_i(t) + \frac{\pi}{2}]$ where $\hat{M}_i(\hat{\alpha}, t) = 0$, illustrated by the open intervals $[A_i^-,\ A_i^+]$ for $i = 1, 2, 3, 4$. A new heading is chosen as the closest obstacle-free interval to the current heading and the middle value $C(t)$ of the interval is calculated by

$$j_i(t) := arg\ \min_i \{|A_i^-|,\ |A_i^+|\} \tag{3.6}$$

where $j_i(t)$ is the index of the A_i^- or A_i^+ that is closest to $\psi_i(t)$. The middle value is found by:

$$C_i(t) = \frac{A^-_{j_i(t)} + A^+_{j_i(t)}}{2} \tag{3.7}$$

where A_3^- is the closest start/end of an interval in this example. Thus $j(t) = 3$ and $C(t)$ is the middle of the interval $[A_3^-,\ A_3^+]$. Note that $C(t)$ is indeed an angle.

Remark. *The braking rule is applied before $C_i(t)$ is chosen, such that $\hat{M}_i(\hat{\alpha}, t)$ is updated with the correct open intervals.*

Remark. *The algorithm for choosing a collision-free heading follows the lines of [21] when the velocity compensation and braking rule are neglected.*

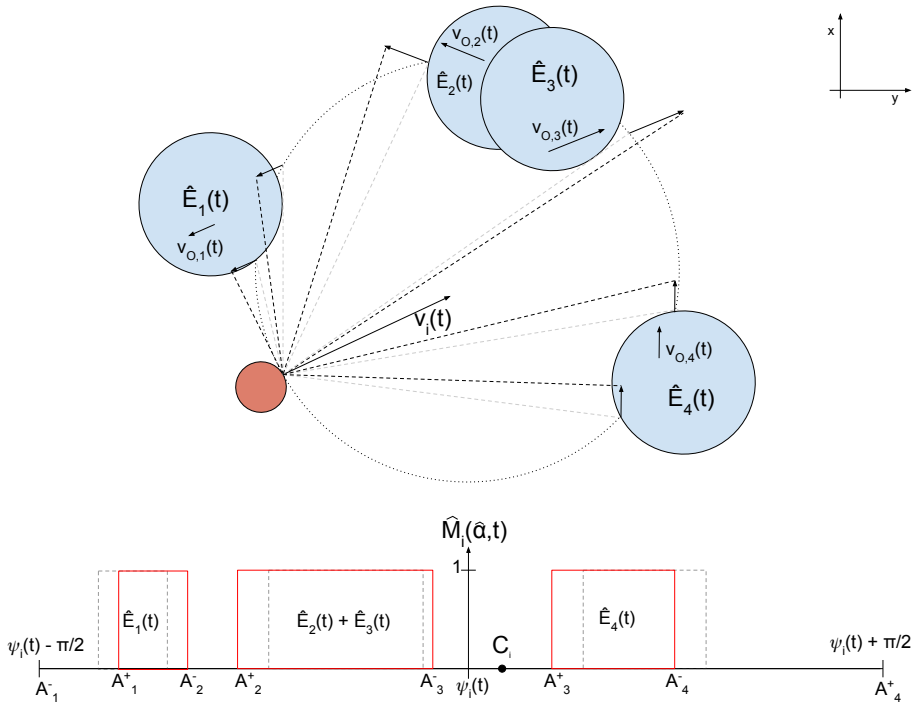


Figure 3.3: Example scenario illustrating multiple and overlapping obstacles with four obstacle-free intervals $[A_i^-,\ A_i^+]$ for $i = 1, 2, 3, 4$.

3.3 Reactive navigation law

With both the guidance law and the collision avoidance algorithm introduced, the complete navigation law is given as

$$r_i(t) := \begin{cases} r_{max,i}\,\mathrm{sgn}(\psi_{d,i}(t) - \psi_i(t)), & if\ \hat{m}_i(t) = 0 \\ r_{max,i}\,\mathrm{sgn}(C_i(t) - \psi_i(t)), & if\ \hat{m}_i(t) = 1 \end{cases} \tag{3.8a}$$

$$a_i(t) := \begin{cases} a_{max,i}, & if\ b_i(t) = 0 \text{ and } v_i(t) < v_{max,i} \\ -a_{max,i}, & if\ b_i(t) = 1 \text{ and } v_i(t) > v_{min,i} \\ 0, & otherwise \end{cases} \tag{3.8b}$$

where $sgn(\cdot)$ returns the sign of its argument, $C_i(t)$ is given by (3.6) and (3.7), and $\psi_{d,i}(t)$ is given by (3.2). The inputs, $r_i(t)$ and $a_i(t)$, are constant during the interval $[t,\ t + \delta]$. The navigation algorithm is fairly simple; if there are one or more obstacles inside the sensor disk, the agent turns as fast as possible towards the closest obstacle-free area. Otherwise the agent turns as fast as possible toward its target. The same goes for the acceleration controller, if the agent is in a yielding position it slows down, otherwise it aims for maximum speed.

Chapter 4

Mathematical Analysis

Mathematical analysis of the proposed algorithm (PA) is performed on an environment with reduced complexity. The outline of this chapter is as follows: First, the collision avoidance algorithm is analyzed in the simplest case with one agent and one obstacle without velocity compensation or the braking rule. Followed by the introduction of velocity compensation. In Section 4.3 the braking rule is analyzed. Lastly, an agent-agent encounter is analyzed for the complete algorithm.

Definition 4. *The set $\hat{O}(t)$ is a closed convex set, consisting of a number of non-overlapping individual and disk-shaped passive obstacles $\hat{O}_i(t)$ with radius $r_{\hat{O}} > 0$. Let $v_{\hat{O}}(t) = |\dot{p}_{\hat{O}}|$ be the obstacle speed and $p_{\hat{O}} = (x_{\hat{O}},\ y_{\hat{O}})$ is defined as the obstacle position.*

The extended obstacle set $\hat{E}(t)$ is reduced to $\hat{O}(t)$ containing only disk-shaped obstacles.

Assumption 10. *The maximum obstacle speed satisfies $v_{\hat{O},max} > v_{\hat{O}}(t) \geq 0\ \forall t \geq 0$*

Assumption 10 bounds the obstacle forward speed by a positive upper limit $v_{\hat{O},max} > 0$. The limit is be extensively used throughout this chapter and can, combined with Assumption 12, be seen as a stricter form of Assumption 5. Note that obstacles where $v_{\hat{O}}(t) = 0$ are static.

4.1 Uncompensated obstacle measurements

Starting with the simplest case, this section analyzes the uncompensated binary function $M_i(\alpha, t)$ and deduce sufficient conditions for collision-free navigation without compensating for obstacle velocity. Note that the analysis below is remain unchanged for constant forward speed, $v_{max} = v_{min}$.

Assumption 11. *All agent are bounded by global constraints, v_{min}, v_{max}, a_{max}, r_{max}, R_r and d_{sen}.*

Assumption 12. *The following inequalities hold:*

$$\frac{v_{min}\sin(r_{max}\delta)}{r_{max}} > v_{\hat{O},max}\delta \tag{4.1}$$

$$\frac{v_{min}(1-\cos(r_{max}\delta))}{r_{max}} > v_{\hat{O},max}\delta \tag{4.2}$$

$$d_{sen} > 2(v_{max} + v_{\hat{O},max})\delta \tag{4.3}$$

Let $d_{ij}(t) = \min_{i,j} \|p_{O,i}(t) - p_{O,j}(t)\|$ be the distance between two obstacle i and j.

Assumption 13. $d_{ij}(t) > d_{sen} + 2v_{max}\delta \quad \forall\ t \geq 0$.

Assumption 14. *No obstacles are, at any time interval t, inside the half-circle directed opposite to the current heading, with the agent as center and radius $2v_{min}\delta$.*

See Figure 4.1.

Remark. *Assumption 12-14 are not too conservative. Assumption 12 is achieved by a large enough d_{sen} and (4.1-4.2) is close to $v_{min} > v_{\hat{O},max}$ for small δ. Assumption 13 restrict the distance between two obstacles such that only one obstacle is inside $\mathcal{D}_i(t)$ in one time interval. This is a rather strict assumption considering that the algorithm is designed for cluttered environments. Note, however, that the assumption is only used for mathematical analysis and as Chapter 5 show, the algorithm works with more obstacles inside $\mathcal{D}_i(t)$. Again, Assumption 14 is not restrictive for small δ.*

Theorem 1. *The navigation strategy (3.8) is collision-free, given Assumptions 12-14.*

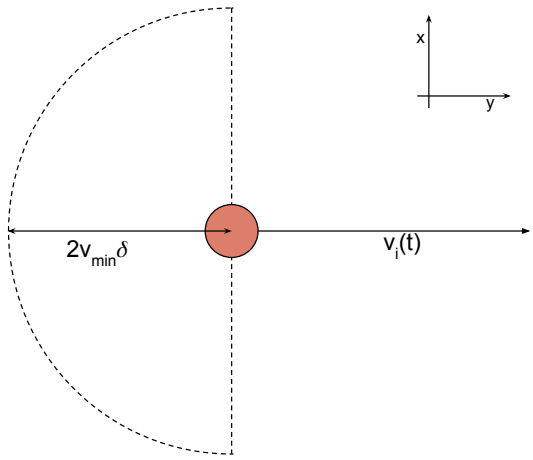


Figure 4.1: Illustrate the half-circle stated in Assumption 14

4.1.1 Proof of Theorem 1

The proof of Theorem 1 follows along the lines of [21]. Consider a new disk $\mathcal{D}_{0,i}(t)$, concentric to $\mathcal{D}_i(t)$, with diameter $d_{sen} - 2v_{min}\delta > 0$, according to (4.3). An illustration is given in Figure 4.2. The proof is divided into two parts, first for $r_i(k\delta) = 0$ then for $r_i(k\delta) \neq 0$, where $r_i(t)$ is the agent's angular velocity.

$r_i(k\delta) = 0$. From the navigation strategy (3.8a) and $r_i(k\delta) = 0$, it is given that $\hat{O}(k\delta)$ does not overlap with the sensor disk $\mathcal{D}_i(k\delta)$. The diameter of $\mathcal{D}_{0,i}(t)$ is chosen such that the agent is at the border of, or inside, $\mathcal{D}_{0,i}(k\delta)$ at $t = (k+1)\delta$ for $r_i = 0$ in $t \in [k\delta,\ (k+1)\delta]$, i.e. the distance between the border of the disks is

$$\frac{d_{sen}}{2} - \frac{d_{sen} - 2v_{min}\delta}{2} = v_{min}\delta \tag{4.4}$$

For $v_i(t) = v_{max}$, Assumption 4.3 and the size of $\mathcal{D}_{0,i}(t)$ guarantee that the agent is inside $\mathcal{D}_{0,i}(k\delta)$ at $t = (k+1)\delta$. By proving that $D_{0,i}(t)$ is obstacle-free at $t = (k+1)\delta$, the interval must be collision-free. An obstacle can maximum move $s_{\hat{O},max} = v_{\hat{O},max}\delta$

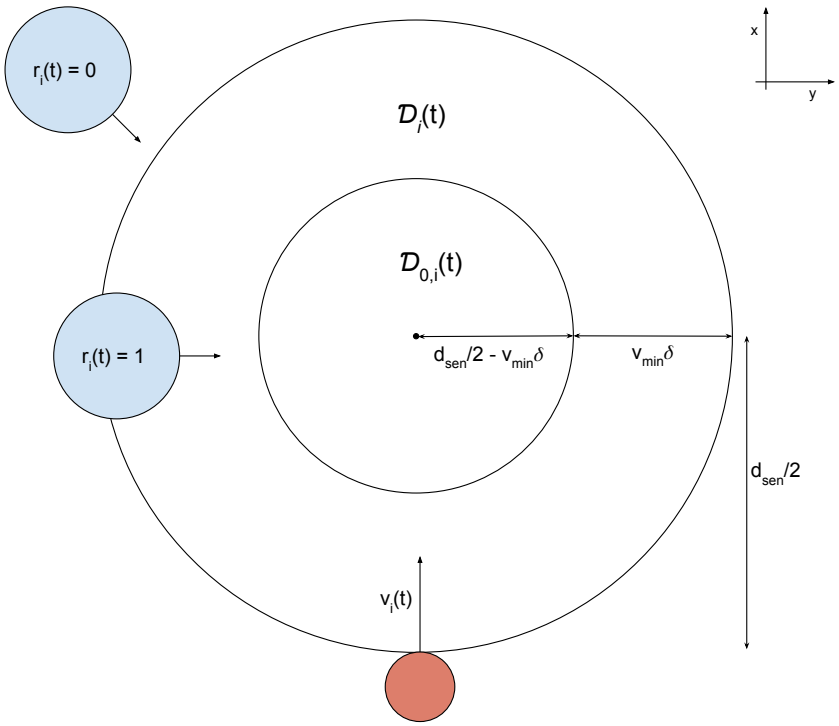


Figure 4.2: Illustrate of how $\mathcal{D}_{0,i}(t)$ relate to $\mathcal{D}_i(t)$. The figure also show possible obstacles positions for the two different cases; $r_i(t) = 0$ and $r_i(t) \neq 0$ at $t = k\delta$.

in one time instance, directly from the basic linear equation $distance = speed \times time$. Consider an obstacle approaching the center of $\mathcal{D}_{0,i}(k\delta)$ with maximum speed $v_{\hat{O},max}$. By (4.1), $v_{min} > v_{\hat{O},max}$ and thus the obstacle can not be inside $\mathcal{D}_{0,i}(k\delta)$ at $t = (k+1)\delta$. Also note that (4.3) ensures that d_{sen} is large enough for $\mathcal{D}_{0,i}$ to exist inside $\mathcal{D}_i$ with the stated diameter.

$r_i(k\delta) \neq 0$. When considering $r_i(k\delta) \neq 0$, $\hat{O}(k\delta)$ intersects with $\mathcal{D}_i(k\delta)$ according to (3.8). There are two possible responses, $r_i(k\delta) = \pm r_{max}$, right and left turn. This proof follows the line of a right turn $r_i(k\delta) = r_{max}$, note that the exact same arguments hold for a left turn. Because $v_{min} > v_{\hat{O},max}$ it is not possible for $\mathcal{D}_{0,i}(k\delta)$ and $\hat{O}(k\delta)$ to intersect in the first time interval an obstacle is detected. According to (3.8) a right turn, $r_i(t\delta) = r_{max}$, can only happen for a single obstacle if the intersection of $\hat{O}(k\delta)$ and $\mathcal{D}_i(k\delta)$ is in the left half of $\mathcal{D}_i(k\delta)$. Hence, proving that $\mathcal{D}_{0,i}((k+1)\delta)$ and $\hat{O}((k+1)\delta)$ does not intersect guarantees that no collision occurs. See Figure 4.3b for an illustration.

Consider the movement in x- and y-direction from $\mathcal{D}_{0,i}(k\delta)$ to $\mathcal{D}_{0,i}((k+1)\delta)$ as shown in Figure 4.3a.

$$\Delta x_i = \frac{v_{max}}{r_{max}} \sin(r_{max}\delta) > R_{turn} \sin(r_{max}\delta) \tag{4.5a}$$

$$\Delta y_i = \frac{v_{max}}{r_{max}} (1 - \cos(r_{max}\delta)) > R_{turn}(1 - \cos(r_{max}\delta)) \tag{4.5b}$$

where $\Delta x_i \triangleq x_i((k+1)\delta) - x_i(k\delta)$ and $\Delta y_i \triangleq y_i((k+1)\delta) - y_i(k\delta)$ is the agent displacement in x- and y-direction respectively. R_{turn} is the minimum turning radius as given by:

$$R_{turn} = \frac{v_{min}}{r_{max}}$$

By equations (4.1) and (4.2), the obstacle to the left and $\mathcal{D}_{0,i}(t)$ do not intersect at $t = (k+1)\delta$, for

$$\Delta x_i > v_{\hat{O},max}\delta \quad \text{and} \quad \Delta y_i > v_{\hat{O},max}\delta$$

Assumption 10 and 14 ensures that no obstacle behind or to the right of the agent is

inside $\mathcal{D}_i((k+1)\delta)$ and hence not in $\mathcal{D}_{0,i}((k+1)\delta)$.

4.2 Velocity compensation angle

By including the obstacle velocity compensation, the analysis changes in some scenarios. First of all, observe that the analysis for $r_i(k\delta) = 0$ applies as it is. While the case where $\mathcal{D}_i(k\delta)$ and $\hat{O}(k\delta)$ intersect changes. Consider the three different scenarios illustrated in Figure 4.4 and observe that the only case that differs from the analysis above is case 2. Note that the same arguments hold for an obstacle heading leftwards. Obstacles heading straight toward or away from the agent are similar to case 1 and 3. However, case 2 differs by causing the agent to left turn in a case that would have resulted in a right turn without velocity compensation.

Arguably, turning left in case 2 is considered a safer maneuver. By Assumption 4, obstacles have no control input and thus does not change path. The maneuver in case 2 causes the agent to move behind or away from, rather than turning in the same direction as the obstacle.

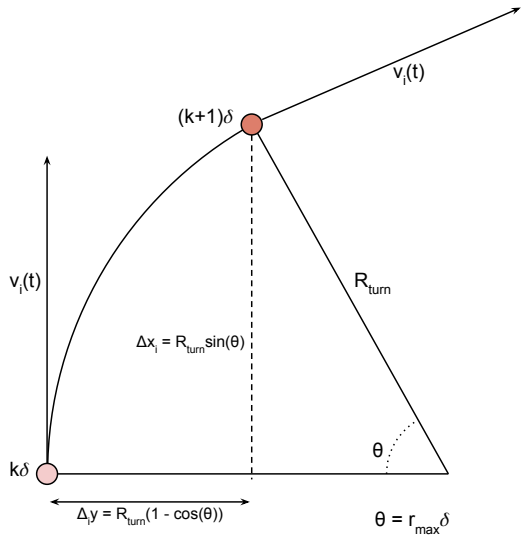
Proposition 1. *If the agent turns left at $t = k\delta$ when an obstacle is in the left half of $\mathcal{D}_i(k\delta)$, the obstacle must be in the upper left quadrant of $\mathcal{D}_i(k\delta)$*

Proof. The velocity compensated angle is calculated according to equation (B.3)

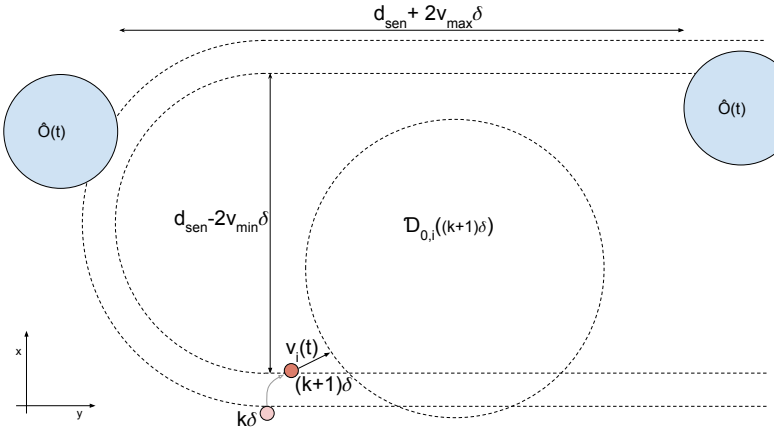
$$\gamma_{vc} = \sin^{-1}\left(\frac{v_{\hat{O},max}\sin(\pi - \psi_{\hat{O}} + \alpha)}{v_i(t)}\right) \tag{4.6}$$

where γ_{vc} is the angle describing the velocity compensation and $\psi_{\hat{O}}$ is the obstacle heading. Referring to Chapter 2, α is the angle of the left sensor-ray because the obstacle of interest is in the left half of $\mathcal{D}_i(k\delta)$. The mathematical condition for a left turn in this case is

$$\gamma_{vc} - \alpha > 0 \longrightarrow \gamma_{vc} > \alpha$$



(a)



(b)

Figure 4.3: Collision avoidance for $r_i(k\delta) \neq 0$. Figure (a) indicate the agent's movement during one time step while performing a right turn. In (b) the scenario is illustrated with two obstacles, the sensor range and $D_{0,i}$.

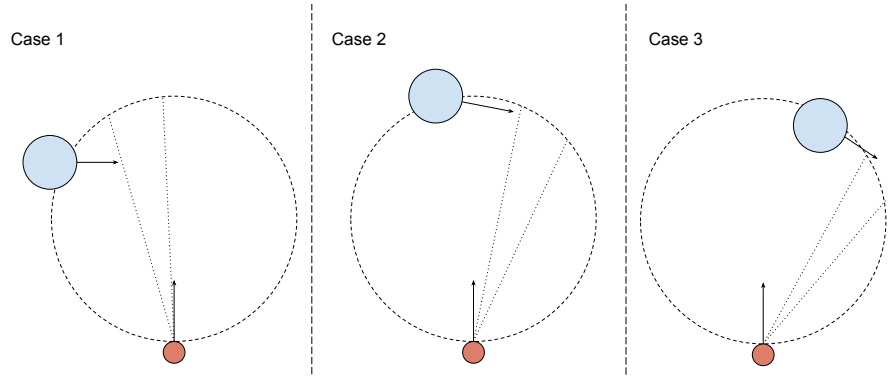


Figure 4.4: Three different cases where the cones indicate the velocity compensated obstacles for binary function $\hat{M}_i(\hat{\alpha}, t)$.

where the γ_{vc} is largest in regard to the obstacle when $\psi_{\hat{O}} = \frac{\pi}{2}$. See Figure 4.5. Thus

$$\alpha < \gamma_{vc} \tag{4.7a}$$

$$\alpha < \sin^{-1}\left(\frac{v_{\hat{O},max}\sin(\frac{\pi}{2}+\alpha)}{v_i(t)}\right) \tag{4.7b}$$

$$\sin(\alpha) < \frac{v_{\hat{O},max}}{v_i(t)}\sin(\frac{\pi}{2}+\alpha) \tag{4.7c}$$

$$\sin(\alpha) < \sin(\frac{\pi}{2}+\alpha) \tag{4.7d}$$

$$\sin(\alpha) < \cos(\alpha) \tag{4.7e}$$

$$\alpha < \frac{\pi}{4}, \quad for\ \alpha \in [0, \frac{\pi}{2}] \tag{4.7f}$$

where $\frac{v_{\hat{O},max}}{v_i(t)} < 1$. Proving that the obstacle must be in the upper left quadrant of $\mathcal{D}(k\delta)$. □

Proposition 2. *An agent turning with maximum turning rate, $\pm r_{max}$ at $t = k\delta$, is not in the upper half of $\mathcal{D}(k\delta)$ at $t \in [k\delta,\ (k+1)\delta]$*

Proof. Figure 4.3a show that the agent translation along the x-axis during one time

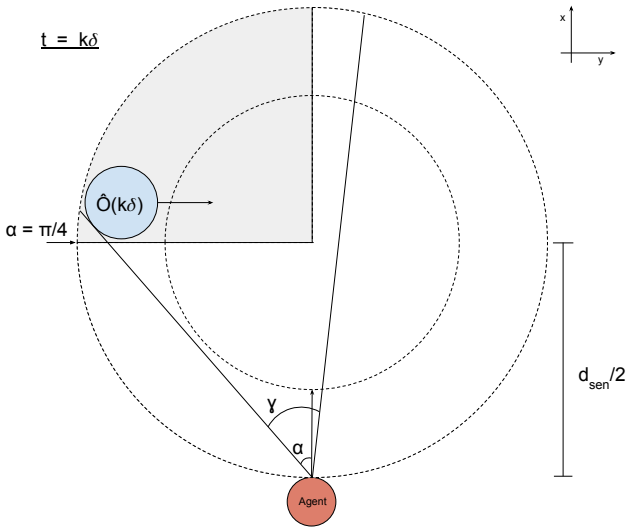


Figure 4.5: Proposition 1 support figure.

interval is $\frac{v_{max}}{r_{max}} \sin(r_{max}\delta)$. Proposition 2 is proved by showing that $\frac{v_{max}}{r_{max}} \sin(r_{max}\delta) < \frac{d_{sen}}{2}$.

$$\frac{v_{max}}{r_{max}} \sin(r_{max}\delta) < \frac{d_{sen}}{2} \tag{4.8a}$$

Because $v_{max} > v_{min}$

$$\frac{v_{min}}{r_{max}} \sin(r_{max}\delta) < \frac{d_{sen}}{2} \tag{4.8b}$$

Applying Assumption 12, equation (4.1) give

$$v_{\hat{O},max}\delta < \frac{d_{sen}}{2} \tag{4.8c}$$

Which is true by Assumption 12, equation (4.3). □

Together proposition 1 and 2 prove that the navigational strategy (3.8) is collision-free with velocity compensation.

4.3 The braking rule

Let's first take a closer look at a pass-maneuver illustrated by A in Figure 3.1. The agent ignores the velocity compensation in this case and mark the area from the obstacle to the current heading $\psi_i(t)$ on $\mathcal{D}_i(t)$ as blocked, see Figure 4.6a. This is in fact the same case as the one analyzed in Section 4.1 and is collision-free by Theorem 1. The second braking rule scenario is the yield-maneuver, illustrated by B in Figure 3.1. This is the only case in the collision avoidance algorithm where the agent changes forward speed by slowing down. Besides slowing down, the agent-obstacle relative position is similar to the problem analyzed in Section 4.2. Proving that the braking rule is collision-free.

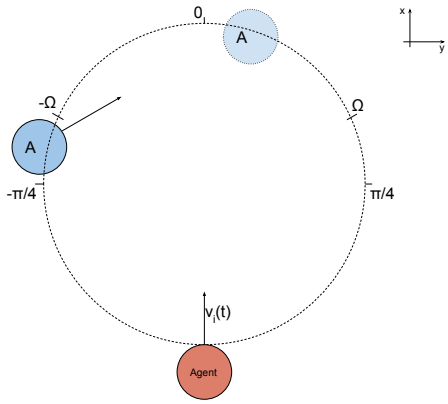
Remark. *According to the navigation strategy (3.8), the lower bounded forward speed is v_{min}, rendering the proof in Section 4.1 and 4.2 valid.*

4.4 Agent-agent encounter

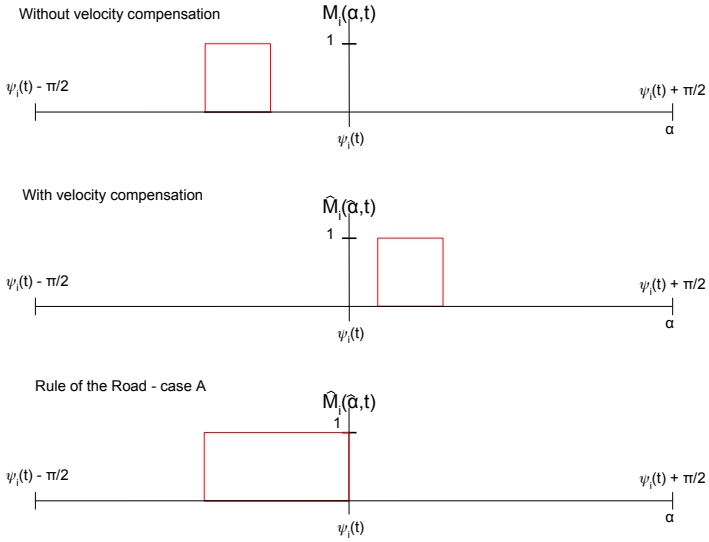
In this section, an agent-agent encounter is analyzed where both agents are guided by the navigation strategy given in (3.8). Keeping in mind that the strategy does not distinguish between encounters with passive obstacles and other agents, the most interesting case to analyze in this section is one where both agents are bounded by the same constraints, speed, heading rate, size etc, creating a scenario furthest from a passive obstacle encounter. As with passive obstacles, lets first look at a case without velocity compensation.

Proposition 3. *Without velocity compensation, two agents detecting the other always turns in opposite directions, away from each other.*

Proof. If agent A detects agent B in the left half of $\mathcal{D}_A(t)$, then agent B has to detect agent A in the right half of $\mathcal{D}_B(t)$, this is true considering the mirrored effect from one agent to the other. Hence, by equation (3.8a) the agents turn away from each other. See Figure 4.7. □



(a) A possible situation where the agent fulfill a pass-maneuver according to the braking rule.



(b) The sensor binary function $\hat{M}_i(\hat{\alpha}, t)$ for uncompensated, compensated and with the braking rule, for the obstacle in 4.6a.

Figure 4.6: The braking rule pass-case.

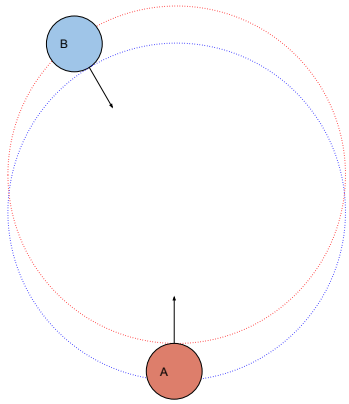


Figure 4.7: Two agents on collision course.

Theorem 2. *Two agents, A and B, where*

$$B \in \mathcal{D}_A(k\delta) \ \cap \ B \notin \mathcal{D}_{0,A}(k\delta)$$
$$A \in \mathcal{D}_B(k\delta) \ \cap \ A \notin \mathcal{D}_{0,B}(k\delta)$$

does not collide at $t = (k+1)\delta$.

Proof. The proof is given directly from Proposition 3. Because A and B turns with r_{max} away from each other, the shortest distance between A and B during $t \in [k\delta, (k+1)\delta]$ is at $t = k\delta$. Thus, making it impossible for A and B to collide. □

Lets now include velocity compensation and again look at the three different cases in Figure 4.4, where only case 2 change the agents' behavior of the above analysis. This problem can be broken down into three new cases, illustrated in Figure 4.8. Case 1 is exactly the same as the one analyzed in Section 4.2. Case 3 is also similar, the difference being that B also turns. The analysis from Section 4.2 still apply to this case, as $v_B(t)$ has to be significantly smaller than $v_A(t)$ for the scenario to be possible. Which leave case 2, in fact the forward speed of A and B has to be in the same magnitude for case 2 to happen. The only way to guarantee safety in case 2 is to include the braking rule.

By doing so one can observe that the case changes such that A slows down and B turns right. Resulting in a scenario similar to the analysis done in Section 4.3. Hence, by the analysis presented in Sections 4.1-4.3 an agent-agent encounter is collision-free when both agents are controlled by the navigation strategy (3.8).

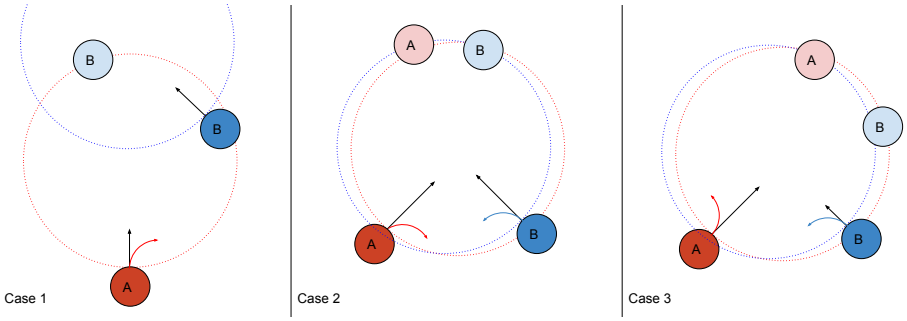


Figure 4.8: The dotted lines indicate the agents' sensor disk, where A is red and B is blue. The faded disk indicate where the velocity compensated agents are seen. The coloured arc indicates turning direction with velocity compensation. Case 1, A detects B and velocity compensation cause A to turn right. Case 2, both agents detect the other and velocity compensation cause the agents to turn towards each other. In case 3, both agents detect the other, but because B has lower forward speed, only agent B change turn direction as a result of velocity compensation.

4.5 Target reaching

Assumption 15. *The distance $d_{\tau,12}$ between two targets τ_1 and τ_2 is restricted by*

$$d_{\tau,12} > 4v_{max}\delta + \max_i R_{r,i}, \quad \textit{for } i \in [1,2]$$

Assumption 15 ensures that agent do not detect other agents close to their targets. Without the assumption agents can possibly block another agents' target by being to close, resulting in a deadlock.

Theorem 3. *The navigation strategy given in (3.8) is target reaching, given Assumptions*

10-15.

Proof. From the navigation strategy (3.8), it is clear that agent i steers towards the target as fast as possible when there are no obstacles inside the sensor disk $\mathcal{D}_i(t)$. Because the strategy switches between $r_i(t) = \pm r_{i,max}$, it is possible for agent i to converge towards a circle with radius R_{turn} around the target center. Assumption 6 solves the problem ensuring that the target's radius is larger, $R_{\tau,i} > R_{turn}$. In Sections 4.1 - 4.4 it has been proven that the strategy is collision-free. Thus by considering that $v_{min} > v_{\hat{O},max}$ it is evident that agent i escapes every obstacle in finite time $t < \infty$. Assumption 13 restricts the distance between obstacles such that there is only one obstacle inside the $\mathcal{D}_i(t)$ at all times. Furthermore, Assumption 15 restricts the distance between two targets such that two agents close to their target are not disturbed by each other. Hence, agent i escapes every obstacle and move toward the target when $\mathcal{D}_i(t)$ is obstacle-free. Thus, the navigation strategy (3.8) is target reaching. □

Remark. *When target reachability is analyzed, it is assumed that no obstacles actively blocks the target. This is obviously a necessary assumption, take for example the case where a static obstacle fully covers a target, then there is no finite path ensuring that the agent reaches its target.*

The navigation strategy given in (3.8) is proven to be both target reaching and collision-free, concluding the mathematical analysis chapter.

Chapter 5

Simulations

Two different approaches, divided into separate sections, are used to evaluate the proposed navigation strategy (3.8). In the first section, specific scenarios are illustrated and addressed to gain insight into the proposed algorithm's behavior as well as exposing strengths and weaknesses. The second section provides results from multiple Monte Carlo experiments, gaining statistical information regarding the performance of the algorithms. Both sections test the algorithm in single-agent and multi-agent environments. Beside the proposed algorithm (PA), both the Integrated Environment Algorithm from [21] (IEA) and the Reciprocal Velocity Obstacles (RVO) from [24] are implemented and compared to PA. By dividing the chapter in two, the goal is to achieve better insights into the workings of the algorithms as well as providing statistical performance results.

All simulations are, unless stated otherwise, executed with sampling interval $\delta = 0.05s$. The agents are modeled by (2.1), with radius $R_r = 1m$, maximum turning rate $r_{max} = 1\ rad/s$ and maximum linear acceleration $a_{max} = 0.05\ m/s^2$. The sensor range is $d_{sen} = 7m$. Finally, the safe distance is equal to the agent radius, $d_{safe} = 1m$, and is illustrated by the stippled lines around the obstacles in the illustrated scenarios. Note that $R_r = d_{safe}$ results in collisions on physical contact, without further safety

margin. Appendix C provide more information regarding the choice of parameter values. For more information about the work behind creating the simulator, the reader is referred to Appendix E.

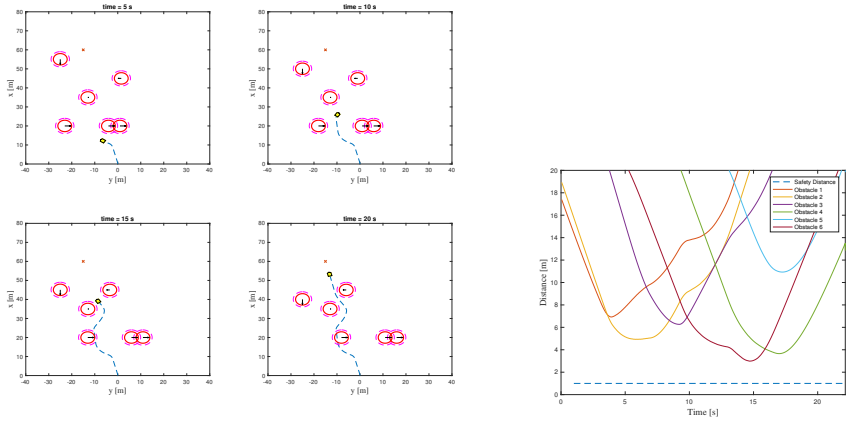
5.1 Simulations

Each path plot shows, as in Figure 5.1a, four separate time-instances of interest. Black arrows on the obstacles indicate linear velocity. The vehicle path is illustrated by a blue stippled line and the target is marked by a cross. Beside the path plot, there is a plot similar to Figure 5.1b, indicating the distance to each obstacle during the simulation. The navigation strategy is considered collision-free as long as the agent maintains the safe margin to all obstacles, marked in the plots by a blue stippled line.

5.1.1 Single-agent environment

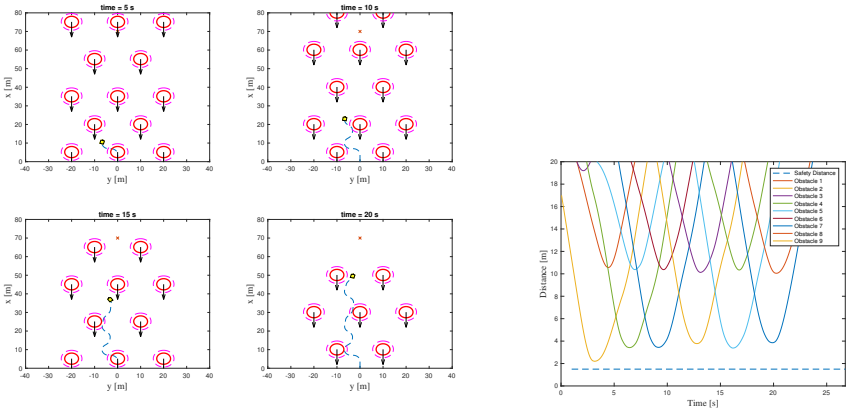
In this subsection, the agent is restricted to constant speed $v_i(t) = V = 3\ m/s$ and operate in environments containing only passive obstacles. Constant speed is used to compare PA to IEA in similar environments. Besides, the braking rule is foremost a tool to solve multi-agent challenges and testing PA with constant speed increases algorithm robustness. In Figure 5.1 the agent navigates the unknown environment consisting of both static and moving obstacles with various speeds. Observe that the agent safely and efficiently navigates the environment and reaches the target without violating the safety margin.

An interesting case is illustrated in Figure 5.2, where obstacles approach the agent at 3 m/s, which is as fast as the agent moves in opposite direction. Nevertheless, the agent safely navigates the environment and reaches the target. From Figure 5.2b it is clear that the agent is now closer to the obstacles compared to Figure 5.1b. In Figure 5.3, a narrow path is taken in a static environment consisting of rectangular obstacles. Considering the first three scenarios, PA successfully navigates both high-speed clut-



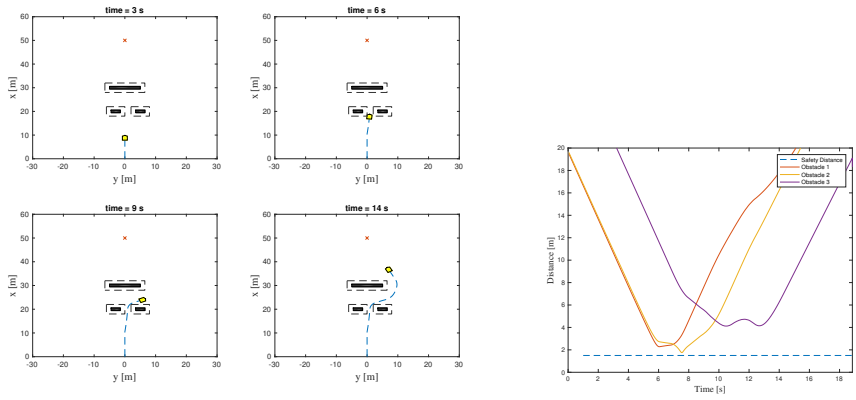
(a) (b)

Figure 5.1: In (a), four time instances with the agent path is shown. Figure (b) provide the distance from the agent to every obstacle during the simulation. The agent navigates a cluttered environment with both moving and static obstacles.



(a) (b)

Figure 5.2: The agent safely navigates the obstacles approaching at speed 3 m/s, which is same speed as the agent in the opposite direction.



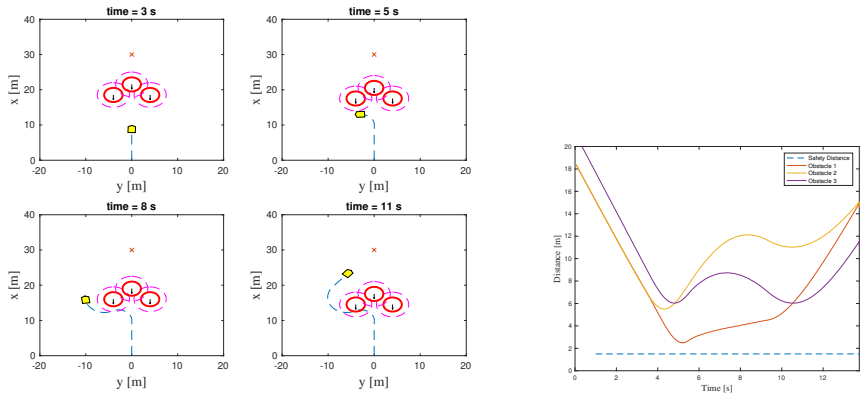
(a) (b)

Figure 5.3: Narrow passage in static maze-like environment.

tered environments and narrow maze-like environments. The scenarios also prove that the algorithm operates safely under conditions that are less strict than assumed in the mathematical analysis.

In Chapter 2, it was mentioned that the algorithm does not assume a specific obstacle shape. This is proven in Figure 5.4 where a concave obstacle approaches the agent at 0.5 m/s. The concave obstacle is created by three overlapping obstacles, where the algorithm reacts to overlapping obstacles as one. Note that the agent is still assumed to measure the velocity of each obstacle separately, which may be difficult in overlapping situations. However, one can argue that overlapping obstacles with different velocities is not a relevant real-world scenario. Figure 5.4 is another example where the algorithm navigates without adhering to the assumptions made in the mathematical analysis.

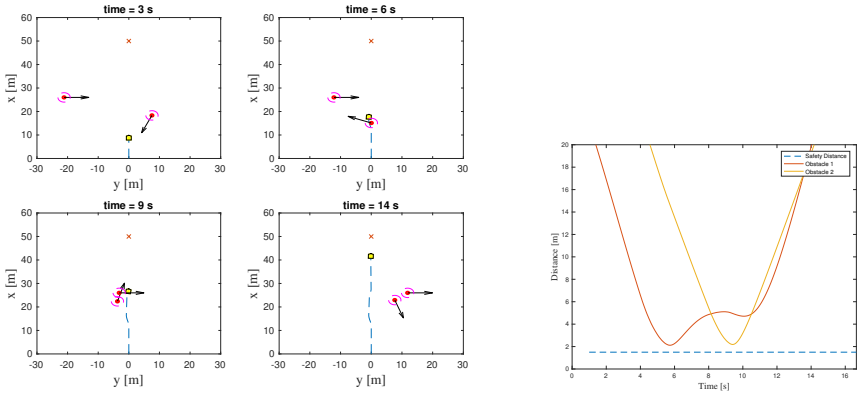
The scenario presented in Figure 5.5, explore the possible weaknesses caused by only considering obstacles in a disk in front of the agent. To do so, two individual obstacles approach the agent at sharp angles. Both obstacles move as fast as the agent. The left obstacle approach in a straight line normal to the agent's velocity and the right



(a) (b)

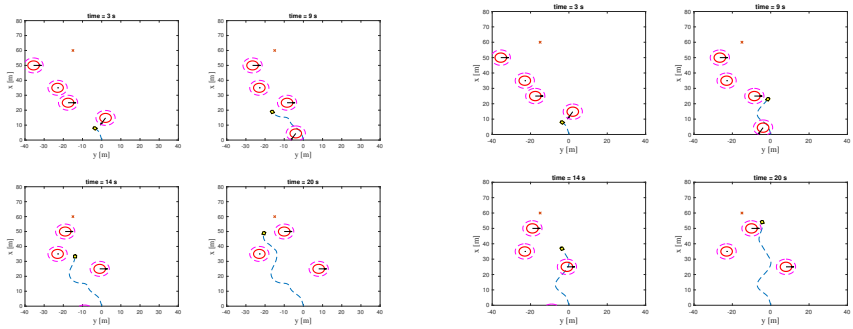
Figure 5.4: Concave approaching obstacle, generated by three overlapping circular obstacles.

obstacle approach the agent from behind in a circular maneuver. The agent does not react to an obstacle outside the sensor disk, thus an obstacle approaching from behind will not immediately be reacted to. From Figure 5.5b one can see that the agent does not violate the safety margin. This makes sense considering that the agent is modeled as point a mass and that the obstacles are extended by the safety margin, relying on the obstacles' velocity not exceeding the agent's velocity. Looking only at the safety distance, no weakness is found with regard to the sensor disk representation and sharply approaching obstacles. Still, one can consider the sensor disk representation a weakness in that it can never be considered safe for obstacles moving faster than the agent, as obstacles approaching from behind are not detected. The scenario also shows that the algorithm handles obstacles moving in curved paths.



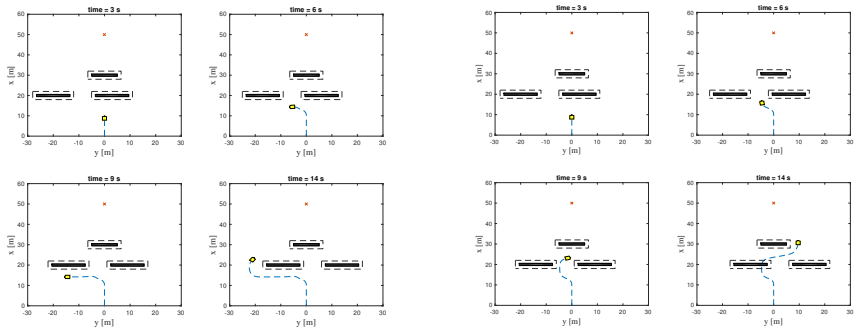
(a) (b)

Figure 5.5: Testing the sensor disk representation by obstacles approaching with high speed from each side. Figure (b) indicate that the scenario is collision-free.



(a) PA (b) IEA

Figure 5.6: Both PA and IEA safely navigates the environment in approximately the same time. PA passes behind the moving obstacles while IEA moves in front of them.



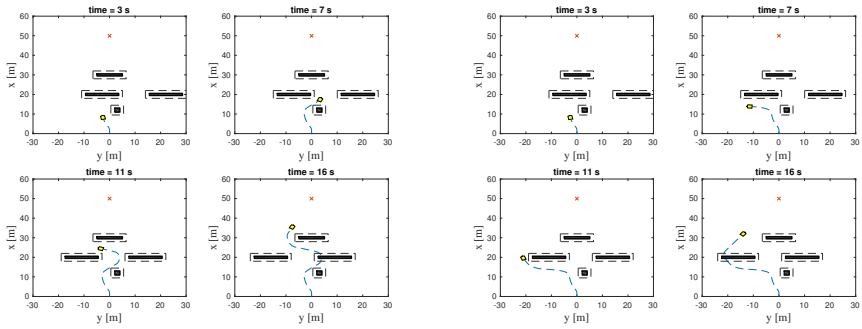
(a) PA (b) IEA

Figure 5.7: The lower obstacles move from left to right. PA(a) moves around the gap while IEA(b) take the narrow passage.

5.1.2 Single-agent environment and IEA

Next, the performance of the proposed algorithm(PA) is compared with the integrated environment algorithm provided in [21], henceforth referred to as IEA. First of all, it is worth noting that PA and IEA is the exact same algorithm for static obstacles when the braking rule is excluded. Lets first look at a cluttered environment with moving obstacles illustrated in Figure 5.6. Both PA and IEA safely navigate the environment and reach the target at approximately the same time. One can argue that PA chooses a safer path as it moves behind the moving obstacles rather than in front, making it generally easier to bypass the obstacles. Using the same argument one can say that PA in some scenarios chooses a more conservative path as shown in Figure 5.7, where IEA navigates through a narrow gap while PA moves around. The desired path in this scenario is application dependent. In Figure 5.8, the result is reversed when the obstacles move in the opposite directions. Considering both scenarios one can argue that the behavior of PA the more desirable, as the average path is shorter by maneuvering behind moving obstacles.

The last single-agent scenario is illustrated in Figure 5.9 and show another example



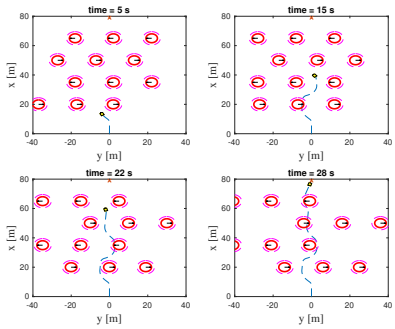
(a) PA (b) IEA

Figure 5.8: The middle obstacles move from right to left. The result is that PA takes the narrow passage and IEA moves around.

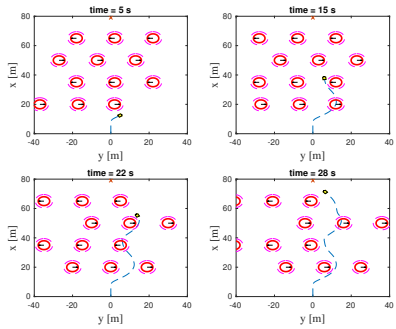
where PA is both safer and more efficient than IEA. Here, the obstacles move in rows from either side with small gaps between the rows. From Figure 5.9a and 5.9b it is clear that the advantage is gained by maneuvering behind the moving obstacles. It is also evident from Figure 5.9c and 5.9d that PA generally maintain a larger distance to the obstacles.

5.1.3 Multi-agent environment

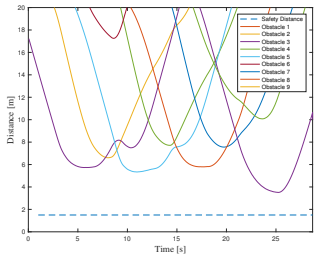
The simulations presented in this section seek to investigate and understand the behavior of the navigation strategy (3.8) in a multi-agent environment. All agents are bounded by the global constraints, $v_{max} = 3\ m/s$, $v_{min} = 1.2\ m/s$, $a_{max} = 0.05\ m/s^2$, $r_{max} = 1\ rad/s$ and $R_r = 1\ m$. The first multi-agent simulation is presented in Figure 5.10, where 40 agents start on a circle with antipodal targets. No collisions occur and the figure illustrate the cluttered nature of the simulation. The simulation is completed within reasonable time, proving the effectiveness of PA. One might expect circular behavior similar to cars in a roundabout in this scenario, it is in fact the non-perfect symmetry and the braking rule that causes this behavior. The symmetry of the scenario is non-perfect in that the agents are discreetly placed on the initial



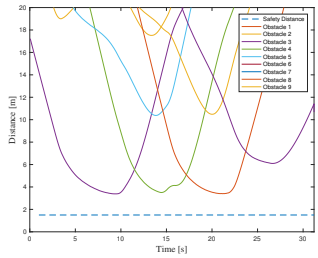
(a) PA



(b) IEA



(c) PA



(d) IEA

Figure 5.9: Comparing PA against IEA in a challenging cluttered scenario. Where PA prove to be both faster and safer.

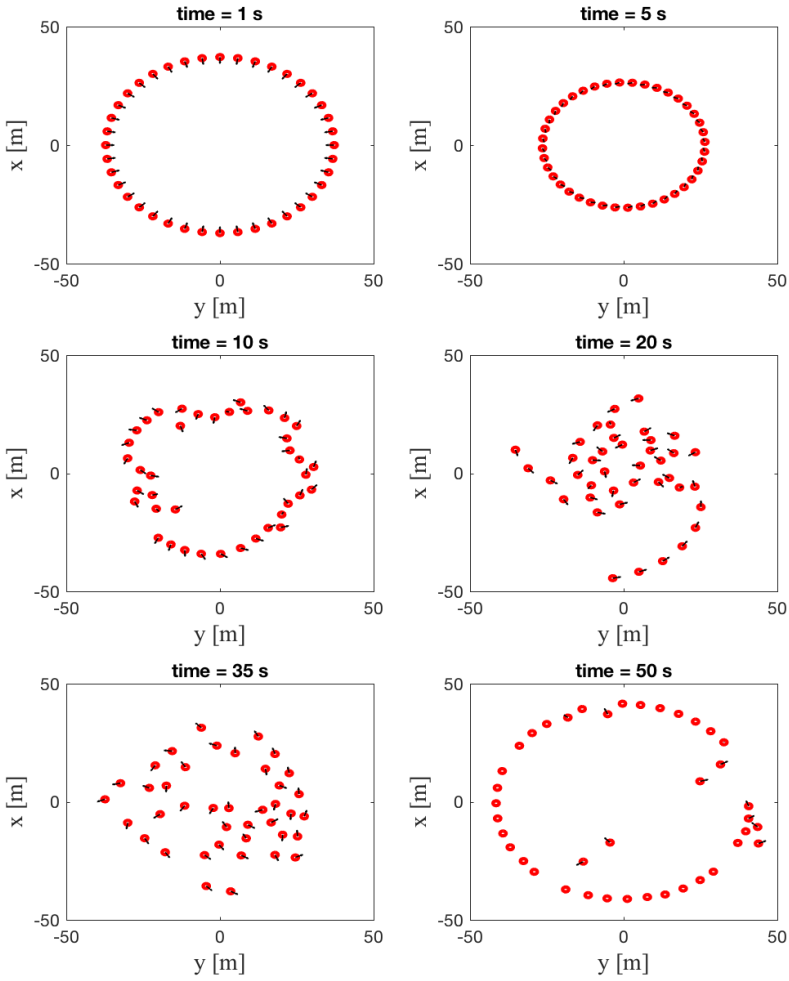
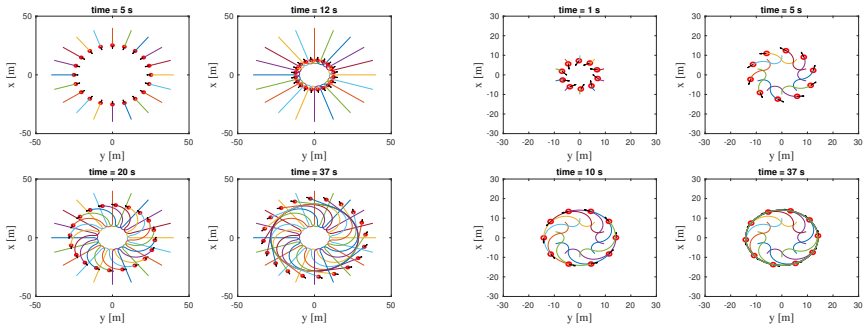


Figure 5.10: Circle scenario with 40 agents having antipodal targets. All agents navigate the cluttered environment without collisions.



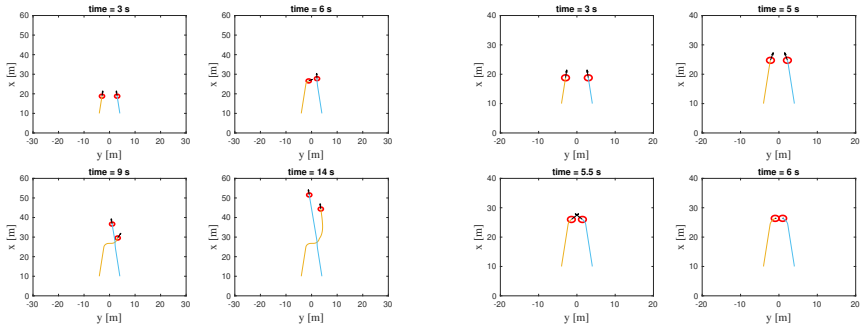
(a) 20 agents on a 40m radius circle

(b) 10 agents on a 10m radius circle

Figure 5.11: Circle scenario without the braking rule. The case in (a) is successful with a roundabout-like behavior. While no collision occur in (b), deadlock hinder the agents to reach their targets.

circle which causes the distance between agents to be slightly different. While the behavior looks chaotic, such behavior is the reason why the presented algorithm does not suffer from deadlocks. To emphasize the effect, Figure 5.11 shows a similar scenario where the braking rule is excluded and the scenario ends in deadlock dependent on the number of agents and the size of the starting circle.

Figure 5.12 present an interesting case emphasizing the importance of the braking rule. Observe that the agents in Figure 5.12a follow the desired yield-pass maneuver, while Figure 5.12b, without the braking rule, ends in a collision. The scenario is equal to the one analyzed in Figure 4.8, where velocity compensation causes two agents to turn toward each other. For completeness, the scenario presented in Figure 5.13 show that the algorithm successfully navigates an environment with multiple agents and both static and dynamic obstacles. The last and most dense scenario is presented in Figure 5.14, where over 200 agents move in rows approaching the center of the simulation. All agents navigates without collision and reaches their target which are located opposite to the initial position.



(a) With the braking rule

(b) Without the braking rule

Figure 5.12: Simulation illustrate the importance of the applied braking rule. In (a) the desired yield-pass behavior is shown, while (b) ends in a collision.

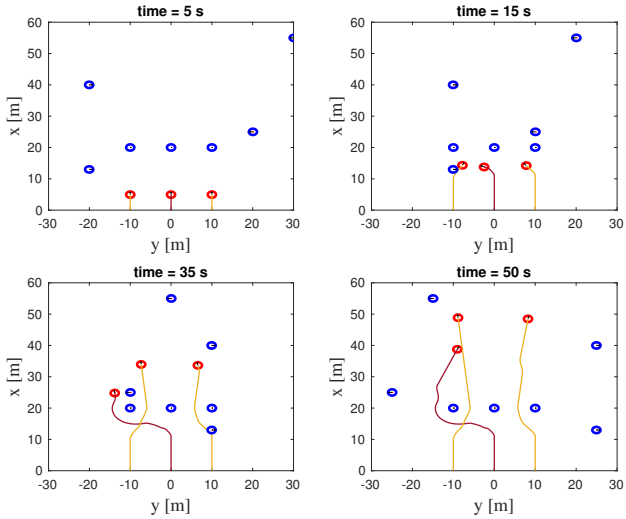


Figure 5.13: Simulation with three agents (red) in an environment containing both static and moving obstacles (blue). The moving obstacles has same forward speed as the agents.

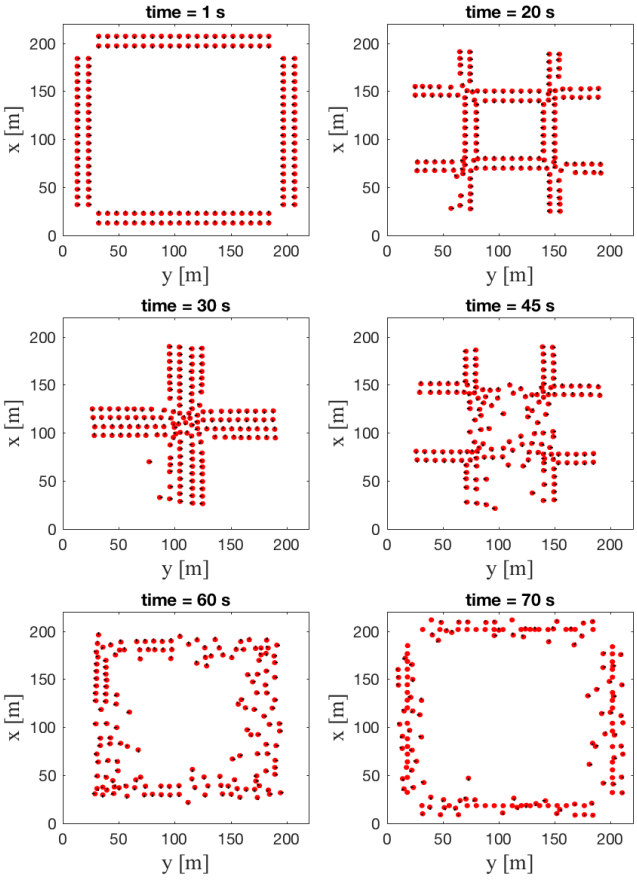
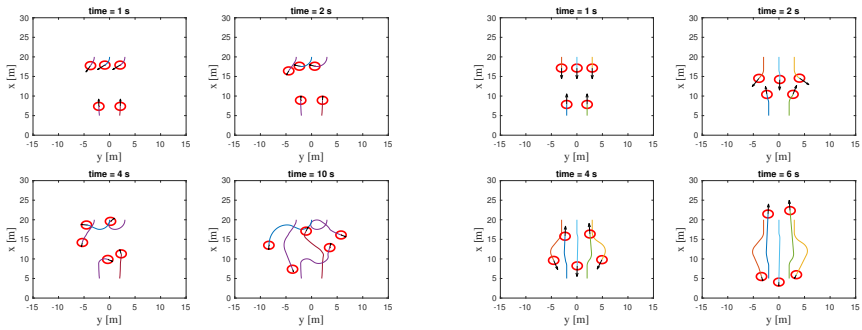


Figure 5.14: This scenario contains over 200 agents with targets on the opposite side of the initial position. The scenario is effectively simulated without collisions.

5.1.4 Multi-agent environment and RVO



(a) PA (b) RVO

Figure 5.15: A simple scenario illustrating the different behavior of PA and RVO in a passing maneuver.

Figure 5.15- 5.17 show three different scenarios where the agents are guided by the RVO algorithm. One of the significant differences in behavior between PA and RVO is the choice of preferred velocity. PA chooses the closest valid velocity to the current velocity, while RVO chooses the closest valid velocity to the maximum speed toward the target. The result can be observed in Figure 5.15, where PA aims at avoiding the approaching agents while RVO navigates more efficiently. Another example of the same effect can be seen in Figure 5.16, where agents pass each other closer when guided by RVO compared to PA in Figure 5.10. The close passage results in a more efficient path. Note that RVO requires a sensor range of 20 m in order to successfully navigate the scenario. Finally, RVO is tested in a scenario similar to that in Figure 5.12. The result is collision-free, but not target reaching, see Figure 5.17. In the scenario, the agents block each other and thus does not reach their targets and shows that RVO might be more susceptible to deadlocks than PA.

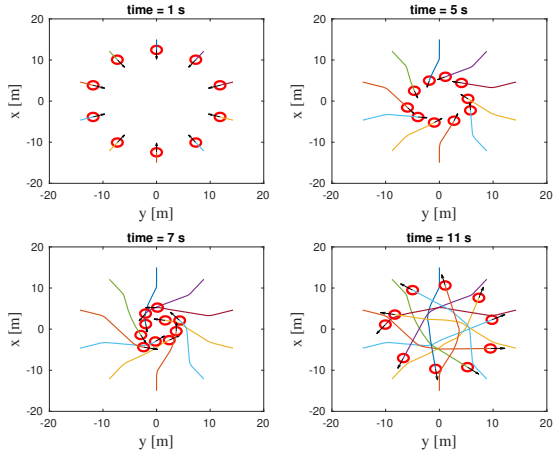


Figure 5.16: Circle scenario performed with 10 agents and a sensor range of 20m to avoid collision. The scenario illustrate the efficiency of RVO

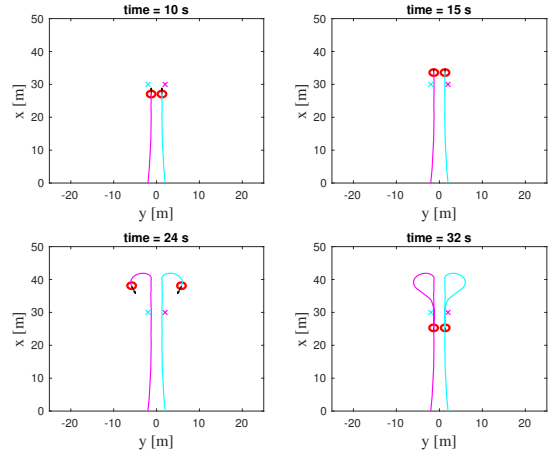


Figure 5.17: Scenario that is comparable to the scenario in Figure 5.12 and depict a possible challenge with RVO algorithm.

5.2 Monte Carlo experiments

By investigating specific scenarios, the previous section addressed behavior understanding of the three algorithms at hand. In this section, statistically performance results of the PA, the IEA and RVO are presented. Using Monte Carlo[1] experiments, the three algorithms have been tested in thousands of simulations for randomly given initial conditions[2]. Four output variables are produced in each experiment to measure the performance of the algorithms:

- Percentage of successful runs
- Percentage of runs where collision occurred
- Percentage of runs that was collision-free but not target reaching, e.i runs that timed out
- Average time to target, calculated using the successful runs

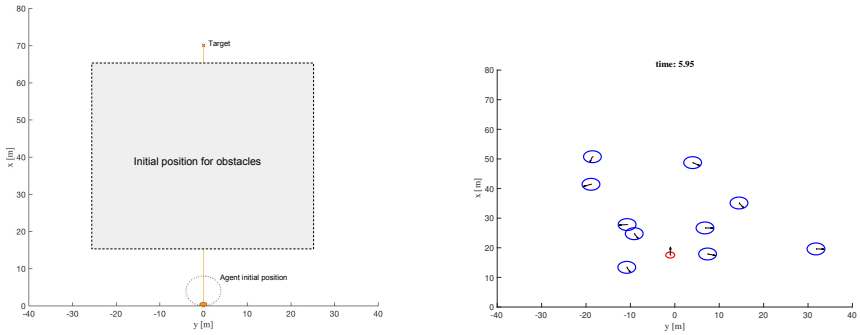
A successful run is a collision-free simulation where all agents reach their targets. For further insight into the choice of parameter values and their impact on system performance, see Appendix C.

5.2.1 Single-agent environment

For single-agent environments, three different experiments of 1000 simulations are tested for both PA and IEA and proposed in this thesis. The result is summarized in Table 5.1-5.3. To ensure a collision-free initialization, a setup like the one illustrated in Figure 5.18a is used. The gray area defined by $x \in [15, 65]$ and $y \in [-25, 25]$, indicate the area where obstacles are randomly initialized. In all cases, the initial position of the agent is $\boldsymbol{p}(0) = (0, 0)$ and the stationary target is located at $\boldsymbol{p}_\tau(t) = (70, 0)\ \forall t$. The maximum time duration of a simulation is 65 seconds and the agent forward speed is constant $v(t) = V = 3\ m/s$, with maximum turning rate $|r_{max}| = 1\ rad/s$. The initial

[1] https://en.wikipedia.org/wiki/Monte_Carlo_method

[2] Remark: The randomly generated initial conditions are seeded and saved, such that all experiments can easily be reproduced.



(a) (b)

Figure 5.18: In (a) the initial position boundary is illustrated. Figure (b) show a typical 10 obstacles initial scenario taken from the simulation in Table 5.1.

heading of the obstacles is in the range $(\frac{\pi}{2}, \frac{3\pi}{2})$ to force interaction with the agent and prevent obstacles from actively blocking the target. In the experiment with 10 obstacles from Table 5.1, 93.1% of the simulations engaged in collision avoidance, while 98.1% did in the experiments with 15 obstacles from Table 5.2, proving that the environment is in fact cluttered. Figure 5.18b shows a typical scenario with the setup used in Table 5.1. It is important to note that both algorithms are tested on the exact same environment with the same initial conditions for all of 1000 simulations, hence the results are comparable.

Table 5.1: Experiment consist of 10 circular obstacles with radius 2 m and speed 2 m/s.

Algorithm	Success	Collision	Timed out	Average time
PA	98.0%	2.0%	0%	23.32s
IEA	93.4%	6.6%	0%	24.53s

For the scenarios tested in Table 5.1, PA outperforms IEA in both safety and efficiency. By testing 1000 simulations created with random initial conditions, the results are more unbiased and in coherence with the results from in Section 5.1. The

Table 5.2: Experiment consist of 15 circular obstacles with radius 2 m and speed 2 m/s.

Algorithm	Success	Collision	Timed out	Average time
PA	95.9%	4.1%	0%	23.66s
IEA	86.4%	13.6%	0%	26.12s

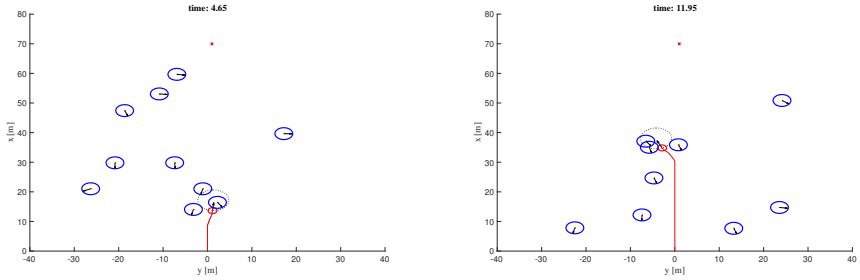
Table 5.3: Experiment with 8 obstacles, where the obstacle speed, 4 m/s, is higher than agent speed, 3 m/s.

Algorithm	Success	Collision	Timed out	Average time
PA	80.9%	19.1%	0%	22.59s
IEA	66.3%	33.7%	0%	24.03s

experiment with results presented in Table 5.2 are from a more densely cluttered environment containing 15 obstacles in the same space as in Table 5.1. Here, PA shows a clear advantage especially with regard to the number of collisions. In the last single-agent experiment the obstacles have higher speed than the agent i.e. 4 m/s. As expected, this is a scenario where PA stand out positively, which make sense considering examples such as Figure 5.9. This is also in coherence with earlier results, and again show that the algorithm works for conditions outside the assumptions made in Chapter 4. Figure 5.19 presents two cases from Table 5.1 where PA collides. The collision in these cases occurs because the agent becomes surrounded by obstacles, which is unavoidable in a cluttered scenario using a local method with limited sensor range and with velocity constraints.

5.2.2 Multi-agent environment

The experiments in this section are conducted in multi-agent environments, where both the algorithm presented in this thesis(PA) and the Reciprocal Velocity Obstacles(RVO) described in Section 1.2.1 are tested. To prevent initial collision and target blocking, all initial positions and targets are separated by at least a distance d_{init}. Contrary to the



(a) (b)

Figure 5.19: Two collision cases for PA taken from the experiments in Table 5.1. The red circle is the agent with path and a red target, the black dotted circle is the sensor disk and blue indicate obstacles.

single-agents experiments, the initialization in this section is randomized for all agents in the complete given area. Results from the experiments are given in Table 5.4-5.6. The first experiment presented in Table 5.4 show the effect of the braking rule in a cluttered environment with 12 agents of radius $R_r = 1m$ and maximum forward speed $v_{max} = 3\ m/s$. By comparing the results with Table 5.1, it is evident that the algorithm works well in a multi-agent system with more difficult initial conditions. The low percentage in Timed out and Average time indicate that the algorithm overcomes multi-agent challenges such as oscillation and deadlock. The braking rule is clearly a positive contribution to the algorithms safety. Next, is an experiment with both agents and passive obstacles, see Table 5.5. This is an interesting case to compare the performance of PA and RVO due to their difference in handling passive obstacles. With 7 agents and 4 obstacles, the experiment is similar to that in Table 5.4 with regard to obstacle density. As the table show, PA is safer with fewer collisions, while RVO is more efficient with lower average completion time. Hence, the results are in coherence with the discussion from Section 5.1.4

Finally, Table 5.6 present results where RVO and PA are tested using the same conditions in a clutters multi-agent environment. There are three experiments, where

Table 5.4: Experiment with 12 agents in an 50x50m area, $r_{max} = 1\ rad/m$ and $v_{max} = 3\ m/s$. The scenario is tested with and without the braking rule.

Algorithm	Success	Collision	Timed out	Average time
With the braking rule	98.0%	1.1%	0.9%	20.70s
Without the braking rule	93.1%	6.5%	0.4%	20.52s

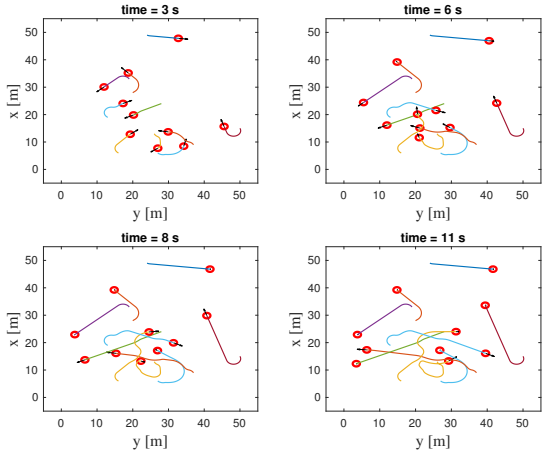
Table 5.5: Experiment with 7 agents and 4 obstacles in an 50x50m area, $r_{max} = 1\ rad/s$, $v_{max} = 3\ m/s$ and $v_{o,max} = 2\ m/s$.

Algorithm	Success	Collision	Timed out	Average time
PA	97.0%	2.6%	0.4%	20.15s
RVO	93.4%	6.6%	0.0%	15.28s

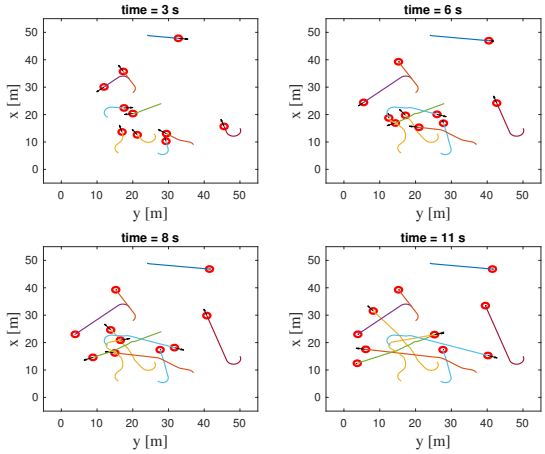
Table 5.6: Three experiment with 10 agents in an 50x50m area for 1000 simulations each and with $d_{init} = 4m$.

Algorithm	Success	Collision	Timed out	Average time
$v_{max} = 2\ m/S$, $r_{max} = 1\ rad/s$				
PA	99.8%	0.0%	0.2%	27.49s
RVO	96.6%	3.4%	0.0%	23.32s
$v_{max} = 3\ m/S$, $r_{max} = 1\ rad/s$				
PA	99.8%	0.2%	0.0%	18.97s
RVO	90.2%	9.8%	0.0%	16.09s
$v_{max} = 3\ m/S$, $r_{max} = 3\ rad/s$				
PA	100.0%	0.0%	0.0%	17.74s
RVO	99.4%	0.4%	0.0%	15.33s

the only differences are the variables r_{max} and v_{max}. RVO does not originally consider nonholonomic constraints, hence the relation between forward speed and turning rate is a critical factor for its' success. As the table show, PA is safer in all cases, and significantly so for the case with large $\frac{v_{max}}{r_{max}}$ ratio. RVO is more efficient in all cases with lower average completion time. Considering that RVO uses bearing to the target as an incorporated variable in its' collision avoidance algorithm, it can be expected that RVO is more efficient. PA, on the other hand, switches between collision avoidance and target reaching, where bearing to the target is neglected during collision avoidance. Nevertheless, the results show that the efficiency comes at the cost of safety. Figure 5.20 present a typical scenario taken from the first simulation in Table 5.6. The general behavior difference is that RVO is less conservative, allowing to be closer to other agents, while PA maintains a larger distance to other agents, resulting in longer paths.



(a) PA



(b) RVO

Figure 5.20: Typical simulation for both PA and RVO taken from the first experiments in Table 5.1.

Chapter 6

Conclusions and Future Work

This chapter present results, discussion and concluding remarks. Further work is stated at the end of the chapter, proposing possible algorithm extensions and tasks outside the scope of this thesis.

6.1 Result discussion

In the single-agent scenarios presented in Section 5.1.1, the algorithm safely navigates the environment without adhering to the assumptions made in Chapter 4. Furthermore, the results from Section 5.1.2 and the Monte Carlo experiments from Section 5.2.1 show that the proposed algorithm (PA) outperforms the Integrated Environment Algorithm from [21] (IEA) in both safety and efficiency. As mentioned in Section 5.1.2, the algorithms are the same for stationary obstacles when the braking rule is not applied. For moving obstacles in a dynamic environment, the general result is that the PA choses both safer and shorter paths by passing behind moving obstacles. There are also examples where PA is more conservative than IEA, however, still considered safer.

In the simulations there are two main collisions causes; hard initial conditions and "impossible situations". Hard initial conditions arise when the agent and an obstacle

start on collision course close to each other. "Impossible situations" are scenarios where the agent become surrounded by obstacles and cannot choose a collision-free velocity. Such situations are unavoidable in a priori unknown and cluttered environments with limited sensor range and bounded velocity.

Contrary to using all available measurements around the agent to avoid obstacles, the sensor disk representation can seem counter-intuitive. It can be understood as a measurement pre-processing method. Using the sensor disk representation, the agents avoids fast obstacles aiming for the agents in an effective manner, see Figure 5.5. There is also the advantage of only looking for obstacles in an area where the agents are required to take action in order to avoid a collision and otherwise ignore the obstacles. However, there is a disadvantage in the fact that the algorithm cannot be considered safe for obstacles moving faster than the agents.

Looking at multi-agent environments, the PA successfully navigates scenarios with hundreds of agents without collision. The scenarios are effectively simulated by exploiting the local nature of the algorithm, making it suitable for parallel computation. Typical scenarios that are prone to deadlocks and oscillations are addressed. PA successfully navigates the scenarios where the braking rule proves to be of vital importance. In Section 5.2.2 the performance of PA is compared to Reciprocal Velocity Obstacles (RVO) in multiple cluttered environments. The results show that PA is generally safer, while RVO is more effective. The difference in efficiency can be explained by the fact that RVO uses bearing to target when choosing a new velocity, resulting in closer passage to other agents. Note that the original RVO do not consider nonholonomic constraints. It is evident from the results in Table 5.6 that the success of RVO is heavily dependent on the minimum turning radius. The simulations also prove that PA performs better for short sensor ranges, making it suitable as a "last measure" avoidance algorithm in a navigation control system. RVO on the other hand, works better (compared to PA) with longer sensor range, making it more versatile. Still, one can argue that the most important feature of a reactive collision avoidance algorithm is collision-free guidance, where PA outperforms RVO.

As a side-note, it is worth mentioning why bearing to the target, $\psi_d(t)$, is not included in the collision avoidance of the proposed algorithm. RVO proves that including the angle can result in a more efficient strategy. The two reasons why $\psi_d(t)$ is not included in the collision avoidance is: One, choosing the closest obstacle-free area to the current heading, as in PA, is arguably the safer among the two possibilities. Least effort to safety is considered safer than minimizing the distance to the target. Two, including $\psi_d(t)$ and making a less conservative move at the current time instant does not guarantee a more efficient path. Using the fact that the environment is unknown, one cannot a priori know which of the two choices that ultimately lead to the most efficient path.

6.2 Conclusion

The reactive navigation algorithm presented in this article assume a nonholonomic agent model without a priori knowledge about the environment. No communication is required and the algorithm is suitable for environments with multiple agents as well as passive obstacles. Agents and obstacles can be of any shapes. The information available to the agents are based on an integrated representation of the environment, compensated with obstacle velocity.

The main features of the proposed algorithm are:

- Safe navigation in complex environments
- Multi-agent navigation without oscillations and deadlocks
- Low computational complexity
- Require only local measurements
- Show promising result compared with other algorithms

Thus the algorithm is suitable for a wide range of vehicles, including vehicles with heavy constraints on linear acceleration and fast vehicles that operate in rapidly changing environments. The algorithm has potential a weakness concerning fast approaching obstacles from behind and also seem to be less efficient regarding target reachability. Mathematical analysis is an important addition to support the safety qualification of the algorithm and is given in Chapter 4. Nevertheless, the algorithm work under conditions that are less strict compared to those provided in the mathematical analysis, as shown in Chapter 5. Appendix C provide insight into the influence system parameters have on the overall performance and contribute to algorithm robustness. The Monte Carlo experiments given in Section 5.2 provide random scenarios for PA, IEA and RVO, where PA seem to be the safest in both single- and multi-agent scenarios. Furthermore, real-time performance can be achieved in such scenarios, as an independent computation is performed for each agent. Dividing the simulation chapter into two main sections provide both understanding of the algorithm behavior, as well as statistical performance results.

6.3 Further work

The original Reciprocal Velocity Obstacle algorithm does not consider nonholonomic constraints, there are however extended versions of RVO that do, such as [25] and [29]. Hence, it would have been interesting to compare the algorithm to a more advanced version of RVO that adhere to the same constraints. Aside from computer simulations, it will eventually be important to test the algorithm on an actual vehicle in an experimenting environment. Such an experiment would test whether the assumptions made in the simulations can be applied to a real-world scenario.

Throughout this thesis, it is assumed that the agents measure the obstacles' position and velocity without error. The real world is not that simple and it is thus a potential weakness that is not yet explored. It would be interesting to research the effect noise and occasional loss of measurements have on the safety and performance of the navigation strategy. As mentioned in Chapter 2, it is assumed that the safety extended

environment is sensed directly. In order to obtain $\hat{E}(t)$ in a physical application, the actual sensed objects, $E(t)$, must be extended by calculating a safety compensation angle based on d_{safe}. Furthermore, the agent model used in this thesis is suitable for describing a large variety of vehicles, it is also fairly simple. In further work, one should consider developing a more complex model that incorporate higher order physical and dynamical terms.

Appendix A

Seeking a Path Through the Crowd by Savkin and Wang

In this appendix, the algorithm from [21] is presented to clearly separate the acquired background from the contributions provided by this thesis. The vehicle and obstacle models are similar and the focus is thus directed towards the integrated sensor representation and the collision avoidance algorithm. To ease readability, the algorithm is presented following the notation used throughout this thesis. The algorithms' main differences are the introduced velocity compensation, braking rule and multi-agent environment. For a complete algorithm description, the reader is referred to the original paper [21].

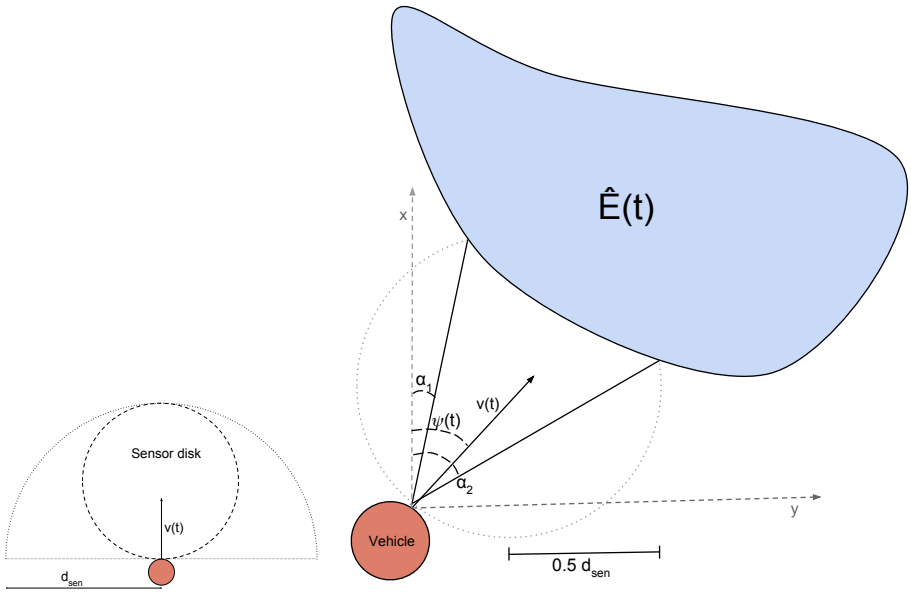
A.1 Integrated sensor representation

A sensor located on a vehicle emit rays in directions denoted α in a half circle defined by $[\psi(t) - \frac{\pi}{2},\ \psi(t) + \frac{\pi}{2}]$ and range d_{sen}, where $\psi(t)$ is the vehicle heading. The sensor disk $\mathcal{D}(t)$ is as a circle in front of the vehicle with diameter d_{sen}, where the vehicle

detects all obstacles inside the disk

$$\mathcal{D}(k\delta) = d_{sen}\ cos(\alpha), \quad \forall\ \alpha \in [\psi(k\delta) - \frac{\pi}{2},\ \psi(k\delta) + \frac{\pi}{2}]$$

at $t = k\delta$. See Figure A.1.



(a) Sensor ability and sensor disk (b) A vehicle sensing an obstacle

Figure A.1: Illustration of the sensor disk $\mathcal{D}$ and the vehicle sensing ability.

The binary function representing the integrated environment is defined as

Definition 5. *A binary function* $M(\alpha, t) \in \{0, 1\}$ *is defined for all* $t \geq 0$ *and* $\alpha \in [\psi(t) - \frac{\pi}{2},\ \psi(t) + \frac{\pi}{2}]$ *as*

$$M(\alpha, t) = \begin{cases} 1, & if \quad d_{\boldsymbol{p}_E} \leq d_{sen} cos\alpha \\ 0, & otherwise \end{cases} \tag{A.1}$$

where $d_{\boldsymbol{p}_E} := \|\boldsymbol{p}_i(t) - \boldsymbol{p}_E\|$ is the distance to the point, $\boldsymbol{p}_E$, where the ray emitted from the vehicle at time t in direction α hits $\hat{E}(t)$. $\|\cdot\|$ is the standard Eucledian vector norm. See Figure A.2 for an example.

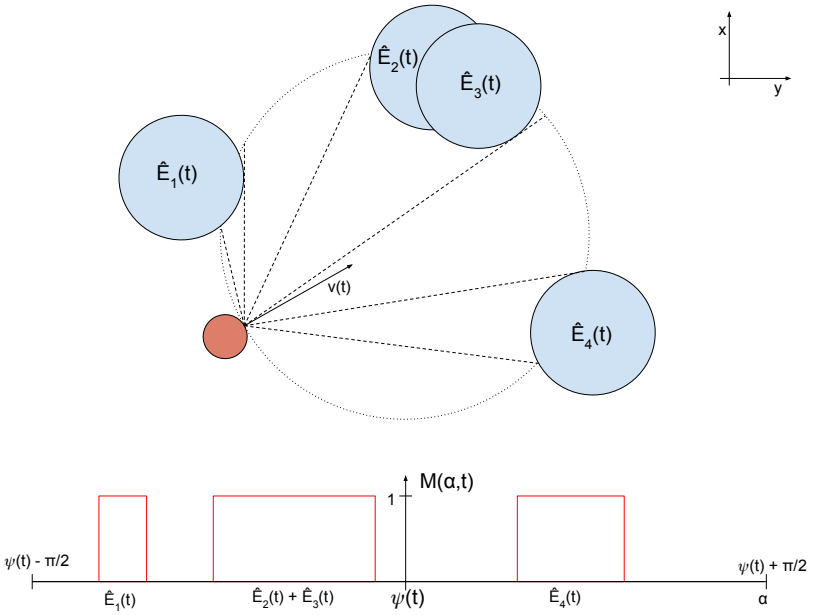


Figure A.2: Example of $M(\alpha, t)$ showing multiple and overlapping obstacles.

A.2 Collision avoidance algorithm

A new binary function $m(t)$ is defined as

$$m(t) := \begin{cases} 0, & if\ M(\alpha,t) = 0\ \forall\ \alpha \in [\psi(t) - \frac{\pi}{2},\ \psi(t) + \frac{\pi}{2}] \\ 1, & otherwise \end{cases} \tag{A.2}$$

for $m(t) = 1$ there are one or more obstacles inside the sensor disk at time t and $m(t) = 0$ if the sensor disk is obstacle-free.

Consider again Figure A.2, where an example of $M(\alpha, t)$ is given in a scenario with four obstacles. In this scenario $m(t) = 1$ and there exist some $\alpha \in [\psi(t) - \frac{\pi}{2},\ \psi(t) + \frac{\pi}{2}]$ where $M(\alpha, t) = 0$, illustrated by the open intervals $[A_i^-,\ A_i^+]$ for $i = 1, 2, 3, 4$. Next, a new heading is chosen as the closest obstacle-free interval to the current heading and the middle value $C(t)$ of the interval is calculated by

$$j_i(t) := arg\ \min_i\{|A_i^-|,\ |A_i^+|\} \tag{A.3}$$

where $j_i(t)$ is the index of the A_i^- or A_i^+ that is closest to $\psi(t)$. The middle value is found by:

$$C(t) = \frac{A_{j_i(t)}^- + A_{j_i(t)}^+}{2} \tag{A.4}$$

where A_3^- is the closest start/end of an interval in the example shown in Figure A.2. Thus $j(t) = 3$ and $C(t)$ is the middle of the interval $[A_3^-,\ A_3^+]$. Note that $C(t)$ is an angle.

The complete collision avoidance algorithm is given as

$$r(t) := \begin{cases} r_{max}\ \mathrm{sgn}(\psi_\tau(t) - \psi(t)), & if\ \hat{m}_i(t) = 0 \\ r_{max}\ \mathrm{sgn}(C(t) - \psi(t)), & if\ \hat{m}_i(t) = 1 \end{cases} \tag{A.5a}$$

where $sgn(\cdot)$ returns the sign of its argument and $\psi_\tau(t)$ is the angle to to target. The input $r(t)$ is constant in the interval $[t,\ \ t + \delta]$. The navigation algorithm is fairly simple; if there are one or more obstacles in the sensor disk, the vehicle turns as fast as possible towards the closest obstacle-free zone. Otherwise the vehicle turns as fast as possible toward the target. Note that it is assumed that the vehicle forward speed is constant $v(t) = V$.

Appendix B

The Velocity Compensation Angle

In Section 2.3 the velocity compensated binary function $\hat{M}(\hat{\alpha}, t)$ was introduced. This appendix derive the mathematics behind calculating the compensation angle and follows the lines of the original paper [23]. Equation (2.5) gave an expression for how to compensate the rays used in $M(\alpha, t)$ to obtain $\hat{M}(\hat{\alpha}, t)$.

$$\hat{\alpha}_i = \alpha_i + \gamma_{ab,i}, \quad for\ i = 1, 2 \tag{B.1}$$

Now a method is presented to find correction angle $\gamma_{ab,2}$ as illustrated in Figure B.1. Here $\gamma_{ab,2}$ is the angle between a vector $\boldsymbol{a}_2$ along the α_2 ray and a new corrected vector $\boldsymbol{b}_2$, where $\|\boldsymbol{b}_2\| = v_i(t)$. The second ray is considered in this derivation, but the exact same theory applies to the first ray α_1.

Step one is to find the angle, $\gamma_{v,2}$ between $\boldsymbol{a}_2$ and $\boldsymbol{v}_{\hat{E}}$. Then apply the sine rule to calculate $\gamma_{ab,2}$ and obtain $\hat{\alpha}_2$. From Figure B.1 $\gamma_{v,2}$ is found by

$$\gamma_{v,2} = \pi - (\psi_v - \alpha_2) \tag{B.2}$$

The length of booth $\boldsymbol{b}_2$ and $\boldsymbol{v}_{\hat{E}}$ are known, thus the sine rule is used to find $\gamma_{ab,2}$.

$$\gamma_{ab,2} = \sin^{-1}\left(\frac{\|\boldsymbol{v}_{\hat{E}}\| \sin(\gamma_{v,2})}{v_i(t)}\right) \tag{B.3}$$

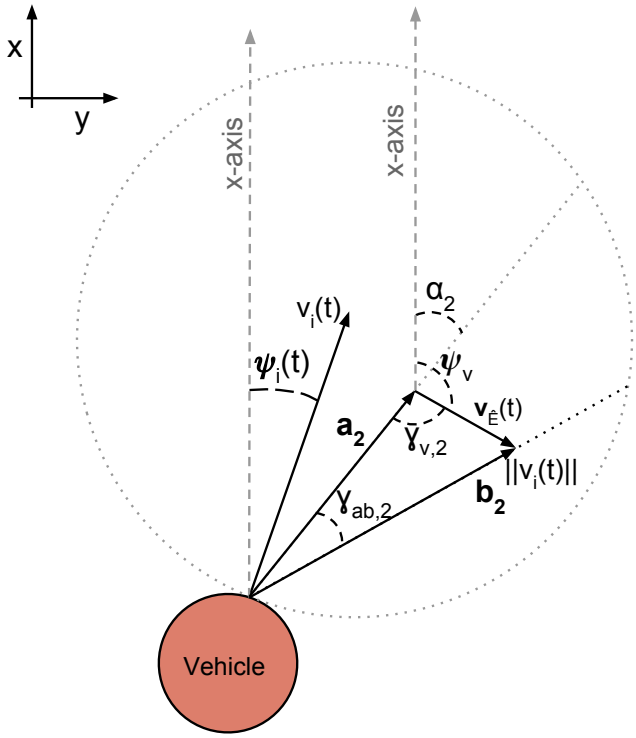


Figure B.1: Velocity compensation where $\alpha_2 + \gamma_{ab,2}$ is the corrected angle of the second ray defining the compensated collision zone.

Appendix C

Additional Monte Carlo Experiments

In these additional Monte Carlo experiments, the system parameters used in Chapter 5 are tested. The goal is to present the influence each parameter value has on the algorithms overall performance. The parameters used in the below simulations are, if not otherwise stated, given by: $v_{max} = 3\ m/s$, $v_{min} = 1.2\ m/s$, $a_{max} = 0.05\ m/s^2$, $r_{max} = 1\ rad/s$ and $d_{sen} = 7\ m$.

Table C.1: Choosing r_{max}: 12 obstacles in a 50x50m environment, where $T_{stop} = 60sec$ and number of simulations is 1000. It is worth noting that a very high r_{max} is physically impossible and, in a way, cancels out the nonholonomic model. A too low r_{max} hinder the maneuverability of the agents.

$r_{max}\ [rad/s]$	Success	Collision	Timed out	Average time
5	99.7%	0.0%	0.3%	19.30s
3	99.3%	0.2%	0.5%	19.66s
1	98.1%	1.2%	0.7%	20.93s
0.5	70.5%	20.2%	9.3%	25.95s

Table C.2: Choosing v_{min}: 12 obstacles in a 50x50m environment, where $T_{stop} = 60sec$ and number of simulations is 1000. As the results show the forward speed controller have only a small influence on the overall result. Note that a higher a_{max} would increase the effects as can be seen in Table C.3.

% of v_{max}	Success	Collision	Timed out	Average time
100%	97.2%	2.3%	0.5%	20.85s
80%	97.3%	2.0%	0.7%	20.87s
60%	97.6%	1.5%	0.9%	21.19s
40%	97.9%	1.3%	0.8%	21.12s
20%	97.9%	1.3%	0.8%	21.15s

Table C.3: Choosing a_{max}: 12 obstacles in a 50x50m environment, where $T_{stop} = 60sec$ and number of simulations is 1000. Naturally there is a decrease in number of collision as a_{max} increase. It is interesting to note that $a_{max} = 0.5\ m/s^2$ have worse performance than $a_{max} = 0.1\ m/s^2$ and is seemingly too high. Note also the variations in number of Timed out cases.

a_{max} [m/s^2]	Success	Collision	Timed out	Average time
0.5	97.6%	0.3%	2.1%	20.68s
0.1	99.5%	0.2%	0.3%	20.68s
0.05	98.3%	1.1%	0.6%	20.88s
0.025	86.3%	3.0%	10.7%	22.09s
0.01	94.0%	4.1%	1.9%	24.81s

Table C.4: Choosing d_{sen}: 6 obstacles in a 25x25m environment, where $T_{stop} = 60sec$ and number of simulations is 500. As expected, larger sensor range d_{sen} leads to less collisions. But, as the table show, there is a trade-off between collisions and average time to reach the target.

d_{sen}	Success	Collision	Timed out	Average time
13m	70.8%	0.6%	28.6%	31.77s
11m	89.8%	0.6%	9.6%	25.38s
9m	93.4%	1.0%	5.6%	19.06s
7m	96.8%	1.8%	1.4%	14.01s
5m	89.8%	9.8%	0.4%	10.62s

Table C.5: Choosing δ: 6 obstacles in a 25x25m environment, where $T_{stop} = 60sec$ and number of simulations is 500. The results clearly show that the time interval has critical influence on system performance.

δ [sec]	Success	Collision	Timed out	Average time
0.5	64.8%	35.2%	0.0%	13.39s
0.1	88.0%	3.4%	8.6%	13.89s
0.05	96.8%	1.6%	1.6%	13.38s
0.025	99.6%	0.2%	0.2%	13.87s
0.01	98.8%	0.0%	1.2%	13.78s

Appendix D

Algorithm by Erlend Hårstad

Erlend Hårstad has, in his thesis, independently addressed the same problem as researched in this thesis. The algorithm produced by Erlend is presented here in short terms and is compared to the proposed algorithm(PA). The comparison is completed in a Monte Carlo experiment with one thousand simulations.

D.1 Algorithm description by Hårstad

This collision cone [34] based reciprocal collision avoidance algorithm uses an extended collision cone which considers the nonholonomic constraints of vehicles [35] to determine possible collisions. The algorithm is designed for vehicles with constant forwards speeds such as large ships which have a limited speed envelope due to high mass. As the main focus is vessels at sea, the reciprocal collision avoidance algorithm is designed such that the vehicles respect the Collision Regulation [36] defined by the International maritime organization. Thus, the collision avoidance maneuver is carried out in a predictable manner. To make use of the reactive nature of other vehicles, the collision cone defined in [35] is reduced such that the responsibility of avoiding a collision is shared equally between the involved vehicles.

D.2 Results

The results from the Monte Carlo experiment are summarized in Table D.1. Besides the difference in the number of collisions, the algorithm proposed by Hårstad is prone to Reciprocal dances when the environment becomes too cluttered. The typical behavior of the reciprocal dances is illustrated by circular paths in Figure D.1. Similar to the comparison with RVO, the algorithm produced by Hårstad seem to create more efficient paths while PA maintains a larger distance to other agents. An example scenario taken from the experiments presented in Table D.1 is given in Figure D.2b. Note that both algorithms are restricted to constant forward speed in the simulations.

Table D.1: Simulations conducted with 10 agents in a $50x50$ m environment. The agents are restricted to constant forward speed 1 m/s. The initial agent separation is 7 m and the maximum simulation time is 150 s.

δ $[sec]$	Success	Collision	Timed out	Average time
PA	99.9%	0.1%	0.0%	51.7s
Hårstad	89.8%	5.8%	4.4%	56.9s

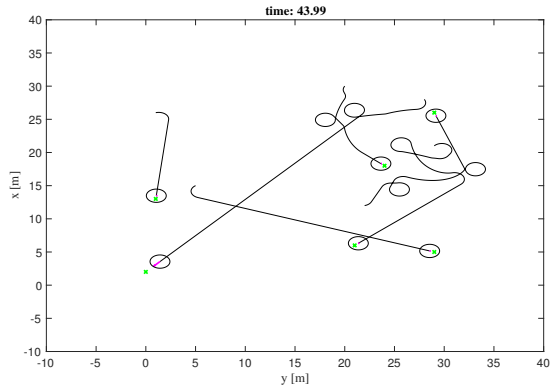
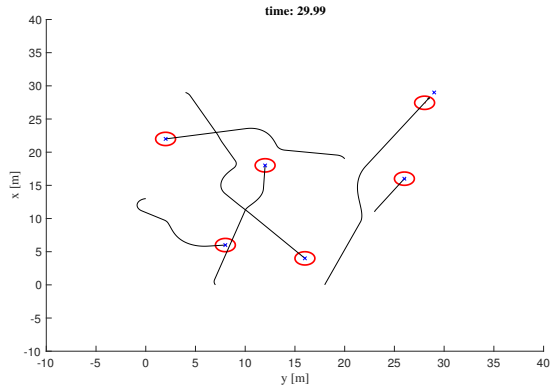
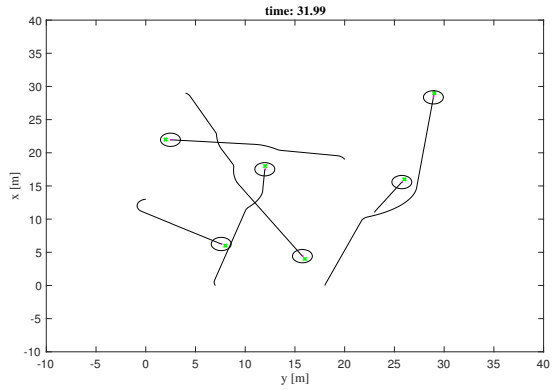


Figure D.1: Undesirable circular maneuvers, called reciprocal dances, as a result of the restrictions caused by a nonholonomic model with constant speed.



(a) Proposed algorithm



(b) Algorithm by Erlend Hårstad

Figure D.2: By comparing the agent paths, one can observe slight differences. As with RVO, the proposed algorithm tend to maintain a larger distance to other agents. The algorithm provided by Hårstad seem to have behavior that resembles RVO.

Appendix E

Creating the Simulator

This appendix presents the work of creating the simulator used in Chapter 5. The simulator is written in MATLAB R2017a. Martin Syre Wiig provided a basic version of the simulator he used in [23], including a basic kinematic nonholonomic vehicle model and live visualization. It is this basis that formed the foundation of the complete simulator used in Chapter 5.

The first task at hand was to implement the algorithm presented in [21], where the most important components are the integrated sensor representation and the collision avoidance function. The integrated sensor model is represented by the function $M(\alpha, t)$ presented in Chapter 2 and is implemented by calculating the distances between rays from the agent to obstacles inside the sensor disk. The closest ray that was not inside the safety distance was chosen. See Figure E.1. Next, the velocity compensation presented in [23] was implemented as part of the integrated sensor representation to account for obstacle velocity. For proof of concept and to improve visualization, the simulation obstacle model was implemented for both circular and rectangular obstacles, static and dynamic. Simulating rectangular obstacles require more computational effort and was hence excluded in the multi-agent environments, note that it is the simulation that is computational complex not the algorithm by itself.

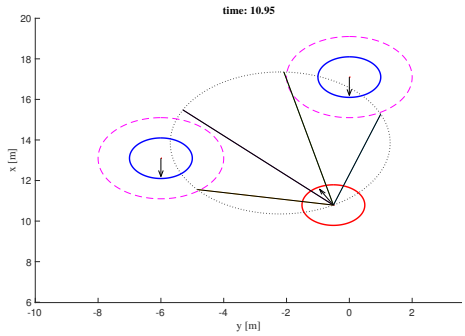


Figure E.1: Two obstacles with safety distance(blue and magenta) and the agent(red) with sensor disk and the rays(black). The angles from the agent heading $\psi_i(t)$ to the rays $\alpha_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ define blocked areas in $M(\alpha, t)$.

With the agent-obstacles simulator up and running, the next task was to implement simulations with multiple agents. The work required substantial revision of the main simulation loop and its functions. Beside implementing the algorithm, all the scenarios presented in Chapter 5 have been created from scratch. All plotting and visualization functions have been rewritten to fit the desired format. Furthermore, the Monte Carlo simulation functionality with data logging was implemented for statistical simulation experiments. Finally, the Reciprocal Velocity Obstacles from [24] was implemented as a comparison algorithm. Below are two code snippets; the main simulator loop of the proposed algorithm and function used to calculate the RVO cone from an agent to an obstacle.

```
%% initiation
h_instant = figure('units','normalized','outerposition',[0.2 0.2 0.6
    0.7]); % Figure for realtime plotting
hold on;

psi_d = zeros(1,time.get_sample_count);
num_agents_finished = 0;

```

```
%% Main control loop
for t=2:time.get_sample_count()-2

    for ii = 1:length(agent_list)
        current_agent = agent_list(ii);
        current_position = current_agent.state(1:2,t);

        % Skip if agent has crashed or reached it's goal
        % Break with a_max until speed is zero
        if current_agent.finished
            if current_agent.get_max_speed() ~= 0
                current_agent.calculate_next_state(t,current_agent.
    state(3,t), 0);
                if current_agent.get_speed() == 0
                    current_agent.set_max_surge_speed(0);
                end
            end
            continue;
        end


        %% Agents
        psi = current_agent.state(3,t);
        obst_inside_sensor_disk = [];
        cs = current_agent.get_sensor_disk_center(database.
    obstacleSenseDistance, t);
        for k = 1:length(agent_list)
            if k ~= ii
                % Check for collision with other agent
                dist_to_agent = agent_list(k).get_distance_to_point(t,
    current_position)...
                                - agent_list(ii).get_d_safe();

                % Collision between agents
                if dist_to_agent < 0
                    disp('---- COLLISION ----')
                    num_agents_finished = num_agents_finished + 1;
                    current_agent.set_max_surge_speed(0);
                    current_agent.finished = true;
```

```
44                  break;
45              end
46
47              % Check for agents inside the sensor disk
48              dist_to_cs = agent_list(k).get_distance_to_point(t, cs)
     ...
49                                  - agent_list(ii).get_d_safe();
50              if dist_to_cs < database.obstacleSenseDistance/2
51                  obst_inside_sensor_disk = [obst_inside_sensor_disk
    agent_list(k)];
52                  if k == closest_agent
53                      closest_agent = length(obst_inside_sensor_disk)
    ;
54                  end
55              end
56          end
57      end % k - inner agent loop
58
59      %% Obstacles
60      % Check for obstacles inside agents sensor range
61      for o = 1:length(obstacle_list)
62          dist_to_cs = obstacle_list(o).get_distance_to_point(t, cs)
    ...
63                                  - agent_list(ii).get_d_safe();
64          dist_to_obst = obstacle_list(o).get_distance_to_point(t,
    current_position) ...
65                                  - agent_list(ii).get_d_safe();
66
67          if dist_to_obst < 0
68              disp('--- COLLSION WITH OBSTACLE ---');
69              num_agents_finished = num_agents_finished + 1;
70              current_agent.set_max_surge_speed(0);
71              current_agent.finished = true;
72              steps_left = length(t+1:time.get_sample_count());
73              current_agent.state(:,t+1:time.get_sample_count()) =
    diag(current_agent.state(:,t))*ones(4,steps_left);
74              break;
75          end
76
```

```
77            if dist_to_cs < database.obstacleSenseDistance/2
78                obst_inside_sensor_disk = [obst_inside_sensor_disk
    obstacle_list(o)];
79                if o == closest_obst
80                    closest_obst = length(obst_inside_sensor_disk);
81                end
82            end
83        end % o - obstacles loop
84
85        %% Guidance
86        psi_guid = guidanceController(current_agent.get_target(),
    current_agent.state(:,t));
87        isCollisionMode = false;
88        if ~isempty(obst_inside_sensor_disk)
89            [M, M_pos, BR] = get_M_extended(current_agent,
    obst_inside_sensor_disk, database.obstacleSenseDistance, t);
90            isCollisionMode = true;
91            psi_d(t) = collision_avoidance(M, psi);
92        else
93             psi_d(t) = psi_guid;
94             BR = 0;
95        end
96
97        % Breaking rule
98        if BR == 1
99            BR_active(ii) = BR_active(ii) + 2;
100           if BR_active(ii) > BR_COUNTDOWN
101               BR_active(ii) = BR_COUNTDOWN;
102           end
103           new_speed = MIN_SPEED * current_agent.get_max_speed();
104       elseif BR_active(ii) > 0 && BR == 0
105           BR_active(ii) = BR_active(ii) - 1;
106           new_speed = MIN_SPEED * current_agent.get_max_speed();
107       end
108
109       % Stop if we are close to target
110       if norm(current_agent.get_target() - current_agent.state(1:2,t)
    )<=TARGET_RADIUS
111           num_agents_finished = num_agents_finished + 1;
```

```
            current_agent.finished = true;
        end

        % Update states
        current_agent.calculate_next_state(t,psi_d(t), new_speed);
    end % ii - agents loop

    % Update obstacle posistion
    for oo = 1:length(obstacle_list)
        obstacle_list(oo).calculate_next_state(t, obstacle_list(oo).
    state(3,t), obstacle_list(oo).get_speed());
    end

    % Plot an instant of the simulation
    if(mod(t,plot_time_step)==0)
        clf('reset');
        plotInstantAgents(agent_list, obstacle_list, t, h_instant, time
    , axis_array);
        drawnow;
    end

    %% Stop when all agents are finished
    if num_agents_finished == length(agent_list)
        disp('All agents finished')
        return;
    end
end % t - time loop
```

```
function vo = calc_cone(agent, obst, d_safe, d_rvo, t)
% Calculate RVO from agent to obstacle

    r = obst.get_radius() + d_safe + d_rvo; % Obst radius with safety
    dist

    a_pos = agent.state(1:2,t);     % Agent position
    o_pos = obst.state(1:2,t);      % Obstacle position

    a_v = agent.state_dot(1:2,t-1); % Agent velocity
    v_v = obst.state_dot(1:2,t-1);  % Obstacle velocity
```

```
11
12
13      % Calc tangent points from pos to obstacle + d_safe
14      P = a_pos' - o_pos';
15      d2 = dot(P,P);
16      Q0 = o_pos'+r^2/d2*P;
17      if d2-r^2 > 0
18          T = r/d2*sqrt(d2-r^2)*P*[0,1;-1,0];
19          p1 = Q0-T;  % Tangent points left
20          p2 = Q0+T;  % Tangent points right
21
22          % Move apex with respect to obstacle speed
23          % Where finished indicate a passive obstacle, equal to VO
24          if obst.finished
25              rvo_v = v_v;
26          else
27              rvo_v = (a_v+v_v)/2;
28          end
29          apex = a_pos + rvo_v;
30          p1 = p1' + rvo_v;
31          p2 = p2' + rvo_v;
32
33          % Calc each tangent line to obstacle through apex
34          ang1 = vector_ang(apex,p1);
35          ang2 = vector_ang(apex,p2);
36
37          vo = cone(apex, [ang1; ang2]); % Class to save RVO
38      else
39          % Corner case where calc above result in complex numbers
40          % RVO is simplified to a halv plane in the direction of the
        obst
41          ang = vector_ang(a_pos,o_pos);
42          if obst.finished
43              rvo_v = v_v;
44          else
45              rvo_v = (a_v+v_v)/2;
46          end
47          apex = a_pos + rvo_v;
48          vo = cone(apex, [ang+1.5; ang-1.5]); % Class to save RVO
```

```
49      end
50
51  end
```

References

[1] D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis. Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, 95:48 – 74, 2007.

[2] H. Sahin and L. Guvenc. Household robotics: autonomous devices for vacuuming and lawn mowing. *IEEE Control Systems*, 27:20–96, 2007.

[3] D. Voth. A new generation of military robots. *IEEE Intelligent Systems*, 19:2–3, 2004.

[4] T. Le-Anh and M. B. M De Koster. A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171:1–23, 2006.

[5] A. S. Matveev, M. Hoy, and A. V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33:463 – 497, 2015.

[6] Z. Shiller. Online suboptimal obstacle avoidance. *The International Journal of Robotics Research*, 9:480 – 497, 2000.

[7] I. Kamon and E. Rivlin. Sensor-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13:814 – 822, 1997.

[8] Y.H. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *The International Journal of Robotics Research*, 11:376 – 382, 1992.

[9] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. *Annual Symposium on Foundations of Computer Science*, 27:49 – 60, 1987.

[10] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[11] P. Ogren and N.E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21:188 – 195, 2005.

[12] K. Yang, S. Gan, and S. Sukkarieh. An efficient path planning and control algorithm for ruav's in unknown and cluttered environments. *Intelligent Robots and Systems*, 57:101 – 122, 2010.

[13] Y. Zhu and U. Ozguner. Constrained model predictive control for nonholonomic vehicle regulation. *Proc. IFAC 17th world congress*, 2008.

[14] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Proc. IEEE International Conference Robotics and Automation (ICRA)*, pages 500 – 505, 1985.

[15] Y. Koren and J. Borenstein. Potential field methods and their inherent limitation for mobile robot navigation. *IEEE int. Conference on Robotics and Automation*, pages 1398–1404, 1991.

[16] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Robotics and Automation*, 2:1398–1404, 1991.

[17] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7, 1991.

[18] K. Fujimura. Motion planning in dynamic environments. *Tokyo, Springer-Verlag*, 1991.

[19] A. Chakravarthy and D. Ghose. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28, 1998.

[20] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17:760 – 772, 1998.

[21] A. V. Savkin and C. Wang. Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation. *Robotics and Autonomous Systems*, 62:1568 – 1580, 2014.

[22] A. V. Savkin and C. Wang. A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles. *Robotica*, 31:993 – 1001, 2013.

[23] M. S. Wiig, K. Y. Pettersen, and A. V. Savkin. A reactive collision avoidance algorithm for nonholonomic vehicles. *Proc. 1st IEEE, Conference on Control Technology and Applications*, 2017.

[24] J. van den Berg, J. Manocha, D., and Ming Lin. Reciprocal velocity obstacles for real-time multi-agent navigation. *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928 – 1935, 2008.

[25] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R Siegwart. Reciprocal collision avoidance for multiple car-like robots. *IEEE International Conference on Robotics and Automation*, 2012.

[26] D. Barreiss and J. van den Berg. Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, 34:1501–1514, 2015.

[27] A. Yasuaki and M. Yoshiki. Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[28] E. Lalish, K. A. Morgansen, and T. Tsukamaki. Decentralized reactive collision avoidance for multiple unicycle-type vehicles. *IEEE American Control Conference*, 2008.

[29] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger. Safe maritime autonomous navigation with colregs, using velocity obstacles. *IEEE Oceanic Engineering*, 39:110–119, 2014.

[30] D. Panagou. A distributed feedback motion planning protocol for multiple unicycle agents of different classes. *IEEE Transactions on Automatic Control*, 62:1178 – 1193, 2017.

[31] S. B. Mehdi, V. Cichella, T. Marinho, and N. Hovakimyan. Collision avoidance in multi-vehicle cooperative missions using speed adjustment. *Proc. 56th IEEE Conference on Decision and Control*, 2017.

[32] M. Hoy, A. S. Matveev, and A. V. Savkina. Collision free cooperative navigation of multiple wheeled robots in unknown cluttered environments. *Robotics and Autonomous Systems*, 60:1253 – 1266, 2012.

[33] H. Teimoori and A. V. Savkin. Equiangular navigation and guidance of wheeled mobile robot based on range-only measurements. *Robotics and Autonomous Systems*, 58:203 – 215, 2010.

[34] A. Chakravarthy and D. Ghose. Obstacle avoidance in a dynamic environment: a collision cone approach. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(5):562–574, Sep 1998. ISSN 1083-4427. doi: 10.1109/3468.709600.

[35] M. S. Wiig, K. Y. Pettersen, and A. V. Savkin. A reactive collision avoidance algorithm for nonholonomic vehicles. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1776–1783, Aug 2017. doi: 10.1109/CCTA.2017.8062714.

[36] Convention on the international regulations for preventing collisions at sea. `https://www.navcen.uscg.gov/pdf/navRules/navrules.pdf`, 1972. Accessed: 2018-02-20.