Novel Reactive Motion Planning Algorithms for Safe Navigation of Automated Guided Vehicles

Zoltán Bálint Gyenes

Supervisor: Emese Gincsainé Dr. Szádeczky-Kardoss

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics

This dissertation is submitted for the degree of
Doctor of Philosophy (Ph.D.)

December 2023

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Zoltán Bálint Gyenes
December 2023

Abstract

Motion planning for mobile agents is a challenging task because the robot has to reach the target without collision with any of the obstacles that occur in the workspace.

The dissertation presents a comprehensive study on the development and evaluation of novel motion planning strategies for mobile robots operating in dynamic environments, a task fraught with numerous challenges. The primary focus is on ensuring safety, efficiency, adaptability, and rule-based velocity selection for an omni-directional mobile agent to facilitate effective navigation and avoid collisions within the workspace. The main goal is to reach the target position in a 2D workspace based on local, sensor-based information. A variety of novel methods, including the Safety Velocity Obstacle method (SVO), the Different Value Safety Velocity Obstacle (DVSVO) method, and the Genetic Algorithm Velocity Obstacle (GAVO) method, have been introduced and tested, each presenting unique merits in terms of speed, safety, adaptability, and collision avoidance.

Novel perception methods utilizing LiDAR measurement data have also been introduced to estimate the position and velocity vectors of the obstacles occurring in the workspace, using Kalman filter and Particle filter-based methods. The introduction of the Uncertainty Velocity Obstacle (UCVO) and Particle Filter Velocity Obstacle methods (PFVO), exploiting the principles of Velocity Obstacles (VO), which can be used to calculate the uncertainty degree, making collision-free path planning robust.

Additional contributions include exploring the use of traffic regulations (TRVO) in motion planning and advancing lane-keeping (LKVO) algorithms in a structured environment. For unavoidable collisions, the Collidable Velocity Obstacles (CVO) method is introduced, where the agent must select the velocity vector that causes the least damage at each sampling time. Furthermore, I introduced the Human Tracking Velocity Obstacles (HTVO) method that provides the possibility for collision-free motion planning of a human-tracking mobile robot.

The research presented in this dissertation demonstrates novel results in designing motion planning algorithms for mobile robots in dynamic environments, with promising practical implications for the future of autonomous robots.

Kivonat

Dinamikus környezetben történő mozgástervezés mobilis robotok számára kihívást jelentő feladatnak számít, mivel a robotnak úgy kell elérnie a célpozíciót, hogy ne ütközzön a munkatérben megjelenő akadályokkal.

A disszertáció egy átfogó tanulmányt mutat be a dinamikus környezetben mozgó mobilis robotok újszerű mozgástervezési stratégiáinak bevezetéséről és értékeléséről, amely feladat számos kihívást foglal magában. Az elsődleges hangsúly a biztonság, a hatékonyság, az adaptibilitás és a szabályalapú sebességválasztás biztosításán van omnidirekcionális mobilis robotok számára, hogy megvalósíthatóvá váljon a hatékony navigáció és ütközések elkerülése a munkatéren belül. A legfőbb cél a célpozíció elérése 2D munkatéren belül, lokális, szenzorinformációkat felhasználva. Számos újszerű módszer, Safety Velocity Obstacle (SVO) módszer, the Different Value Safety Velocity Obstacle (DVSVO) módszer, illetve a Genetic Algorithm Velocity Obstacle (GAVO) módszer került bevezetésre, amelyek mindegyike újszerű megoldást biztosít az ütközésmentes célelérési feladat megvalósításában.

LiDAR méréseken alapuló újszerű állapotbecslő módszereket vezettem be a munkatérben található akadályok pozíció és sebességvektorának becslésének céljából, felhasználva a Kalman filter és Particle filter algoritmusokat. Bevezetésre került az Uncertainty Velocity Obstacle (UCVO) és a Particle Filter Velocity Obstacle (PFVO) módszer, melyeknek a felhasználásával a bizonytalanság fokának számítására nyílik lehetőség, mindezzel robosztussá téve az ütközésmentes pályatervezést.

Továbbá, felhasználva az alapvető közlekedési szabályokat és sávtartási algoritmust, újszerű mozgástervezési módszer került bevezetésre strukturált környezetben (Traffic Regulation Velocity Obstacle (TRVO)- Lane Keeping Velocity Obstacle method (LKVO)). Az elkerülhetetlen ütközések esetére bevezetésre került a Collidable Velocity Obstacles (CVO) módszer, ahol az ágensnek a legkisebb kárt eredményező sebességvektort kell kiválasztania minden mintavételi időpillanatban. Továbbá az újonnan bevezetett Human Tracking Velocity Obstacles (HTVO) módszer lehetőséget biztosít az emberkövető mobilis robot ütközésmentes mozgástervezésére.

A bevezetett újszerű módszerek hozzájárulnak a mobilis ágensek ütközésmentes mozgástervezési algoritmusainak megvalósításához dinamikus környezetben és a jövőben számos felhasználási lehetőségre nyílik opció.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

$\mathbf{a}$	acceleration vector with dimension of 2, with unit of $[m/s^2]$
A	robot
B	obstacle
$\mathbf{Bez}(t)$	Bezier spline at LKVO method
$b_{i,n}(t)$	Bernstein polynomial at LKVO method
$C_G(\mathbf{v}_\mathrm{A})$	speed part of the cost function at SVO method
$C_h(\mathbf{v}_\mathrm{A})$	heading value at UCVO method
$C_{LK}(\mathbf{v}_\mathrm{A})$	cost value at LKVO method
$C_S(\mathbf{v}_\mathrm{A}, VO)$	safety part of the cost function at SVO method
CV_i	change of the velocity of the obstacle at UCVO algorithm
$dist_{OR_i}$	actual distance between the obstacle and the agent at UCVO algorithm
D_L	maximum range of the LiDAR sensor
D_{max}	given maximum distance at SVO method
$d_{min_{A,Bi}}$	minimum distance between the agent and the obstacle during their motion
d_p	saturated deviation of the particles at PFVO method
D_{ra_i}	distance between the robot and the obstacle at APF method
$D_{ra_{max}}$	largest distance that should be considered at APF method

$Dsafety$	distance between the human and the virtual target position at HTVO method
$D_S(\mathbf{v}_\mathrm{A})$	minimum distance between the agent and the VO cone
$\mathbf{F}$	force vector
f	fitness function
$\mathbf{F}_R$	sum of repelling and attractive forces vector at APF method
$\mathbf{F}_{rg}$	attractive force vector at APF method
$\mathbf{F}_{ro_i}$	repelling force vector at APF method
GO	speed component of the fitness function
goalV	virtual goal' position vector at HTVO algorithm
i_c	index of the obstacle that can be eliminated at CVO method
$inLane$	logical variable that shows whether the agent is inside the lane or not at LKVO method
K_{grid}	number of the velocity vectors in the grid
$\mathbf{K}_k$	Kalman gain matrix
m_A	mass of the robot at CVO method
m_B	mass of the obstacle at CVO method
$\mathbf{M}_k$	Error covariance matrix
MN_d	measurement noise
$\mathbf{p}$	position vector
$\mathbf{p}_\mathrm{A}$	position vector of the robot with dimension of 2, with unit of $[m]$
$\mathbf{p}_A(0)$	start position vector of the robot
$\mathbf{particles}_k^{(i)}$	particles matrix
$\mathbf{p}_\mathrm{B}$, $\mathbf{O}_\mathrm{i}$	position vector of the obstacle
P_{CV_i}	probability term depending on the change of the velocity of the obstacle at UCVO algorithm

P_{dist_i}	distance based probability at UCVO algorithm
$\mathbf{p}_{\text{goal}}$	position vector of the goal
P_i	probability at UCVO algorithm
$\mathbf{P}_i$	two-dimensional control point vector of a segment at LKVO method
P_{MV_i}	velocity based probability at UCVO algorithm
$\mathbf{p}_x$	position vector in the workspace where the obstacle and the robot would intersect their path at TRVO method
r	length of velocity vector
R_L	resolution of the LiDAR sensor
R_r	radius of the robot
$\mathbf{R}_v$	covariance matrix of the process noise matrix
$\mathbf{R}_z$	covariance matrix of the measurement noise vector
S_d	velocity set that contains every velocity vector resulting in a divergent maneuver
S_f	velocity set that contains every velocity vector resulting in a front maneuver
$\mathbf{S}_k$	weighted set of points in Particle filter algorithm vector
S_r	velocity set that contains every velocity vector resulting in a rear maneuver
t	time
t_H	given time horizon at LKVO method
$t_L(\mathbf{v}_A)$	time when the agent will reach the left boundary of the lane at LKVO method
T_{max}	maximum time interval
$t_{min_{A,Bi}}$	minimum time at the filtering method when the agent and the obstacle are at the closest point to each other
$T_{precheck}$	predefined constant at Precheck algorithm

$t_R(\mathbf{v}_A)$	time when the agent will reach the right boundary of the lane at LKVO method
T_s	sampling time
T_u	actual uncertainty time parameter at UCVO algorithm
$\mathbf{u}_k$	control input vector
$\mathbf{v}$	velocity vector with dimension of 2, with unit of $[m/s]$
$\mathbf{v}_1, \mathbf{v}_2, v_{1\mathrm{x}}, v_{2\mathrm{x}}$	velocity vectors and scalars at the recombination at GAVO method
$\mathbf{v}_\mathrm{A}$	velocity vector of the robot
$\mathbf{v}_\mathrm{B}$, $\mathbf{v}_{\mathrm{Bi,new}}$	actual velocity vector of the obstacle
$\mathbf{v}_{\mathrm{Bi,old}}$	previous velocity of the obstacle
$\Delta\mathbf{v}$	changes in the velocity vector
$\mathbf{v}_\mathrm{goal}$	velocity vector of the goal at HTVO algorithm
$\mathbf{v}_\mathrm{goalV}$	virtual goal velocity vector at HTVO algorithm
$\mathbf{v}_{grid}$	velocities on the grid
v_{max}	maximum length of the velocity
$\mathbf{v}_\mathrm{new}$	new velocity vector after the recombination at GAVO method
$\mathbf{v}_{VO}$	nearest point position vector on the VO cone
$w_k^{(i)}$	weight in Particle filter algorithm
$\hat{\mathbf{x}}_k$	Kalman filter prediction vector, Particle filter prediction vector
$\bar{\mathbf{x}}_k$	a priori state estimation vector
$\mathbf{y}_k$	output vector in Kalman filter equations
z_{p_x}	measured x center point of the obstacle
z_{p_y}	measured y center point of the obstacle

Greek Symbols

α	safety parameter in SVO and GAVO methods

β, β_d	speed, distance parameter in SVO, GAVO, UCVO methods
β_h	heading parameter at UCVO method
$\Delta\theta$	difference between the angle of the goal and the angle of the velocity vector
Δv	maximum change of the velocity
γ	APF parameter
γ_1, γ_2	random parameters at GAVO method
κ	random parameter
ξ_k	system noise at Particle filter algorithm
ρ	APF parameter
Σ_{k-1}	variance of the prior estimate
θ	angle of a vector with x axis
θ_{rg}	angle of the goal at GAVO algorithm
θ_{rv_i}	angle of the velocity vector at GAVO algorithm
$\theta_1, \theta_2, d\theta$	angle parameters at TRVO algorithm

Acronyms / Abbreviations

APF	Artificial Potential Field
CALU	Collision Avoidance with Localization Uncertainty
COCALU	Collision Avoidance under Bounded Localization Uncertainty
CVO	Collidable Velocity Obstacle method
DC	Directive circle
DVSVO	Different Value Safety Velocity Obstacle method
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
GA	Genetic algorithm

$GAVO$	Genetic Algorithm Velocity Obstacle method
$HTVO$	Human Tracking Velocity Obstacle method
ICS	Inevitable Collision States method
$IE-MO$	Inertial Estimation for the moving obstacle at the PFVO method
$IE-SO$	Inertial Estimation for stationary obstacle at the PFVO method
KF	Kalman filter
$LKVO$	Lane Keeping Velocity Obstacle method
$NSGA$	Non-dominated Sorting Genetic Algorithm
$ORCA$	Optimal Reciprocal Collision Avoidance
PF	Particle filter
PFM	Probabilistic Foam Method
$PFVO$	Particle Filter Velocity Obstacle method
$PRVO$	Probabilistic Velocity Obstacles
RAV	Reachable Avoidance Velocities
RRT	Rapidly-Exploring Random Tree
RV	Reachable Velocities
RVO	Reciprocal Velocity Obstacles
SA	safety component of the fitness function
$SLAM$	Simultaneous Localization and Mapping
SVO	Safety Velocity Obstacle method
$TRVO$	Traffic Regulation Velocity Obstacle method
$UAVs$	Unmanned Aerial Vehicles
$UCVO$	Uncertainty Velocity Obstacle method
UKF	Unscented Kalman Filter
VO	velocity obstacle cone

Chapter 1

Introduction

1.1 Field of Robotics

Robots have become an integral part of our lives, transforming the way we live, work, and interact with the world around us. From industrial manufacturing [1, 2] and healthcare [3–5] to education [6, 7] and entertainment [8, 9], robots are increasingly being deployed in various sectors, enhancing efficiency, productivity, and even creativity.

There are several types of robots, each designed to perform specific tasks or operate in certain environments. Industrial robots, for instance, are widely used in manufacturing processes, performing tasks such as assembly [10], painting [11], welding [12], and packaging [13] with high precision and speed. These robots often operate in highly structured environments and follow pre-programmed instructions.

Humanoid robots are designed to mimic human form and behavior, enabling them to interact with human environments and tools in ways that other robots cannot [14, 15]. They are used in a variety of applications, including research, entertainment, and assistive technologies.

Medical robots are another important category, with applications ranging from surgical assistance [16] and rehabilitation therapy [17] to patient care [18] and drug delivery [19]. These robots can perform tasks with a level of precision and consistency that surpasses human capabilities, improving patient outcomes and transforming healthcare delivery.

Mobile robots, on the other hand, are designed to move around in their environment. This category includes autonomous vehicles [20], drones [21], and service robots [22]. These robots are often equipped with sensors and sophisticated algorithms that allow them to navigate complex, dynamic environments. They are used in a wide range of applications, from delivering goods [23] and inspecting infrastructure [24] to assisting in search [25], rescue missions [26] and exploring remote or hazardous locations [27].

Mobile robots have become increasingly prevalent in various sectors, including warehouses [28], factories [29], and public spaces [30]. The ability of these robots to autonomously navigate through dynamic environments is crucial for their effective operation. However, this poses a significant challenge, as it requires the robots to continuously plan and execute motions that are both efficient and safe. The motion planning aspect involves generating both the path and velocity profiles for the robot, taking into account the positions and velocity vectors of obstacles in the environment.

One of the primary challenges in motion planning for mobile robots is ensuring collision avoidance, especially in dense and dynamic environments where obstacles, including humans, may have unpredictable movements. Collision avoidance is not just about finding a path that avoids obstacles but also about dynamically changing the path as the environment changes. This requires the robot to have a perception of its environment and be able to estimate the state of both itself and the obstacles around it.

State estimation plays a pivotal role in this context, as it involves processing sensor data to discern the mobile robot's state, such as its position and velocity, at discrete time steps [31–33]. Moreover, it is essential for the robot to estimate the state of the obstacles in its workspace. However, sensor data is often noisy and incomplete, leading to uncertainties in state estimation [34–36]. Addressing these uncertainties is critical for ensuring the safety and reliability of the robot's movements.

Reactive motion planning algorithms are often employed to address these challenges. These algorithms make use of sensor data to make decisions for the robot's movements [37–39]. However, traditional reactive motion planning algorithms primarily focus on finding the fastest path to the goal [40–42], which may not always be the safe enough, especially in environments with inherent uncertainties in sensor measurements.

The main goal of this dissertation is to introduce novel motion-planning algorithms for Autonomous Guided Vehicles that can generate collision-free motion from the start to the goal position in a dynamic environment while using local, sensor-based information. In the last two decades, a lot of researchers focused on motion planning algorithms from mobile agents. In the next Section, the related work is presented.

1.2 Related work

The main goal of the motion planning algorithms is to generate a safe, collision-free motion for the mobile agent. Based on the level of knowledge about the environment, motion planning for robots can be classified into global and local algorithms. At the

local motion planning algorithms, the environment is unknown, only local, sensor-based information is available.

Offline *global motion planning* methods assume that complete information about the environment exists a priori. Such information is usually only available for static environments. Examples of global motion planning methods include hybrid A* [43], Rapidly-Exploring Random Tree (RRT) [44–46].

1.2.1 Local motion planning methods

In contrast, *local motion planning* techniques consider only information about the immediate neighborhood of the robot, information that is usually accessible from the robot's own sensors. A typical example is the artificial potential field (APF) method, where obstacles create virtual potential fields acting as repulsive virtual forces while the destination generates an attractive virtual force. The vector sum of these forces allows the planner to calculate the velocity vector of the robot at every sampling interval [47–49]. While initially proposed for static environments, the potential field model was also extended for dynamic obstacles, such as robot soccer [50]. The APF method often finds only the local optimum solution. The method will be presented in detail in Section 2.2.

The Dynamic Window Approach (DWA) is a widely used motion planning method for mobile agents. This motion planning method can be used to generate appropriate solutions. Non-holonomic constraints, such as limited turning ability can be considered and the method can be used in dynamic environments to generate the collision-free path [51, 52]. The DWA algorithm was used recently for the solution of the motion planning problem of Forklift Automated Guided Vehicle [53].

Another well-known local motion planning algorithm is the Velocity Obstacles (VO) algorithm [54]. The VO calculated by the algorithm is the set of velocity vectors that would cause a collision between the agent and the obstacles in a future step. At every sampling step, the robot selects a velocity vector that would result in a collision-free motion, following a specific strategy. The main concept of the VO method will be presented in Section 2.1. Various strategies are possible, including strategies that temporarily move away from the goal in order to avoid collisions. The VO technique had been extended to a range of scenarios including differential-driven robots [55, 56], multi-agent navigation [57], ships [58], UAVs (Unmanned Aerial Vehicles) [59] and multi-robot collision avoidance [60–62]. Although the VO method is basically developed for mobile robots, it can also be applied to robotic manipulators [63]. This reactive motion planning method is presented in detail in Section 2.1. The main limitation of the VO method is that it generates the fastest

solution to the goal position that could results in a collision due to sensor measurement uncertainty. It is important to note that there is also an approach called reachable set-based trajectory planning [64], which can be used for motion planning of autonomous vehicles considering those sets in 2D environment that cannot be reached. It can be used in small and complex solution spaces considering the safety as an important aspect.

The εCCA is an extended version of the Reciprocal Velocity Obstacles algorithm [57], using kinodynamic constraints of the agent. The algorithm provides a suitable solution for the multi-robot collision avoidance problem in a dense environment. The computational time plays an essential role in this method. The whole environment of the robot is divided into a grid-based map. The agent selects a collision-free velocity vector using both convex and nonconvex optimization methods [65].

Differential constraints method was presented in [66] where a special search space was constructed to satisfy all of the constraints. The basic of the algorithm is a deterministic search method using a discretized state space. All of the reachable neighbors will be calculated that are available resulting feasible motion. This method uses connected search graph. Dynamic programming can be applied at this method. One of the biggest advantage of this method is that it executes a fast solution in highly changing dense environment.

The Inevitable Collision States method (ICS) generates every states of the agent where there is no control command that would cause a collision-free motion. The main concept is to guarantee that the robot never finds itself in an ICS situation. The algorithm is appropriate both in static and dynamic environments [67, 68].

The Model Predictive Control (MPC) is often used in motion planning algorithms for mobile robots because it can be used in a dynamic, often unknown environment using sensor measurement data [69–72]. At the heart of MPC is the creation of a predictive model that includes both the robot's dynamics and its interaction with the surrounding environment. This model is not static but continually evolves, taking into account a sequence of future states over a given time horizon. The real power of MPC lies in its ability to solve an optimization problem at each time step, recalibrating the robot's trajectory in real-time. This constant recalibration is based on immediate sensor feedback, allowing the robot to avoid obstacles and address other environmental changes. The MPC-based motion planning algorithm for mobile agents was also used in military aspects [73]. The MPC has also limitation comparing with the other reactive motion planning methods: it has to solve the optimization problem in every control step, so the computation time can be increased.

1.2.2 Artificial intelligence and probability-based motion planning methods

One of the first methods that proposed the use of Genetic Algorithms (GAs) in path planning was [74], where the objective was to reach a goal in an environment with no obstacles. [75] used GAs in a global path planning setting to find an optimal path in a large-scale grid environment. [76] combined GAs with a probabilistic roadmap model for path planning in a static environment. Multi-objective genetic algorithms, such as NSGA (Non-dominated Sorting Genetic Algorithm) can also be used for these tasks [77]. [78] used the GA for multiple agent tasks. GA was combined with recurrent neural networks in [79]. Most of these research projects assumed a static environment. While projects like [80, 81] demonstrated that the GA model can be adapted to dynamic environments with moving obstacles, it was not possible to run the algorithms in real time. At the moment, artificial intelligence-based motion planning algorithms cannot reach better results than the reactive methods but in the future, considering the development of the technology, it can be changed.

Several other types of Artificial Intelligence methods were used for motion planning algorithms considering mobile agents. The Convolutional Neural Network (CNN) was used for the path planning of the robot using a camera as a sensor on the mobile agent [82–84]. Learning from demonstration is also an opportunity to solve the motion planning task of a mobile agent [85]. The Reinforcement Learning-based motion planning algorithms can be also used for collision-free navigation of mobile agents [86–88]. A comprehensive survey was written comparing the Artificial Intelligence-based motion planning algorithms that can be used for the collision-free navigation of a mobile agent [89]. It must be noted that because of the computational demand, usually, it is hard to use the AI algorithms in real time.

The Probabilistic Velocity Obstacles (PRVO) algorithm is also an extended version of the Reciprocal Velocity Obstacles (RVO) method [57], using probabilistics. The time scaling algorithm and Bayesian decomposition are used. This method provided better performance in traversal times than the existing bound-based methods. The algorithm was tested using simulation results [90].

The probability-based motion planning for mobile robots is a highly researched topic for a long time. The PRVO method incorporates probabilistic principles alongside time scaling mechanisms and Bayesian decomposition. Demonstrably, this technique outperforms extant bound-based methods in terms of traversal times. The efficiency of the algorithm has been validated through a series of simulation results [90]. In [91], a pyramid structure was used that provided conditional probabilities for every pixel considering the uncertainty of the environment. The Probabilistic Foam Method

(PFM) [92] was introduced to guarantee a safe path for the agent. In this method, the environment is structured with bubbles and uses a breadth-first search to generate the optimal solution. The probabilistic representation of the uncertainty was used in [93] that planned the future probabilistic distribution of the state of the agent and also the collision's probability. The solution is provided by solving the linear program considering linear chance constraints. A probabilistic method [94] was also introduced to calculate the probability of human motion for the agent's motion planning method. To solve this task, a probability grid was introduced from the observation.

1.2.3 State estimation methods

As it was presented in Section 1.1, the state estimation methods play a huge role in the motion planning algorithms for mobile agents. In mobile robotics, the Kalman filter has been a main algorithm for addressing the self-localization problem, which entails estimating the position and velocity of the mobile agent [95–99]. Initially designed for linear systems, the Kalman filter's scope was later complemented by the introduction of the Particle filter, a Bayesian filtering method, which is adept at handling nonlinear systems with non-Gaussian distributions [100–102]. While these algorithms and their derivatives have proven effective in localization [103, 104], their primary focus has not been on providing information about obstacles in the workspace.

The Extended Kalman Filter (EKF) broadens the applicability of state estimation to include not only linear systems but also nonlinear ones [105, 106]. By linearizing the nonlinear model at each time step and implementing the Kalman filter on the linearized model, the EKF permits system state estimation. It has found notable utility in mobile robotics, particularly in addressing the agent localization problem [96, 107, 108], and is widely acknowledged within the robotics community, having been applied across various research studies and applications over the past decade [109, 110].

The Unscented Kalman Filter (UKF) offers an alternative to the EKF for mobile agent localization in nonlinear systems. Compared to the EKF, the UKF can provide a more accurate solution to the localization problem by sliminating the linearization step [111–113].

The Particle Filter algorithm, introduced initially in 1955 [114], estimates system state by simulating numerous particles or "molecules." The algorithm was later re-branded as the Bootstrap filter in 1993 [100], upon its implementation as a recursive Bayesian filter. The fundamental premise of this algorithm is to construct a posterior distribution using an ensemble of differently weighted samples, which are calculated and updated at each sampling interval based on measurement data. For nonlinear

systems, this form of Bayesian filter has been shown to outperform the EKF and UKF, even when the number of particles is constrained [115].

The Collision Avoidance with Localization Uncertainty (CALU) algorithm, designed for multi-robot collision avoidance, leverages the Particle filter to tackle the localization challenge intrinsic to mobile robotics. This innovative method integrates the Optimal Reciprocal Collision Avoidance (ORCA) approach [116] to forge collision-free motion plans. The central aim of this method is to constrain the inherent error within the localization process.

The Simultaneous Localization and Mapping (SLAM) methodology comprises dual interlinked stages: revision of an environmental map, conducted in tandem with tracking the agent's or robot's location within this same environment [117–120]. The SLAM methodology has been applied in conjunction with both the Kalman filter [121] and the Particle filter [121, 122]. It is important to note that the original SLAM algorithm did not possess the capability to segment obstacles from the overall environment, and was restricted to defining the position information within the map.

The algorithm termed 'Collision Avoidance under Bounded Localization Uncertainty' (COCALU) [103] proposes an innovative mechanism of convex hull peeling, with an aim to curtail the error associated with localization. This approach yields superior performance when juxtaposed with the preceding 'Multi-robot CALU algorithm [123].

The main goal of these localization methods was to focus on the estimation of the agent which has an onboard sensor, not on the obstacles that occur in the workspace of the agent. The state perception method, that I introduced to perceive the obstacles, is based on Kalman filter and Particle filter. The main concept of the Kalman filter is presented in Section 4.2.1 and the basics of the Particle filter-based estimation method is presented in Section 4.2.2.

1.3 Research questions and methods

In examining a wide range of state estimation methodologies, one finds that much prior work in the field of Robotics has been dedicated to tackling the self-localization problem. At the same time, reactive motion planning techniques are predominantly aimed at generating the quickest path to a target location.

The fastest target-reaching methods result in a tangential motion taking into account all of the obstacles in the workspace. Although these results generate the fastest solution, in some cases, they are not safe, if the movement of the obstacles is not known. Such observations raised the first research question: Could a novel motion

planning method be introduced that weighs both safety and speed at the same time, utilizing a cost-function-based algorithm? This cost-function-based algorithm can generate a collision-free motion for the agent reaching the target position. A subsequent query surfaced: Could the results be further augmented via an Artificial Intelligence method, like a Genetic Algorithm-based motion planning algorithm? It is important to add, that in principle, the notion of safety can be evaluated in binary terms, since if the agent collides with one of the obstacles in its environment, the movement is not safe, and if it does not collide, the result of the movement planning is safe. However, the degree of safety is also correlated with the value of the distance from the obstacles. Since the position and velocity of the obstacle are known only with some uncertainty, the probability of no collision is the higher the further the obstacle is from the agent. If the agent performs a tangential motion with respect to one of the obstacles, then if the sensor measurements are imperfect, the planned tangential motion may result in a collision. Of course, if the agent performs its movement at an unnecessarily large distance from the obstacle, then the length of the path will be far from optimal, so it is worth using a cost function to determine the velocity vector of the robot.

Moreover, it is of note that many earlier state estimation methods were primarily focused on the self-localization problem of the agent, paying little attention to the obstacles occurring in the workspace. However, through the use of techniques such as the Kalman filter and Particle filter, I found it possible to discern the position and velocity vectors of these obstacles in the workspace. Earlier techniques frequently treated the uncertainty parameter as a constant. This raised another question: Could the uncertainty parameter be dynamically calculated, by estimating the uncertainty from sensor measurements of the obstacle? Additionally, is it an opportunity to connect the state perception method with the motion planning algorithm?

After that, other questions arose, whether the Traffic Regulation rules and lane-keeping algorithms could be taken into account in the motion planning of mobile robots in a structured environment. The structured environment means that there are also lanes and traffic rules in the workspace of the mobile agent.

What should the velocity vector of an agent be if a collision is unavoidable? The research also considered this question. How the motion planning algorithm should be changed if there are not only robots but also people in the environment? Could a technique be devised to allow a robot to follow a human at a fixed distance, a prospect with numerous potential future applications?

I introduced novel motion planning algorithms that can generate collision-free motion for mobile agents, considering the question that arose. I used the basics of the Velocity Obstacle method and introduced novel motion planning algorithms. It can be assumed that the velocity vector of the obstacles is constant until one sampling time. I

implemented and tested all novel methodologies primarily in a simulation environment, using platforms such as MATLAB, CoppeliaSim, and Gazebo. The efficiency of these novel algorithms was compared with existing motion planning methods for mobile robots. I also engineered with colleagues an omnidirectional mobile robot outfitted with an Rp2 LiDAR sensor [124] to gather data about the surrounding environment and using the information about the workspace, to estimate the state of the obstacles.

The remaining parts of the dissertation have the following structure: Section 2 presents the Background algorithms. After that, Section 3 shows the steps of the introduced novel motion planning algorithms where speed and safety can be considered at the same time. Later on, Section 4 presents the Kalman filter and Particle filter-based state perception methods that were introduced to perceive the state of the obstacles in the workspace. This Section introduces novel motion planning methods that can consider the uncertainty of the states of the obstacles. After that, Section 5 introduces novel motion planning methods that can be used in a structured environment or in human presence. Finally, Section 6 will sum up the results of the research and show the opportunities for future work.

Chapter 2

Background

In this Chapter two main algorithms are presented, the Velocity Obstacles (VO) method and the Artificial Potential Field (APF) method. The introduced algorithms are based on VO method and the comparison with the APF method was used to evaluate the introduced algorithms.

2.1 Velocity Obstacle method

The main objective of the Velocity Obstacles (VO) algorithm [54] is to select a velocity vector that generates no collision for the agent if the position and velocity information of the obstacles are known or measurable at the time of decision-making. The basic version of the method selects the velocity vector for an omnidirectional mobile robot moving in 2D workspace.

As an assumption, both the agent and the obstacles are considered as disk-shaped objects (if not, then a circle that can be written around can be used). In the beginning, the radius of the agent is reduced to zero, by increasing every obstacle's radius with the radius of the agent. In this case, the agent is a point-like object. The velocity obstacle cone VO_i is the set of all velocity vectors of the agent A that would result in a collision with the obstacle B_i at a later time:

$$VO_i = \{ \mathbf{v}_\mathrm{A} \mid \exists\, t : A(\mathbf{p}_\mathrm{A} + \mathbf{v}_A t) \cap B_i(\mathbf{p}_\mathrm{Bi} + \mathbf{v}_\mathrm{Bi} t) \neq 0 \} \tag{2.1}$$

In this equation, $A(\mathbf{p})$ and $B_i(\mathbf{p})$ are the set of points in the workspace which are occupied by the robot (respectively obstacle i) if its position is $\mathbf{p}$. (2.1) says that applying a robot velocity vector $\mathbf{v}_\mathrm{A} \in VO_i$ will result in a collision between A and B_i at time t. (As an assumption, the velocities of the obstacles and the robot are unchanged

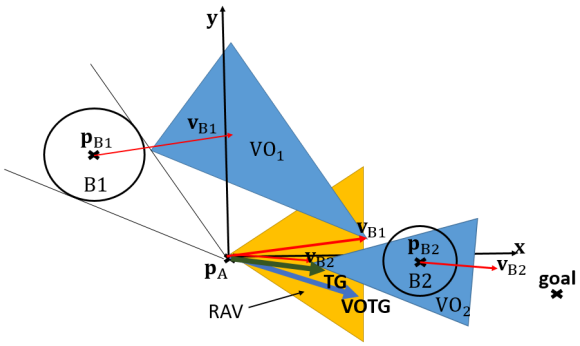

Fig. 2.1 TG and VOTG strategies

until t.) On the other hand, the $\mathbf{v}_A \notin VO_i$ velocity selection ensures that A and B_i will not collide until they do not change their velocities $\mathbf{v}_A$ and respectively $\mathbf{v}_{Bi}$.

Considering all the obstacles, the complete velocity obstacle VO is the union of the VO_i sets:

$$VO = \bigcup_{i=1}^{m} VO_i \tag{2.2}$$

where m denotes the number of obstacles. The shape of the VO_i is a triangle because it would be an open cone but as a limitation, it is reduced to a triangle. It is presented in Figure 2.2.

The reachable velocities set RV is the set of all feasible velocity vectors $\mathbf{v}_A$ that can be reached by the robot by the sampling time considering its motion capabilities (e.g. bounded inputs). The key step in the VO method is the calculation of the reachable avoidance velocities (RAV) set, which can be calculated by subtracting from RV the VO set, i.e. determining the robot velocities that are feasible and will not cause any collision. These are the velocities from which the robot can choose to get a collision-free path. Using the TG (to goal, TG) strategy, the maximum velocity will be selected in the line to the goal position. In the VOTG strategy, the maximum velocity is selected at the closest angle to the goal position. The TG and VOTG strategies can be seen in Figure 2.1. A frequent approach is to discretize the RAV set by overlaying a grid on the RAV area. Using these strategies, results in the fastest solution which could generate a collision in every situation where the noise of the sensor measurement occurred.

Figure 2.2 shows the application of the VO method for the case of a robot navigating an environment with a static and a moving obstacle. In the case of the static disk-shaped obstacle B_2, the VO_2 is a triangle in the plane, such that its vertex is at $\mathbf{p}_A$ and its sides are tangent to the obstacle's circle. For the moving obstacle B_1, the cone has to be translated by the velocity vector $\mathbf{v}_{B1}$ of the obstacle to get the corresponding VO_1 set.

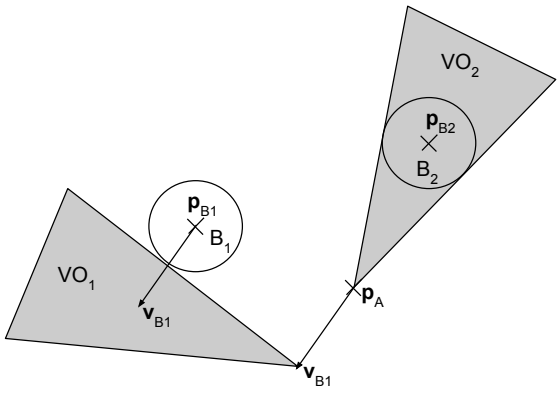

Fig. 2.2 An example of the VO method for a robot at position $\mathbf{p}_{\mathrm{A}}$ navigating an environment consisting of a moving obstacle B_1 at position $\mathbf{p}_{\mathrm{B1}}$ with velocity $\mathbf{v}_{\mathrm{B1}}$ and a static obstacle B_2 at $\mathbf{p}_{\mathrm{B2}}$. The gray area illustrates $VO = VO_1 \cup VO_2$.

2.2 Artifitial Potential Field method

The APF method is an often-used reactive motion-planning algorithm. The main concept is to calculate the summation of the attractive (between the robot and the target) and repelling (between the agent and the obstacles) forces [47, 125, 48]. One weakness of this algorithm is that sometimes only the local optimum can be found. The steps of the APF method are as follows: During motion planning, in every sampling time step, $\mathbf{F}_R$ force will influence the motion of the agent. The $\mathbf{F}_R$ force depends on the repelling ($\mathbf{F}_{ro}$) and the attractive ($\mathbf{F}_{rg}$) forces. The closer the robot is to the obstacle, the larger the volume of the repelling force is. The repelling force can be calculated by

$$\mathbf{F}_{roi} = \frac{\gamma\sqrt{\frac{1}{D_{ro_i}} + \frac{1}{D_{romax}}}}{{D_{ro_i}}^2}\mathbf{F}_{ROi} \tag{2.3}$$

where D_{ro_i} denotes the distance between the robot A and obstacle B_i, $\mathbf{F}_{ROi}$ is the vector between the obstacle and the agent, γ is a specific parameter that identifies the role of the repelling force in the motion-planning algorithm and $D_{ro_{max}}$ is the largest distance that should be considered in the motion-planning algorithm.

The attractive force can be calculated as

$$\mathbf{F}_{rg} = \rho\mathbf{F}_{RG} \tag{2.4}$$

where $\mathbf{F}_{RG}$ is the vector between the robot and the target and ρ is the parameter of the attractive force (depending on the usage of the algorithm). The force that influences the

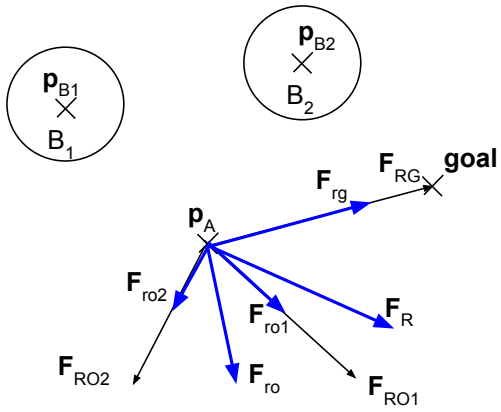

Fig. 2.3 APF method

motion of the robot can be calculated with the sum of the repelling and the attractive forces:

$$\mathbf{F}_R = \mathbf{F}_{ro} + \mathbf{F}_{rg} \tag{2.5}$$

where $\mathbf{F}_{ro} = \sum_{i=1}^{m} \mathbf{F}_{roi}$.

Figure 2.3 illustrates the different forces presented. There are two obstacles in the workspace (B1) and (B2) with the position $\mathbf{p}_{B1}$ and $\mathbf{p}_{B2}$ as it was seen in Figure 2.2. The agent is at the $\mathbf{p}_A$ position at this point, and the summation of the forces can be checked. If the mass of the agent is known, then the acceleration can be calculated using Newton's second law:

$$\mathbf{a} = \frac{\mathbf{F}_R}{m_a} \tag{2.6}$$

where m_a is the mass of the robot. The changes in the velocity vector can be calculated if the force and the sampling time are known:

$$\Delta\mathbf{v} = \mathbf{a}T_s \tag{2.7}$$

So the actual velocity can be calculated using the previous velocity and the change in velocity:

$$\mathbf{v}_{\text{new}} = \mathbf{v}_{\text{prev}} + \Delta\mathbf{v} \tag{2.8}$$

Additionally, the APF method has also been developed for unmanned arial vehicles, while human–robot interaction was also simulated using the APF motion-planning method by using the motion characteristics of household animals [126]. It is important to mention that the APF method can sometimes find only the local minimum. There are instances when it may encounter a deadlock situation, rendering it unable to reach the desired goal position. In numerous robotics disciplines, the APF method is frequently adopted. Consequently, I also employ it for comparison of the introduced motion planning methods.

Chapter 3

Novel Safety Motion Planning Algorithms for Mobile Robots

As was seen in Section 1.2, most motion-planning algorithms try to reach the goal position as fast as it is possible for an omnidirectional mobile agent. My first goal was to introduce a novel reactive motion planning method that can consider speed and safety at the same time at the collision-free target reaching, using a cost-function-based solution. The main task of the reactive motion planning algorithm is to select a velocity vector from the RAV set.

3.1 Safety Velocity Obstacle (SVO) method

In the Safety Velocity Obstacle method, the main goal was to maintain the target reaching by providing safety at the same time. If there is a dense environment with moving obstacles, the algorithm can be faster if it does not consider every obstacle in every sampling time. For the introduced algorithm, a pre-condition is that both the agent and the obstacles are represented as disk-shaped objects with known radii. The agent collects information about the obstacles in the workspace at each sampling time instant from the sensor placed on it, determining the position and velocity vector of the obstacle. It is assumed that the agent and the obstacles in its workspace do not change their velocity during a sampling period. For the VO method, which is the basis of the methods, the VO sets can be determined such that no velocity change occurs until the collision. The introduced method is similar to the global motion planning methods in that it assumes the robot's immediate environment to be known for a small period of time. Additionally, the obstacle motion is measured by the onboard sensor. Because of the assumption of constant velocity, it is important to keep a bigger distance between

the agent and the obstacles to avoid collisions if the constant velocity assumption is not fulfilled.

3.1.1 Precheck algorithm

An obstacle selection method can be used called Precheck method. In this case, not every obstacle must be considered at the sampling time. Only those VO_i of the obstacles must be taken into account during the calculation process of the *RAV* set that fulfill the precheck algorithm. Two cases of the obstacles will not be considered:

- those obstacles that are far from the agent and

For all obstacles, the minimum time and distance must be calculated when the agent and the obstacle are closest to each other during their motion.

For this calculation, first, the relative position and velocity vector between the agent and the obstacle must be determined.

1. **Relative Position and Velocity**:

$$\mathbf{p}_{\text{rel}} = \mathbf{p}_{\text{A}} - \mathbf{p}_{\text{Bi}} \tag{3.1}$$

$$\mathbf{v}_{\text{rel}} = \mathbf{v}_{\text{A}} - \mathbf{v}_{\text{Bi}} \tag{3.2}$$

2. **Determine Distance Over Time**:

$$D^2(t) = \mathbf{p}_{\text{rel}} \cdot \mathbf{p}_{\text{rel}} + 2(\mathbf{p}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}})t + \mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}t^2 \tag{3.3}$$

3. **Differentiate with respect to** t:

$$\frac{d}{dt}D^2(t) = 2(\mathbf{p}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}) + 2\mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}t \tag{3.4}$$

4. **Set the derivative to 0 to find the minimum value of** $t_{min_{A,Bi}}$. In the precheck algorithm, $t_{min_{A,Bi}}$ means the time when the agent and the obstacle will be the closest to each other during their motion:

$$2(\mathbf{p}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}) + 2\mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}t = 0 \tag{3.5}$$

From the above:

$$\mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}t = -\mathbf{p}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}} \tag{3.6}$$

Thus:

$$t_{min_{A,Bi}} = \frac{-\mathbf{p}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}}{\mathbf{v}_{\text{rel}} \cdot \mathbf{v}_{\text{rel}}} \tag{3.7}$$

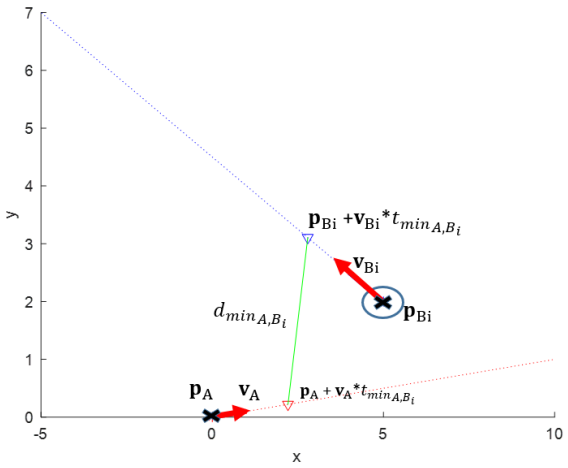

Fig. 3.1 Precheck algorithm. The goal is to calculate the minimum distance between the agent and the obstacle in a future time.

Substituting back the position and the velocity vectors, the time, when the agent and the obstacle will be closest to each other:

$$t_{min_{A,Bi}} = \frac{-(\mathbf{p}_A - \mathbf{p}_{Bi})(\mathbf{v}_A - \mathbf{v}_{Bi})}{||\mathbf{v}_A - \mathbf{v}_{Bi}||}, \tag{3.8}$$

where $t_{min_{A,Bi}}$ presents the time interval when the robot and the obstacle will be nearest to each other. If the value of this parameter is a negative number then it was in the past. $||.||$ represents the secondary norm.

The minimal distance can be calculated:

$$d_{min_{A,Bi}} = ||(\mathbf{p}_A + \mathbf{v}_A t_{min_{A,Bi}}) - (\mathbf{p}_{Bi} + \mathbf{v}_{Bi} t_{min_{A,Bi}})||, \tag{3.9}$$

So only those obstacles must be considered that fulfill the next equation:

$$\begin{aligned} 0 < t_{min_{A,Bi}} < T_{precheck} \quad \text{AND} \\ d_{min_{A,Bi}} < \mathrm{v}_{\max} * T_{precheck} \end{aligned} \tag{3.10}$$

where $\mathrm{v}_{\max}$ means the maximum velocity that the robot can reach and $T_{precheck}$ is a parameter of the algorithm that must be tuned. The experiments showed that if the value of the $T_{precheck}$ parameter is too small, it generates not a smooth path for the agent.

The precheck algorithm is illustrated in Figure 3.1.

Because of the precheck algorithm, the RAV set can be increased hence not every VO_i should be considered only those that fulfill (3.10).

3.1.2 Cost function-based velocity selection

The base of the robot velocity vector is at the position of the robot, the choice of its endpoint is the optimization task at each sampling time instant, i.e., determining the coordinates v_x, v_y for the agent. Initially, the robot is located at the origin, its velocity vector is 0. An important constraint is that the velocity vector must be reachable and avoidable, i.e. it must be located within the RAV set, the maximum velocity is v_{max} and the minimum velocity is 0 m/s. To take into account the computational complexity of heuristic motion planning, the cost function is evaluated in discrete points, but of course, for this optimization step, other algorithms can be used, such as optimization combined with the genetic algorithm.

The optimization step, using a cost function can be described using the following equation:

$$\mathbf{v}_\mathrm{A} = \arg \min_{\mathbf{v}_\mathrm{A} \in RAV} C(E, \mathbf{p}_A(0), \mathbf{p}_{goal}) \tag{3.11}$$

where $\mathbf{p}_{goal}$ is the position of the target, $\mathbf{p}_A(0)$ represents initial position of the agent, E is the environment and C means the cost function. The main goal of the mobile agent is to select a velocity vector from the RAV set that can minimize the cost function resulting in a collision-free solution from the start to the goal position considering the dynamic environment.

Using a cost function that must be minimized, the velocity vector from RAV set can be determined by considering different aspects (speed, safety). The different parameters are weighted in the cost function, so they can play different role during the motion planning method:

$$Cost(\mathbf{v}_\mathrm{A}) = \alpha\, C_S(\mathbf{v}_\mathrm{A}, VO) + \beta\, C_G(\mathbf{v}_\mathrm{A}) \tag{3.12}$$

where different parameters can be defined as α as the safety parameter and β as the speed. $C_S(\mathbf{v}_\mathrm{A}, VO)$ means the normalized distance between the velocity vector and the nearest VO cone and $C_G(\mathbf{v}_\mathrm{A})$ denotes the distance between the target position and the position that the robot can reach by using the selected velocity vector, and it is normalized by the initial distance between the position of the agent and the goal position. Additionally, values of α and β parameters are bigger or equal to 0 (they can be bigger than 1, their ratio matters to each other).

The safest motion can be generated using the *SVO* method. That velocity vector must be selected for the agent at every sampling time which has the largest distance from nearest VO cone and it is naturally in *RAV* set. In (3.13), this distance is calculated for every obstacle. In that case, the safest velocity vector can be calculated. $D_S(\mathbf{v}_\mathrm{A})$

means the distance which is minimum and $\mathbf{v}_{VO}$ represents the point on the VO cone:

$$D_S(\mathbf{v}_\mathrm{A}) = \min_{\mathbf{v}_{VO} \in VO} ||\mathbf{v}_\mathrm{A} - \mathbf{v}_{VO}|| \tag{3.13}$$

$D_S(\mathbf{v}_\mathrm{A})$ is changeable, because if the VO set of the obstacles is really far from the possible velocity vector, then it does not influence the velocity selection at the actual sampling time:

$$D_S(\mathbf{v}_\mathrm{A}) = \min(D_S(\mathbf{v}_\mathrm{A}),\, D_{max}) \tag{3.14}$$

where D_{max} is the given maximum distance.

$D_S(\mathbf{v}_\mathrm{A})$ can be normalized into the interval of [0,1]. The $C_S(\mathbf{v}_\mathrm{A}, VO)$ in (3.12) can be specified:

$$C_S(\mathbf{v}_\mathrm{A}, VO) = 1 - \frac{D_S(\mathbf{v}_\mathrm{A})}{D_{max}} \tag{3.15}$$

because the cost function must be minimized.

In the situation when the *RAV* set consists a velocity vector for the agent, then the velocity vector can be selected by the agent using the introduced cost-function. The C_S, which has the minimum value, can result in the safest motion in the workspace of the robot considering the actual situation of the different obstacles.

The next part of the cost function presents a part where the speed plays a role in the motion planning algorithm, it is presented by $C_G(\mathbf{v}_\mathrm{A})$:

$$C_G(\mathbf{v}_\mathrm{A}) = \frac{||\mathbf{p}_A + \mathbf{v}_A T_s - \mathbf{p}_{goal}||}{||\mathbf{p}_A(0) - \mathbf{p}_{goal}||} \tag{3.16}$$

where T_s has a meaning of the sampling time. $C_G(\mathbf{v}_\mathrm{A})$ represents the distance between the target position and the position where the agent can go using the selected velocity vector for a sampling time. This distance must be always normalized by the initial distance between the agent and the target position. The reachable velocities (RV) set can be discretized. In that case, the velocity can be selected from a grid in every timestep.

3.1.3 Experiments and Results

In this section, the results of the SVO method are presented. The SVO method was compared with two well-known motion planning algorithms, the APF method (which was presented in Section 2.2) and the VOTG method which were presented in Section 2.1.

Figure 3.2 represents the distances between the agent and the obstacle and Figure 3.3 shows the resulted paths considering the VOTG method and the SVO algorithm

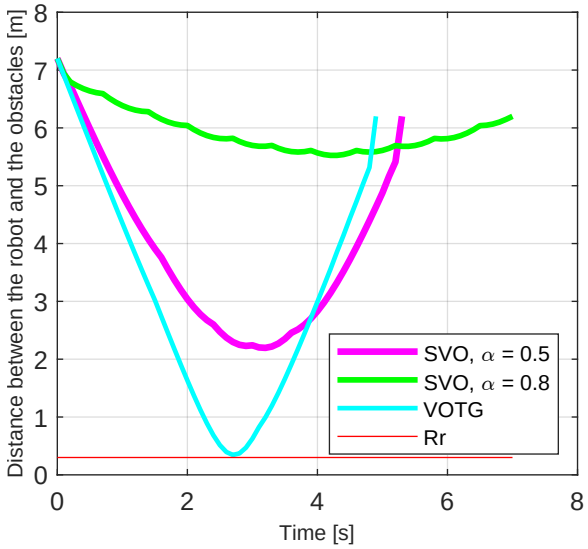

Fig. 3.2 Distances between the agent and the obstacle during the motion, where R_r means the radius of the robot, the distance is between the center point of the agent and the periphery of the obstacle (because the obstacles can have different radii).

with different α parameters. The start position for the agent was in the origin and the goal position was in [15; 0], there is one static obstalce in the workspace in position of [8; 0.5]. It is notable that the VOTG method generates the fastest solution resulting in a tangential motion to the obstacle (distance between the robot and obstacle is 0 in this case) but considering sensor estimation error, this maneuver could result in a collision immediately. Using $\alpha = 0.5$ and $\beta = 0.5$ parameters in the cost function, it generates an evasive maneuver resulting in a bigger distance from the obstacle. In the third scenario, using $\alpha = 0.8$ and $\beta = 0.2$, it generates the highest distance, the safest solution but the longest motion at the same time. The exact value of the α parameter can be always determined considering the actual situation of the environment because as it is presented in this result, it represents the actual safety parameter of the motion of the mobile robot. The basic APF method cannot result in a target reaching motion in this example because the obstacle is in the line of the start position of the agent and the target position, resulting in an oscillating motion at the APF method.

In Table 3.1, I present a detailed comparative analysis of the Artificial Potential Field (APF) method, the Safety Velocity Obstacles (SVO) method, and the VOTG method, focusing on average running time and the number of collision across 100 different scenarios. In the different scenarios, there were 10 obstacles, moving in different directions and changing their velocity vectors randomly. The APF method exhibits varying performance depending on the choice of parameters. For instance, with parameters $\gamma = 0.1$ and $\rho = 6$, the APF method achieved an average running time (to reach the goal position) of 98.43 s and recorded 3 accidents. However, it is

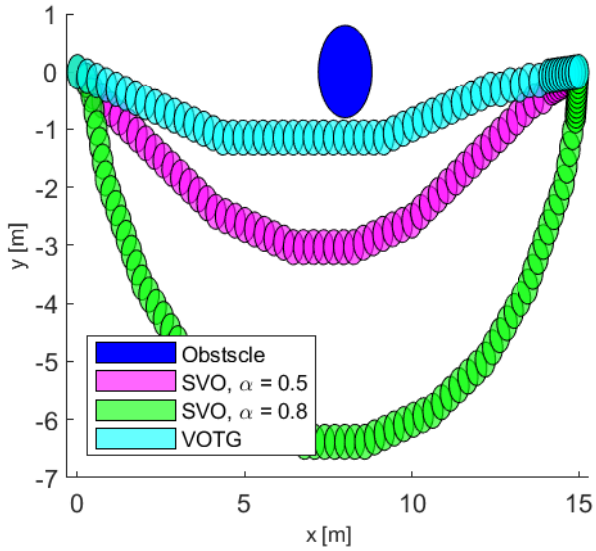

Fig. 3.3 The paths of the agent using different parameters.

observed that as the parameter γ decreases to 0.01, the average running time increases significantly, and the number of accidents also tends to increase. Considering the least accidents, the best result can be reached by using parameters $\gamma = 0.1$ and $\rho = 3$, resulting in 50.61 s average time and 2 collisions. Considering the average running time, the best result can be reached by using parameters $\gamma = 0.1$ and $\rho = 2$, resulting in 38.79 s average time and 5 collisions.

On the other hand, the SVO method demonstrates remarkable performance, particularly with $\alpha = 0.5$ and $\beta = 0.5$, where it registers an average running time of 32.089 s and, notably, zero accidents. This is a significant achievement in terms of both efficiency and safety. The average running time is less than the best-performing APF configuration (considering the least number of collisions), and the absence of accidents highlights the robustness of the SVO method in ensuring safe navigation.

The VOTG method achieved the lowest average running time of 30.91 s but recorded a significantly higher number of accidents (8), which raises concerns regarding its reliability in scenarios where safety is important. Even if the VOTG algorithm selects velocity vectors from the RAV set, it can result in a collision because it selects the velocity vector that can reach the fastest solution but if the obstacle changes its velocity vector randomly, the method cannot ensure the collision-free motion anymore.

Taking into account both the safety aspect, as indicated by the number of accidents, and efficiency, as indicated by the average running time, the conclusion can made that the SVO method is the most effective reactive motion planning algorithm among the ones considered in this investigation. Its ability to ensure safe navigation while maintaining high efficiency makes it a promising choice for applications where both safety

Table 3.1 Average run times and number of accidents depending on the APF and SVO parameters. The best parameter set is highlighted.

	γ	ρ	α	Average time [s]	Collisions
APF	0.1	6	-	98.43	3
	0.1	5	-	79.5	3
	0.1	4	-	63.86	2
	0.1	**3**	-	**50.61**	**2**
	0.1	2	-	38.79	5
	0.1	1	-	42.42	22
	0.01	6	-	144.41	49
	0.01	5	-	138.93	25
	0.01	4	-	132.06	12
	0.01	3	-	118.47	7
	0.01	2	-	118.02	17
	0.01	1	-	87.24	30
SVO	-	-	0.2	31.06	5
	-	-	**0.5**	**32.089**	**0**
	-	-	0.8	39.85	0
VOTG	-	-	-	**30.91**	**8**

and performance are critical. It must be noted, that the selection of the parameters plays a huge role in the motion planning algorithm. The automatic selection of these parameters will be discussed later.

A Chinese robotics and AI laboratory introduced a novel motion planning algorithm in 2022, called the Improved Speed Barrier Method [127] based on my Safety Velocity Obstacle method (they called it Adaptive Speed Barrier Method). Figure 3.4 compares a classic Velocity Obstacle (speed barrier) method combined with a dynamic obstacle avoidance path window, an adaptive speed barrier method (SVO method) for the obstacle avoidance path, the algorithm proposed in the Chinese lab for the obstacle avoidance path, where the blue path is the traditional VO method, black is the SVO method, and red is the Improved algorithm. It must be noted, that it is not known, which α parameter was used in the SVO method. In combination with the results reported in Table 3.2, it can be seen that the traditional obstacle avoidance method did not consider safety in the obstacle avoidance process, so it chose the path with the fastest path when discovering and avoiding obstacles. Although it took less time and the length of the path was shorter, as it came closer to the obstacle, the method cannot be considered conducive to dealing with unexpected situations in the driving process. As it can be seen, the SVO and the improved method need the same timesteps to achieve the goal position, the path of the agent and the minimal distance from

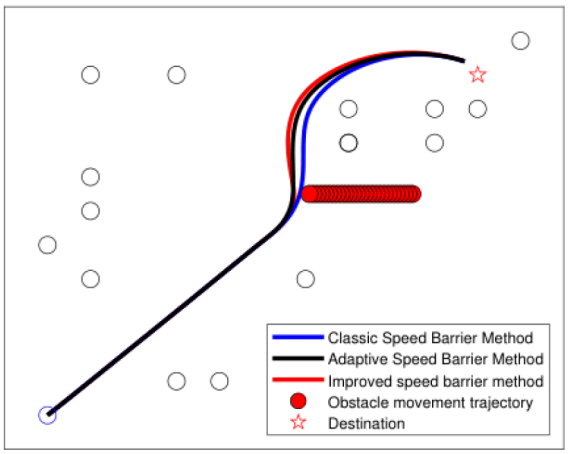

Fig. 3.4 Result of the Improved Speed Barrier method [127].

Table 3.2 Comparison between the VO, SVO and the Improved Speed Barrier method [127]

Algorithm	Timesteps	Path length [m]	Closest distance to Obstacle [m]
Normal VO	348.0	16.8247	0.4113
SVO	360.1	17.5441	0.5331
Improved Speed Barrier	359.8	17.5525	0.5558

the closest obstacle is slightly less using the SVO method, generating a safe target reaching.

3.1.4 Different Value Safety Velocity Obstacle (DVSVO) method

The SVO can be further developed using different safety parameters for the different obstacles, introducing the Different Value Safety Velocity Obstacle (DVSVO) method. Introducing the new α_i parameter, means that the different obstacles have different values considering the reliability aspect in the workspace using the sensor data or there are other robots in the workspace who share the position information with the agent and these data are more reliable than those which was collected using the on-board sensor.

$$Cost(\mathbf{v}_\mathrm{A}) = \sum_{i=1}^{m} \alpha_i \, C_S(\mathbf{v}_\mathrm{A}, VO_i) + \beta \, C_G(\mathbf{v}_\mathrm{A}) \tag{3.17}$$

where α_i presents the safety parameter of the measurement of the $\mathbf{p}_{Bi}$ and the sum of α_i must be 1. It also shows how close the agent can generate its motion next to the obstacle and $\alpha_i \geq 0$, $\beta \geq 0$.

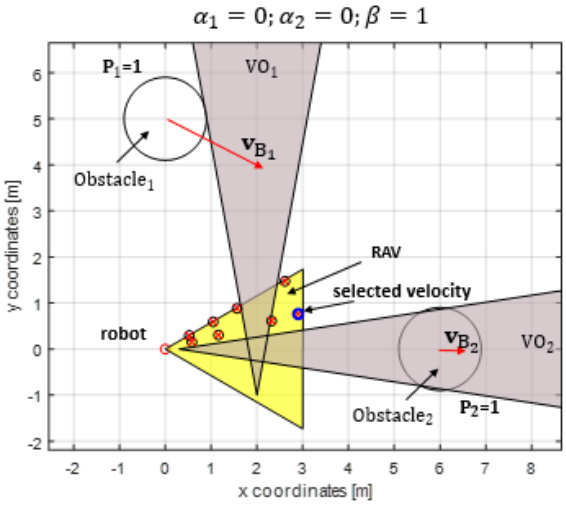

Fig. 3.5 Result at a sampling time with parameter set of $\alpha_1 = 0; \alpha_2 = 0; \beta = 1$, the target position is in [15; 0]

The agent will execute a maneuver near the obstacles that have high reliability (small α_i number) and it will generate its motion far from the obstacles if their position and velocity vectors are not so reliable. In this case, the different obstacles can be divided considering the aspect of reliability. If the value of α_i parameters is chosen to be the same then we cannot receive (3.12) back, because it calculates only the closest distance of VO_i, and here every VO_i distance exists.

In this sequel, an example will be presented for the novel cost function-based velocity selection. There are two moving obstacles in the workspace of the agent.

The velocity selection at a sampling time is presented in Figure 3.5. The velocities on the grid, are depicted with red x-shapes in black circles. The VO cones are presented in grey color, the RAV set is yellow colored. The agent is in the origin, and the obstacles are depicted by black circles.

The obstacles have reliability considering the measurement data about the velocities and the position vectors. Define P_i as the reliability of an obstacle. To get the parameter α_i for using the cost function, which was defined in (3.17), at this aspect, it can be used with the value as $\alpha_i = 1 - P_i$.

In the first example as an assumption at both obstacles, the received measurement data are completely reliable ($P_1 = 1, P_2 = 1$). In that case, the fastest solution will be resulted as it is presented in Figure 3.5. With using the selected velocity vector, the robot will execute the fastest motion to the target position.

In the next example, the first obstacle has a low and the second obstacle has a high reliability considering the data of the measurements ($P_1 = 0.05, P_2 = 0.95$). As a result, the motion planning algorithm selects the velocity far from the first VO_1 cone

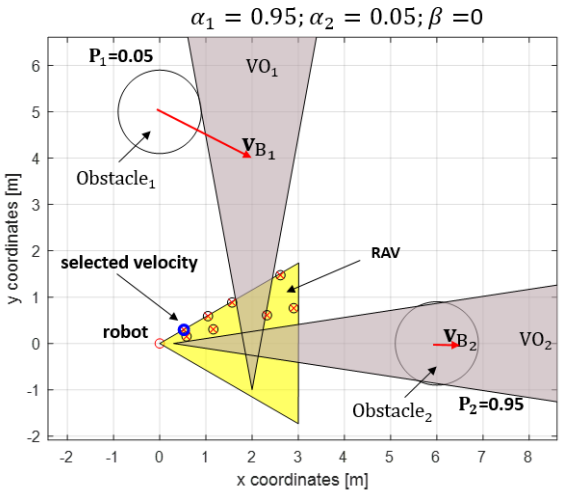

Fig. 3.6 Results with parameter set of with parameters of $\alpha_1 = 0.95; \alpha_2 = 0.05; \beta = 0$

and near to the second VO_2 cone as it is shown in Figure 3.6. If the speed (the value of β is big) has also a high influence for the motion planning at the same time, then the length of the velocity will be bigger but it will be selected considering the uncertainty of the obstacles as it is presented in Figure 3.7.

Another situation is if the first obstacle has a higher and the second obstacle has a lower probability $P_1 = 0.95, P_2 = 0.05$). The motion planning algorithm selects the velocity near the first VO_1 cone and far from the second VO_2 cone as shown in Figure 3.8. In this case, if the speed influences the motion then at the beginning of the motion planning algorithm the same velocity vector is selected as in the presented in the previous example in Figure 3.7.

It has been shown, that different strategies and uncertainties generate different velocity selection results. As a conclusion, the motions of the agent using different strategies are presented in a video YO[1]. It was also tested that a marker was introduced which means that a special agent is in the workspace and has the right to move in front of the agent.

3.2 Genetic Algorithm-based Velocity Obstacle Method

The primary challenge of techniques derived from the VO model is the selection of the velocity vector. All the vectors from the *RAV* set can be part of a collision-free trajectory, but they are not equally useful for the robot in its goal to reach the destination.

The idea for our genetic algorithm-based velocity-obstacle (GAVO) method is to calculate the VO and RAV areas and use a GA to evolve a solution that is feasible (i.e.

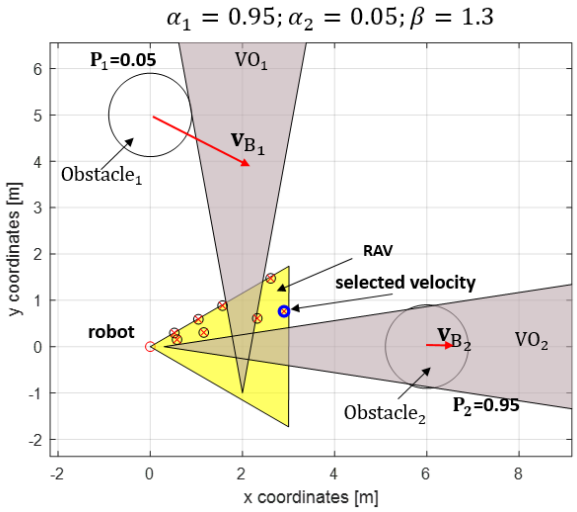

Fig. 3.7 Results with parameter set of $\alpha_1 = 0.95; \alpha_2 = 0.05; \beta = 1.3$

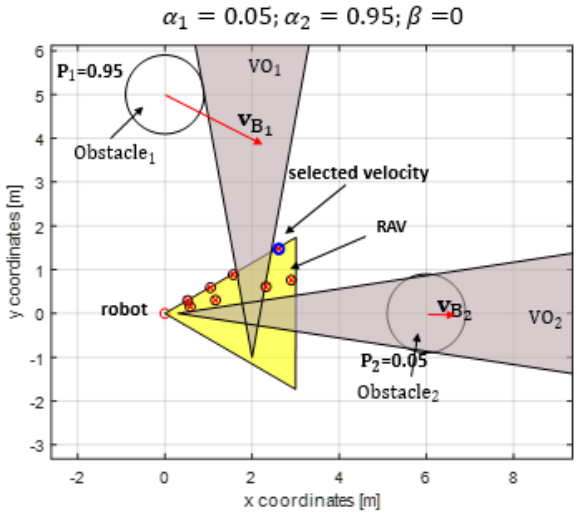

Fig. 3.8 Results with parameter set of $\alpha_1 = 0.05; \alpha_2 = 0.95; \beta = 0$

it is part of the RAV) while aiming to optimize the performance criteria desirable for the user.

The search space of the possible solutions is infinitely large. While this can be discretized by applying a grid as it was seen e.g. in Figure 3.7 to the *RAV* area, performing a high-resolution grid search can make the process prohibitively expensive. The GA algorithm allows us to search areas outside the grid and direct the search toward promising areas. However, GAs can also be computationally expensive. To achieve a good performance in the limited time allotted to the robot, the GAVO algorithm needs to make intelligent decisions about prefiltering solutions, the choice of the encoding as well as the recombination and mutation operators.

3.2.1 Fitness value

In GAVO, the individuals subject to selection are a representation of a robot velocity vector, while their fitness must reflect the performance of the given vector in fulfilling the objectives of the robot of staying safe and moving towards the destination. Accordingly, the fitness function I designed reflects the two components of speed and safety. Now, at the GAVO algorithm, the fitness function represents an objective function, that should be maximized, so the previously introduced cost function must be changed.

The speed component measures the progress towards the target position:

$$GO(\mathbf{v}_\mathrm{i}) = \frac{r_i \cos \Delta\theta_i}{v_{max}} \tag{3.18}$$

where v_{max} is the maximum velocity that the agent can reach considering the kinematic constraints. $r_i = ||\mathbf{v}_\mathrm{i}||$ is the length of the velocity vector while $\Delta\theta_i = \theta_{rg} - \theta_{rv_i}$ is the difference between the angle of the goal (θ_{rg}) and the angle of the velocity vector (θ_{rv_i}) as it can be seen in Figure 3.9.

The safety component of the fitness measures whether there is a risk of collision. It is sufficient to consider only the closest VO. Furthermore, a situation where the robot cannot reach even the closest obstacle during the considered time interval is considered perfectly safe, and keeping farther away would not improve the safety. These considerations lead us to the following expression:

$$SA(\mathbf{v}_\mathrm{i}) = \min\left(1, \frac{\min\limits_{\mathbf{v}_{VO} \in VO} ||\mathbf{v}_i - \mathbf{v}_{VO}||}{v_{max} T_{max}}\right) \tag{3.19}$$

Finally, I combine the speed and safety components of the fitness function as a weighted average, with the parameter β capturing the importance of the progress

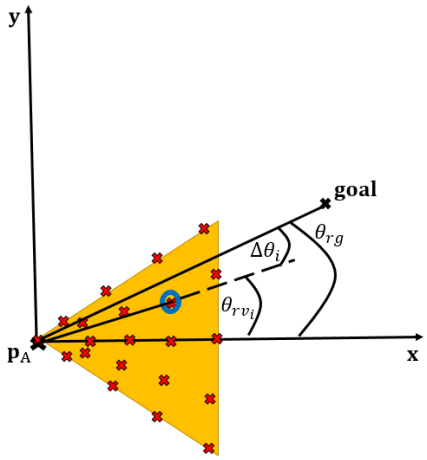

Fig. 3.9 Angles definition

towards the destination. For velocity vectors that are not in the *RAV* set, I artificially set the fitness to zero:

$$f(\mathbf{v}_i) = \begin{cases} \beta\ SA(\mathbf{v}_i) + (1-\beta)\ GO(\mathbf{v}_i) & \text{if } \mathbf{v}_i \in RAV \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

where $\beta \geq 0$.

3.2.2 Recombination method

In implementing the genetic algorithm component of GAVO, I experimented with three different recombination methods, all of them designed to take advantage of the problem domain and the chosen representation model. For all three techniques, the selection of the chromosomes chosen for recombination is performed using stochastic universal sampling [128].

In the *linear recombination* (GAVO-1D) method the new velocity vector is a random linear combination of the odd and even parents $\mathbf{v}_1$ and $\mathbf{v}_2$

$$\mathbf{v}_{\text{new}} = \mathbf{v}_1 + \kappa\ (\mathbf{v}_2 - \mathbf{v}_1) \tag{3.21}$$

where κ is a random parameter with a pre-defined range and it is the same for all of the alleles in the recombination.

In the *intermediate recombination* (GAVO-2D) method I am using two different random parameters κ_x and κ_y for the two components of the new vectors:

$$v_{newX} = v_{1x} + \kappa_x\ (v_{2x} - v_{1x}) \tag{3.22}$$

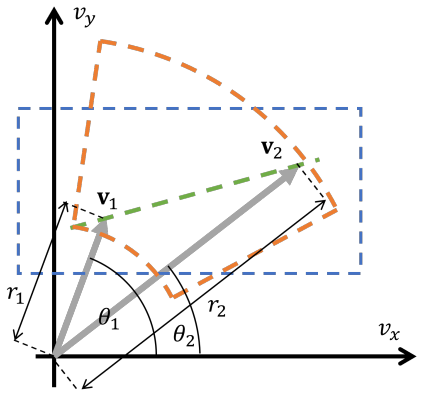

Fig. 3.10 Illustrating the three recombination techniques for the velocity vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. The green line shows the possible outcomes of the linear recombination, the blue area the possible endpoints of the intermediate recombination while the orange area the possible endpoints of the polar coordinate recombination method.

where $v_{2\mathrm{x}}$ and $v_{1\mathrm{x}}$ are the x components of the velocity vectors that are used for the recombination.

$$v_{newY} = v_{1\mathrm{y}} + \kappa_{\mathrm{y}} \left(v_{2\mathrm{y}} - v_{1\mathrm{y}}\right) \tag{3.23}$$

where $v_{2\mathrm{y}}$ and $v_{1\mathrm{y}}$ are the y components of the velocity vectors that are used for the recombination.

Finally, in the *polar coordinate recombination* method, the recombined vector is a result of a separate combination of the angle and radius in a polar coordinate representation of the two parent vectors:

$$v_{newX} = (\min(r_1, r_2) + \kappa_1 \left|r_1 - r_2\right|) \cdot \cos(\min(\theta_1, \theta_2) + \kappa_2 \left|\theta_1 - \theta_2\right|) \tag{3.24}$$

$$v_{newY} = (\min(r_1, r_2) + \kappa_1 \left|r_1 - r_2\right|) \cdot \sin(\min(\theta_1, \theta_2) + \kappa_2 \left|\theta_1 - \theta_2\right|), \tag{3.25}$$

where r_1 and r_2 mean the length and θ_1 and θ_2 mean the angles of the velocity vectors that are used for the recombination $|r_1 - r_2|$ and $|\theta_1 - \theta_2|$ are the absolute values of the differences.

Figure 3.10 illustrates the three recombination methods.

3.2.3 Mutation

The mutation technique I use is based on adding random values γ_1 and γ_2 drawn from a predefined range to the two components of the velocity vector representing the chromosome.

$$v_{newX} = v_{1\mathrm{x}} + \gamma_1 \tag{3.26}$$

$$v_{newY} = v_{1x} + \gamma_2 \tag{3.27}$$

In scenarios where recombination is being used, mutation is applied to a small random fraction of the resulting vectors. In scenario where no recombination is used, mutation is applied to every phenotype.

3.2.4 Experiments and Results

In this section, I describe experiments that compare variations of the GAVO approach along the dimensions of the quality of solution found and computational cost. We are particularly interested whether the approach can reach the optimal solution (or get very close to it) and whether the technique is suitable for real-time operation. For our purposes, we define as real-time a path planner which can reach decisions faster than the sensors' acquisition rate, which, in the case of the robot architecture we are considering is 10 Hz corresponding to a 100 ms time left for the computation.

The variations of the GAVO algorithm we are considering are as follows:

- GAVO-1D: uses the 1D recombination method with $\kappa \sim [-0.25, 1.25]$ and mutation rate $1/N$, where N means the number of individuals.
- GAVO-2D: uses a 2D recombination method with $\kappa_x \sim [-0.25, 1.5]$, $\kappa_y \sim [-1, 1]$ and mutation rate $1/N$, where the parameters were set using empirical results.
- GAVO-POLAR: uses the polar coordinate recombination method with $\kappa_1 \sim [-0.15, +0.15]$, $\kappa_2 \sim [-5, 5]$ and mutation rate $1/N$.
- GAVO-MUT: mutation only - uses mutation at a rate of 100% and does not perform recombination.

These variants have been tested with different parameters. The number of individuals in each generation N was chosen from the range of $[20 \ldots 200]$. The generation gap parameter *GAP* describing the level of elitism in the strategy was chosen from the range $[5 \ldots 10]$. To ensure a consistent comparison between the GAVO variants, in every experiment, every variation was started with the same initial population which was generated randomly. For the sake of consistence, in all our experiments we run the experiment for 100 generations, although usually only a subset of these generations would fit into the available time of 100 ms.

These algorithms were compared with two variations of the original velocity-obstacle algorithm: the previously introduced SVO method, which was introduced in Section 3.1 performs a grid search over the various locations outside the VO areas in

the pursuit of a similar fitness function, while VO-RandomSearch performs a random search on the grid.

For all algorithms, the experimental methodology involved (a) presenting the algorithm with a specific scenario (b) running the algorithm up to a specific point and (c) evaluating the best movement solution found by the algorithm up to that point. For the single shot algorithms VO-GridSearch and VO-Random, there is a single value; while for the GAVO approaches, we record the best available solution at every generation.

The algorithms were implemented in MATLAB 2021a. To keep the computational time measurements consistent, all the experiments were run on a computer with an Intel i5-3320M CPU with 8GB of memory. The memory was found sufficient for the computation making the experiments CPU-limited. No significant disk activity was observed during the experiment and no GPU acceleration was used. The capabilities of this system are comparable with a typical onboard computer of mobile robots.

A sparse environment

The first set of experiments were performed in a comparatively easy scenario that takes place in a sparse environment with only two obstacles. We assume that the robot uses LiDAR-based sensing as shown in Figure 3.11, with a particle filter-based estimator that will be introduced in Section 4.2. Figure 3.12 shows the scenario, the VOs, and the from RAV randomly selected initial population used by the GAVO variants. The fitness value can be converted to cost value by using 1-$f(\mathbf{v}_i)$.

One of the first considerations for a genetic algorithm is the evolution of fitness with the generations. While the use of elitism in the population guarantees that the maximum fitness will not decrease, there is no guarantee that the optimal solution will be reached in a finite time. Figure 3.13-left compares the fitness values for different GAVO variants for a scenario using a population size of $N = 20$ and $GAP = 10$. The algorithms started from the same population (the one in Figure 3.12), and were run for 100 generations. The figure also contains the fitness reached by VO-RandomSearch, which serves as a baseline, and the VO-GridSearch which, due to its exhaustive search nature, will serve as our optimal baseline. We find that there is a significant difference between the different GAVO variations. Except for GAVO-1D which did not find the optimum in the experiment, all the other algorithms find the optimal solution within 33 generations. The best performing algorithm was GAVO-2D, which reached the best solution in 23 generations.

As the robot needs to make movement decisions in real time, counting the generations in the GAVO optimization provides only part of the answer. We also need to

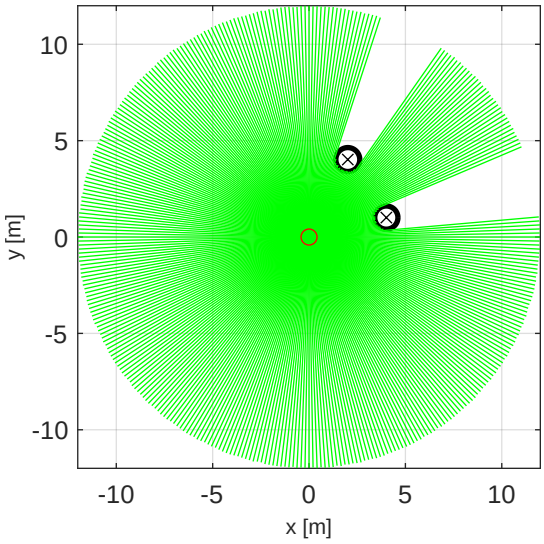

Fig. 3.11 LiDAR sensor data simulation from the workspace of the agent. The agent is in the origin, there are two obstacles in the workspace, represented by black circles, and the estimated positions of the obstacles are shown by black x-s. The maximum range of the LiDAR sensor is 12 m, the resolution is 0.5°.

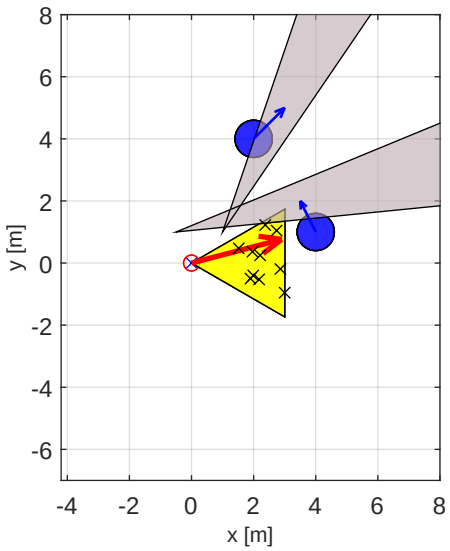

Fig. 3.12 VO-type diagram of the sparse scenario. The blue disks represent the obstacles with their velocities, while the gray triangles are the corresponding VOs. The x-marks distributed within the area of reachable velocities are the initial population of the GAVO variants. The fact that the area of reachable velocities only minimally overlaps with any VO shows that this is an "easy" scenario. The other signs in the Figure are the same as introduced previously. The selected velocity vector of the agent is presented with a red arrow.

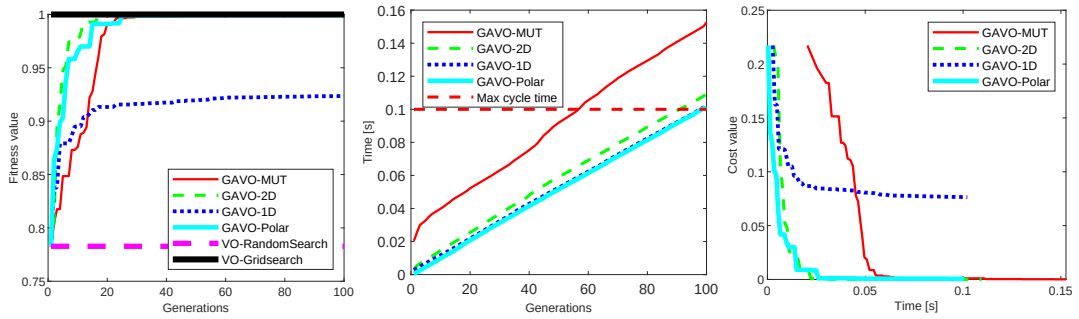

Fig. 3.13 Results of experiments with the sparse scenario for the variations of GAVO with $N = 20$ and $GAP = 10$. VO-RandomSearch and VO-GridSearch serves as baselines that do not change with the generation. (up left) Evolution of the fitness value with the generations, (up-right) time for a specific number of generations, (down) the trade-off between wall clock time.

investigate how long a certain number of generations take. This number obviously varies with the speed of the computer and the size of the population. We also speculate that it also varies for the different GAVO variants. Figure 3.13 describes the time at different generation points for the optimization. The figure also shows the important time point of the maximum cycle time which in our current setup is 0.1s (corresponding to the 10Hz sampling rate). In general, it is not practical to run the algorithm longer than this time. As expected, we find that the time spent increases roughly linear with the number of generations. The fastest versions are GAVO-1D and GAVO-Polar, which are essentially undistinguishable, with GAVO-2D being slightly slower, while GAVO-MUT being significantly slower. A way to interpret these results is that to remain in real-time we can run 98 generations of GAVO-1D and GAVO-Polar, about 93 for GAVO-2D and about 56 for GAVO-MUT. Note that this is sufficient for GAVO-Polar, GAVO-MUT and GAVO-2D to reach the optimal values.

Finally, another perspective is to consider time spent in optimizing versus previously intorduced cost value. In a real-time system, a system might often prefer a satisficing (suboptimal, but "good enough") solution that can be obtained quickly to an optimal solution that requires significantly more computation. Figure 3.13 shows the tradeoff between the wall clock time and the cost. We notice that the optimal result can be reached by GAVO-2D and GAVO-Polar in roughly the same time (about 26ms), while GAVO-MUT requested about 55.4ms to reach the same result. The figure also shows, that very close results, of cost of about 0.04 can be reached even faster, in about 10ms using GAVO-2D or GAVO-Polar.

The second series of experiments repeated the same experiments for a larger population of $N = 100$ with the results shown in Figure 3.14. We find that the overall shape and trends in the results are the same, however, the concrete values shifted,

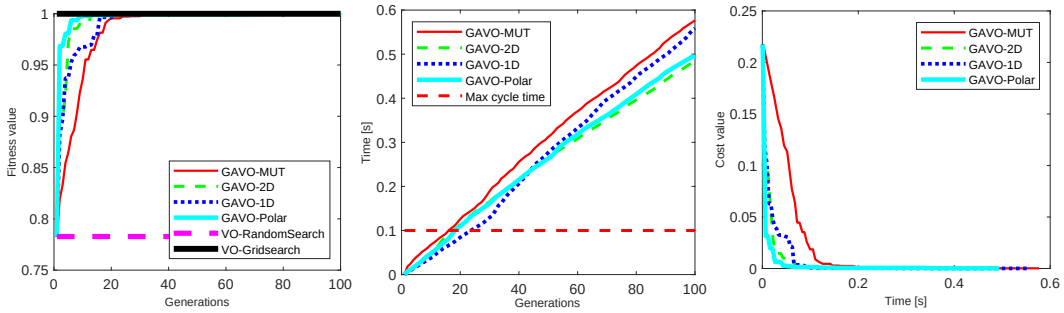

Fig. 3.14 Results of experiments with the sparse scenario for the variations of GAVO with $N = 100$ and $GAP = 10$. VO-RandomSearch and VO-GridSearch serves as baselines that do not change with the generation. (left) Evolution of the fitness value with the generations, (center) time for specific number of generations, (right) the trade-off between wall clock time and the cost.

which have significant implications over the real-time performance of the system. Due to the larger population, all the GAVO variants needed a smaller number of generations to reach the optimal solution. The most significant shift occurred for GAVO-1D which in the smaller population could not reach the optimum in 100 generations, but in this case it reached it in 22 generations, a better value than GAVO-MUT. The variant which required the lowest number of generations to reach the optimum remained GAVO-2D. The ordering for the time needed for a given number of generations (Figure 3.14-center) remained the same, however, the times are significantly longer, corresponding to the larger population. This means that we can run much smaller number of generation before we run into the real-time constraint. For the mutation-only strategy GAVO-MUT, the system cannot reach the optimum before the time runs out. The other three variants, however can reach the optimum before the maximum cycle time expires.

Finally, the wall-clock time / cost tradeoff also changed. While GAVO-POLAR and GAVO-2D remain the best choice, the GAVO-1D strategy is quite close to them, with the GAVO-MUT strategy achieving a significantly worse tradeoff.

The GAVO variants with different population sizes are all possible alternatives for a real time obstacle avoidance system. Figure 3.15 compares them for the time it takes for the system to reach the optimal solution. We found that all the combinations considered, except the GAVO-MUT variant with N=100 which takes longer than the maximum cycle time to reach the optimum, and GAVO-1D with N=20, which did not reach the optimum in our experiments, and thus is not present in this figure. The best time was reached by GAVO-2D with N=20. These values are better compared to previous GA based algorithms for path planning in dynamic environments, such as [80, 81]); however a direct comparison is not possible as the computational power

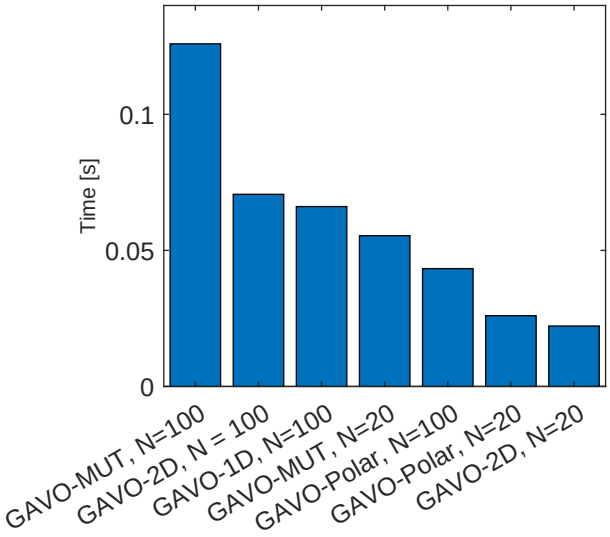

Fig. 3.15 Comparison of the wall-clock time to optimal solution for GAVO variants with different parametrizations.

of typical machines had significantly increased in the decade since the development of those algorithms.

We also examined the impact of varying the number of individuals (N) on fitness value outcomes in both GAVO-MUT and GAVO-Polar methods. Figure 3.16-left depicts the fitness value outcomes of the GAVO-MUT method using different values of N. Notably, the results demonstrate that the fitness value outcomes are not significantly influenced by the number of individuals in use; rather, the optimal fitness value can be achieved within approximately 20 generations. While the number of individuals does not substantially affect the number of generations required to attain the optimal fitness value, it does play a critical role in the overall running time of the algorithm. As depicted in Figure 3.15, a real-time solution cannot be attained using 100 individuals. Thus, the GAVO-MUT method necessitates the use of even fewer individuals to attain acceptable running time outcomes.

Figure 3.16-right presents the outcome of the GAVO-Polar method. In most instances, the optimal solution can be obtained within 20 generations; however, if N is set to a smaller value (20), it takes 38 generations to reach the optimal solution. Notably, based on the outcomes depicted in Figure 3.13 and Figure 3.14, the GAVO-Polar method can achieve a real-time solution in all scenarios. Therefore, the user can define the number of individuals based on other relevant considerations, and the optimal solution can be reached in all cases.

We conducted an investigation into the impact of varying the GAP parameter between [5; 10] on the fitness value using both the GAVO-MUT and GAVO-Polar methods. The results are presented in Figure 3.17. The curves of the lines in the

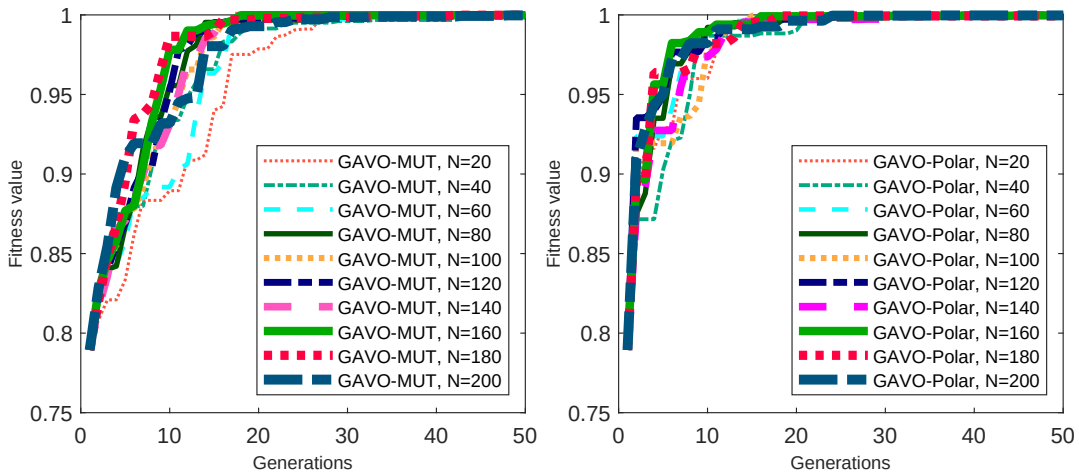

Fig. 3.16 Results of experiments with the sparse scenario for the various number of individuals N. (left) Evolution of the fitness value with the generations using the GAVO-MUT method, (right) Evolution of the fitness value with the generations using the GAVO-Polar method.

GAVO-MUT method (left side) are similar across different *GAP* values; however, it is evident that the optimal fitness value is achieved after more generations compared to the GAVO-Polar method (right side). Therefore, the GAVO-Polar method appears to be a superior choice for the velocity selection task.

Crowded environment

The second series of our experiments considered a "crowded" environment, as shown in Figure 3.18. Note that this time there are four obstacles, with three dynamic obstacles that are heading roughly towards the current position of the robot. As the figure shows, the feasible set of velocities are almost completely covered by the VOs, making this a difficult scenario. The current location of the robot actually falls within one of the VOs, thus the robot needs to move in order to avoid colliding with one of the mobile obstacles. Finally, the goal position, at location [8,0], is obstructed by one of the obstacles, thus in the initial position the target is not visible for the robot.

I run a series of experiments for the four GAVO variants, with different values for the population size N (20, 50 and 100) and the GAP parameter (5 and 10). The results are listed in Table 3.3. In this table, *Max Gen.* is the maximum number of generations that can be executed during the maximum cycling time of 0.1s. *Best Gen.* is the first generation from which the Fitness value is not changing during the algorithm - a value marked as *Best Fitness*. Best time is the time needed to reach this value.

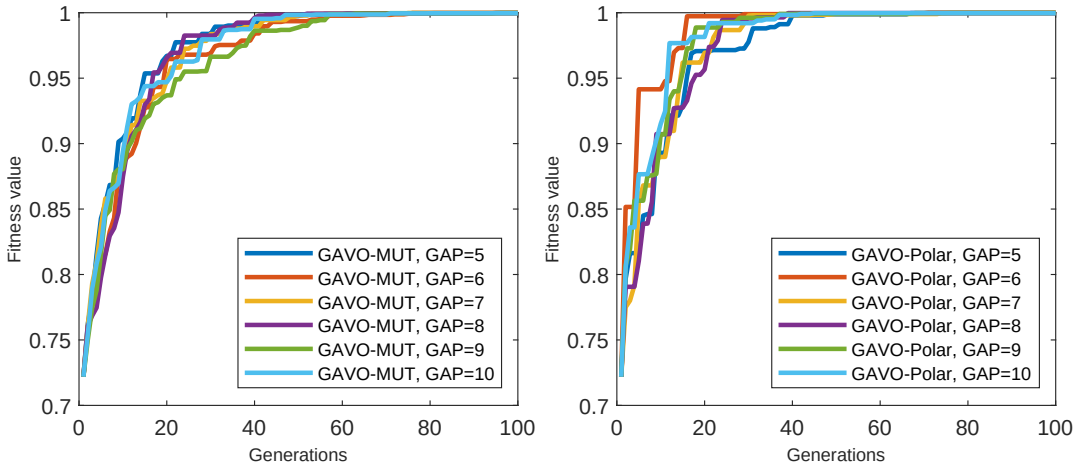

Fig. 3.17 Results of experiments with the sparse scenario for the different *GAP* parameters. (left) Evolution of the fitness value with the generations using the GAVO-MUT method, (right) Evolution of the fitness value with the generations using the GAVO-Polar method.

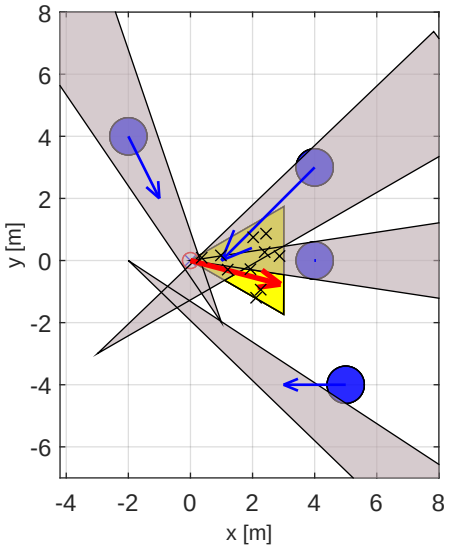

Fig. 3.18 VO-type diagram of the crowded scenario. The robot is positioned in the origin and is denoted by a red circle, the goal position is at [8,0]. The fact the majority of the yellow area is covered by the VOs shows that this is a "difficult" scenario.

The four different recombination methods were compared in this situation as well. Table 3.3 represents the results using different parameters at the genetic algorithm. Variant means the actual recombination strategy that was executed.

Sometimes, the GAVO-1D recombination method cannot reach the best solution in real-time, e.g.: $N = 20$, $GAP = 10$ (reaching the best Fitness value in 0.1485 s), or using the parameters $N = 50$, $GAP = 5$ (resulting the beast Fitness value in 0.1526 s). The best solution can be reached considering the least number of generations using the GAVO-2D recombination method with parameters of $N = 100$, $GAP = 5$, resulting the best Fitness value in 5 Generations. The fastest solution can also be reached considering the previous parameter set, resulting in a solution of 0.0133 s. Comparing all of the methods, the Best Fitness value is 0.67 which can be reached using the GAVO-1D method with parameters of $N = 100$, $GAP = 10$. To sum up, the result of the Best Fitness values comparing the different parameters in different recombination methods are quite close to each other and usually, they can be reached in real-time (except in some cases which were presented).

Finally, it is of interest to compare the trajectories generated by the different GAVO algorithm variants. Figure 3.19 shows the successive locations generated by two baseline heuristics (TG and MV strategies) and several GAVO variations with different values of β parameter. The TG heuristic (in cyan in the figure) finds the fastest path from the ones that directly head toward the goal. In this scenario, the goal is obstructed by a static obstacle, thus the robot will pick a trajectory that gradually comes to a stop in front of the static obstacle and never reaches the goal. The MV heuristic, in green in the figure, finds a strategy that allows the robot to move with the fastest velocity, possibly not directly in the direction of the goal. This trajectory reaches the goal, however, it takes a risky trajectory where the robot gets to a zero distance from the static obstacle. In this scenario, any sensor noise can lead to a collision. The same trajectory would be obtained by the GAVO method with parameter β=1. By varying the β parameter, we can trade off the speed and safety of the trajectory found by GAVO, with β=1 (magenta) corresponding to the safest trajectory, and β=0.7 an intermediate solution balancing speed and safety.

We have seen that the overall performance of the variations of the GAVO technique achieves a good result (in many situations finding an optimal solution) while they can also provide a real-time algorithm, provided that the hyperparameters are carefully chosen. Our current ability to deploy techniques such as GAVO in real-time is due to two separate developments: the increase in performance of computational devices that can be deployed in a robot (at least three to four orders of magnitude) and the better understanding of genetic algorithms and their dependence on representation and hyperparameters. For instance, Figure 3.15 shows a factor of 8-times computational

Table 3.3 Results of experiments on the crowded scenario using different GAVO variants, population sizes N and GAP parameters

N	*GAP*	Variant	Max Gen.	Best Gen.	Best Time [s]	Best Fitness
20	5	GAVO-MUT	45	13	0.0621	0.7000
20	5	GAVO-2D	97	7	0.0136	0.7000
20	5	GAVO-1D	103	38	0.0427	0.6800
20	5	GAVO-Polar	100	15	0.0209	0.7000
20	10	GAVO-MUT	51	17	0.0536	0.7000
20	10	GAVO-2D	69	10	0.0257	0.7000
20	10	GAVO-1D	80	124	0.1485	0.6800
20	10	GAVO-Polar	73	16	0.0228	0.7000
50	5	GAVO-MUT	42	12	0.0402	0.7000
50	5	GAVO-2D	42	8	0.0190	0.7000
50	5	GAVO-1D	55	76	0.1526	0.6900
50	5	GAVO-Polar	49	15	0.0340	0.7000
50	10	GAVO-MUT	36	14	0.0384	0.7000
50	10	GAVO-2D	32	9	0.0190	0.7000
50	10	GAVO-1D	38	28	0.0770	0.6900
50	10	GAVO-Polar	42	16	0.0374	0.7000
100	5	GAVO-MUT	31	12	0.0369	0.7000
100	5	GAVO-2D	32	5	0.0133	0.7000
100	5	GAVO-1D	31	14	0.0457	0.6800
100	5	GAVO-Polar	32	17	0.0517	0.7000
100	10	GAVO-MUT	27	11	0.0400	0.7000
100	10	GAVO-2D	28	15	0.0140	0.7000
100	10	GAVO-1D	23	7	0.0254	0.6700
100	10	GAVO-Polar	31	18	0.0577	0.7000

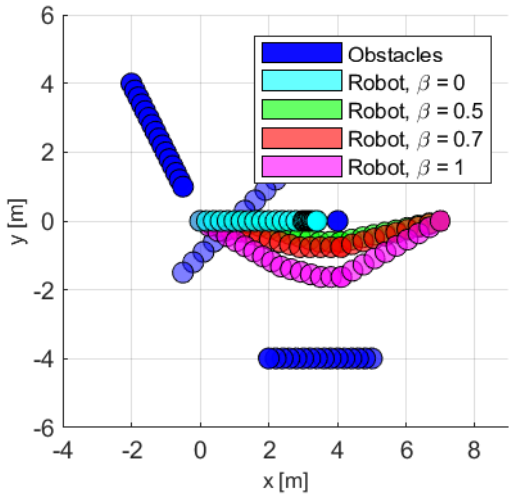

Fig. 3.19 The successive locations of the robot (starting from $\mathbf{p}_{\mathrm{A}} = (0\mathrm{m}, 0\mathrm{m})^T$) and obstacles outlining the paths taken by them during the scenario. Blue: obstacles (B_1 is a static obstacle at $\mathbf{p}_{\mathrm{B1}} = (4\mathrm{m}, 0\mathrm{m})^T$. B_2 moves from $(-2\mathrm{m}, 4\mathrm{m})^T$ to $(-0.5\mathrm{m}, 1\mathrm{m})^T$. B_3 moves from $(4\mathrm{m}, 3\mathrm{m})^T$ to $(-0.7\mathrm{m}, -1.45\mathrm{m})^T$. B_4 moves from $(5\mathrm{m}, -4\mathrm{m})^T$ to $(2\mathrm{m}, -4\mathrm{m})^T$.). Cyan: the robot using VO with the TG heuristic. Green: the robot using VO with the MV heuristic, identical with GAVO using $\beta = 1$. Red: the robot using GAVO with $\beta = 0.7$. Magenta: robot using GAVO with $\beta = 0$.The target position is $\mathbf{p}_g = (9\mathrm{m}, 0\mathrm{m})^T$

cost difference between the best-performing vs. worst-performing GAVO variants. Undoubtedly, there are further possibilities for performance increase.

A possible weakness of the proposed approach is the fact that, as with any stochastic search technique, it offers only probabilistic guarantees about reaching a specific performance level in a given amount of time. Nevertheless, this weakness is partially compensated by the anytime nature of the algorithm and the elitism in the selection strategy that allows the algorithm to return at any given moment a good although possibly not optimal solution.

3.3 Discussion

I introduced the Safety Velocity Obstacle method that can consider the safety and the speed at the same time in the motion planning algorithm using a cost function. The algorithm was compared with the VOTG and APF methods, showing better performance considering the collision avoidance aspect. The introduced algorithm was also main idea of Chinese researchers who introduced a novel cost function-based algorithm. The SVO method was further developed using different reliability parameters for every obstacle resulting in different motions of the agent to the different obstacles. The Genetic Algorithm Velocity Obstacle method was introduced as an Artificial Intelligence method for the collision avoidance task. It is important to note that the GAVO can be executed in real time reaching the same solution as the SVO method. All of these algorithms were introduced using omnidirectional robots in a dynamic environment. As a next step, the introduced novel motion planning methods could be introduced also on different types of robots. With colleagues, we implemented the SVO method to differential-driven robots as well. Both the SVO and GAVO methods can be used if the safety and speed should be considered at the same time. The DVSVO method can be used if there is information about the reliability of the different obstacles (e.g.: they share their information or it can be measured.)

Thesis group I.

I introduced the Safety Velocity Obstacle method, the Different Value Safety Velocity Obstacle method, and the Genetic Algorithm Velocity Obstacle method, which approaches generated a solution where the agent's velocity selection is guided by a cost-function-based method in a dynamic environment. These methods incorporate predefined parameters for safety and speed to optimize the robot's navigation in a dense dynamic environment. Through simulations, I have shown that the introduced methods

can achieve better performance and robustness in different scenarios compared with well-known motion planning algorithms considering the number of collisions and running time.

Thesis I.1

I introduced the Safety Velocity Obstacle approach where the agent's velocity selection is guided by a cost-function-based method using the safety and speed predefined parameters. The motion planning method can generate a collision-free path for the agent to the goal position. Simulations have shown that it has better performance in target reaching than the APF or the VOTG algorithms considering the main aspects as the number of collisions and running time. Building on the aforementioned method, I further enhanced its application by introducing the Different Value Safety Velocity Obstacle method, by employing variable safety parameters for distinct obstacles encountered in the robot's workspace.

Thesis I.2

I have devised a novel Artificial Intelligence-driven motion planning technique for mobile robots by merging the principles of the Genetic Algorithm and the Safety Velocity Obstacles method, called Genetic Algorithm Velocity Obstacle method. The introduced method can achieve the best results using the intermediate and polar recombination methods considering the running time and number of collisions compared with other algorithms.

Journal publications to the thesis [J]: [1], [2], [3]
Conference publications to the thesis [C]: [1], [2], [3], [4]
In progress [IP]: [1]

Chapter 4

Path planning and state perception considering uncertainties

The motion planning of mobile robots in dynamic environments poses significant challenges as was seen in Section 1.2. To generate effective evasive maneuvers, mobile robots may utilize internal or external sensors to gather data about the environment.

By applying state estimation methodologies to this sensor data, the state of the obstacles in the workspace can be estimated at discrete time intervals. For the VO motion planning method, not only the position but also the velocity vectors of the obstacles must be known. There exists a variety of estimation methods that can be employed for this purpose. Uncertainty plays also a huge role in the motion planning of mobile robots. In Section 4.1, the Uncertainty Velocity Obstacle method is presented, later on, Section 4.2 shows the perception methods for obstacles using the Kalman filter and Particle filter. Then Section 4.3 the Particle filter-based perception method is combined with the SVO method to generate the perception of the environment and the collision avoidance at the same time.

4.1 Uncertainty Velocity Obstacle (UCVO) method

Using the reactive motion planning algorithm, it is challenging to generate evasive maneuvers that can ensure a safe motion for the agent and its environment as it was seen previously as well. The task is more difficult if the uncertainties of the measured data (position and velocity vectors) are taken into consideration. In this section, a novel reactive motion planning method, the Uncertainty Velocity Obstacle method is introduced that can calculate the uncertainties for every obstacle using their velocities and distances from the agent.

4.1.1 Calculation of changing uncertainties

The uncertainties can be calculated using the measured data. The main idea is that the measured information has a higher reliability if the obstacles are closer to the agent. First, the distance-based reliability term can be calculated:

$$P_{dist_i} = \max(1 - \frac{dist_{OR_i}}{D_{max}}, 0) \tag{4.1}$$

where $dist_{OR_i}$ is actual distance between the obstacle and the agent, and D_{max} is the same notation as it was introduced in (3.14).

The magnitude of the velocity of the obstacle plays also a role in the calculation of the uncertainties. The introduced method focused on the calculation of the distance between the agent and the obstacle, which can vary to a greater extent at higher obstacle speeds, and at higher obstacle speeds the agent has less time and opportunity to change its movement, thus avoiding a possible collision.

$$P_{MV_i} = \max(1 - \frac{||\mathbf{v}_{\mathrm{Bi}}||}{v_{max}}, 0) \tag{4.2}$$

where P_{MV_i} is the velocity based reliability term, where $||\mathbf{v}_{\mathrm{Bi}}||$ means the actual magnitude of the velocity of the obstacle.

The change of the velocity of the obstacle also influences the uncertainties. The change of the velocity of the obstacle can be calculated:

$$CV_i = ||\mathbf{v}_{\mathrm{Bi,k}} - \mathbf{v}_{\mathrm{Bi,k\text{-}1}}|| \tag{4.3}$$

where CV_i means the change of the velocity of the obstacle, $\mathbf{v}_{\mathrm{Bi,k}}$ the actual velocity of the obstacle, and $\mathbf{v}_{\mathrm{Bi,k\text{-}1}}$ is previous velocity of the obstacle.

$$P_{CV_i} = \max(1 - \frac{CV_i}{v_{max}}, 0), \tag{4.4}$$

where P_{CV_i} is the reliability term depending on the change in the velocity of the obstacle.

I used data derived from the available information to calculate the cost function components underlying the motion planning. The main goal was to maintain a larger distance between the agent and the obstacle, hence the defined factors were used to calculate the reliability. The reliability can be calculated:

$$P_i = \frac{P_{dist_i} + P_{MV_i} + P_{CV_i}}{3} \tag{4.5}$$

From the calculated reliability the degree of uncertainty can be calculated:

$$\alpha_i = 1 - P_i, \tag{4.6}$$

where this α_i uncertainty parameter must be calculated for every obstacle ($i = 1...m$, if there are m obstacles in the workspace). The α_i parameter is not constant during the whole motion planning algorithm, it is recalculated in every sampling time.

4.1.2 Cost function

The cost function (3.12) which was introduced in the SVO method (Section 3.1) was extended by using changing $\alpha_i(t)$ parameters (they are calculated in every sampling time) for the different obstacles in consideration of the reliability of the velocity and position information of the obstacles.

The angle of the actual velocity vector of the agent (presented in Figure 3.9) plays also a role in the cost function. The heading value of the cost function can be calculated:

$$C_h(\mathbf{v}_\mathrm{A}) = \frac{|\theta_{rg} - \theta_{rv_A}|}{\pi}, \tag{4.7}$$

The same criteria apply to the cost function as demonstrated in Section 3.1.

The whole cost function can be determined using the previously introduced parameters:

$$Cost(\mathbf{v}_\mathrm{A}) = \sum_{i=1}^{m} \alpha_i(t)\, C_S(\mathbf{v}_\mathrm{A}, VO_i) + \beta_d\, C_G(\mathbf{v}_\mathrm{A}) + \beta_h\, C_h(\mathbf{v}_\mathrm{A}) \tag{4.8}$$

where $\alpha_i(t)$ means the actual calculated uncertainty parameter of an obstacle; β_d is the distance parameter, and β_h is the heading parameter.

The different parameters of the cost function have a huge impact on the velocity selection, as it will be presented in Section 4.1.3.

4.1.3 Experiments and Results

In this section some results of the simulation of the motion planning algorithm will be presented considering the changing uncertainties.

Two static obstacles

In the first example, there are two static obstacles. In that case, using the introduced cost function, the algorithm will select a velocity vector for the robot that is exactly in the middle of the two obstacles because the two obstacles have the same uncertainties. This situation is presented in Figure 4.1, where the VO_i are presented with the grey

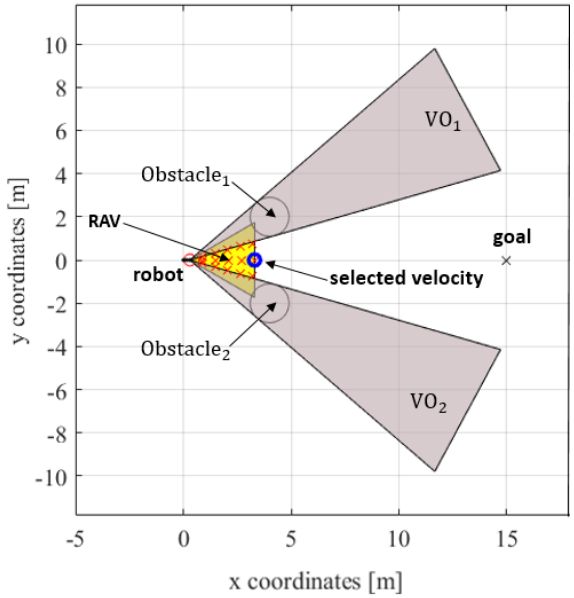

Fig. 4.1 First example: Velocity selection

areas, the robot is the red circle, blue circle means the selected velocity vector and the goal is depicted by black x.

The magnitudes and the changes in the velocities of the obstacles do not influence the calculation process of the uncertainties because there are two static obstacles. So in that case, only the distances between the robot and the obstacles have an impact on the calculation. Always the velocity vector between the two obstacles will be selected, so the distances between the agent and the two obstacles will be the same during the whole motion resulting the same uncertainties for both obstacles as it is presented in Figure 4.2 (at each time step, the uncertainties for the obstacles can be seen next to each other).

Using the original APF method that was introduced in Section 2.2, the agent cannot reach the target position. This is because at the beginning of the motion, the APF method results in a velocity vector that causes a motion in the opposite direction from the target position, as can be seen in Figure 4.3 (the values of the constant parameters were $\gamma = 2$ and $\rho = 0.1$). $\mathbf{F}_{\text{ar1}}$ and $\mathbf{F}_{\text{ar2}}$ represent the repelling forces for the different obstacles, and $\mathbf{F}_{\text{arsum}}$ is the summation of the repelling forces (the other notations are the same as those introduced in previous sections). Eventually, the summation of the forces will become a force in the direction of the target position. The agent will then move towards the target position. This sequence is repeated, resulting in an oscillation without reaching the target position.

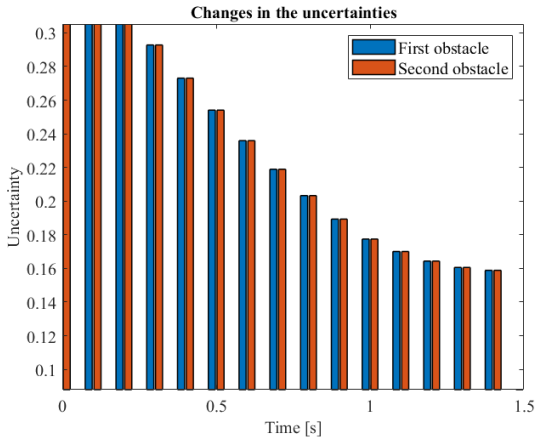

Fig. 4.2 First example: two static obstacles; changing uncertainties during the motion

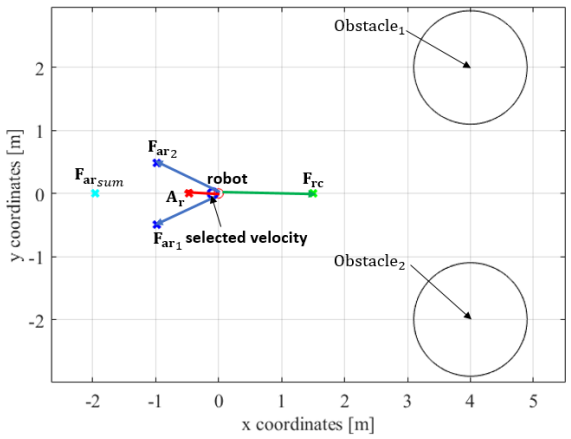

Fig. 4.3 First example: Original APF method

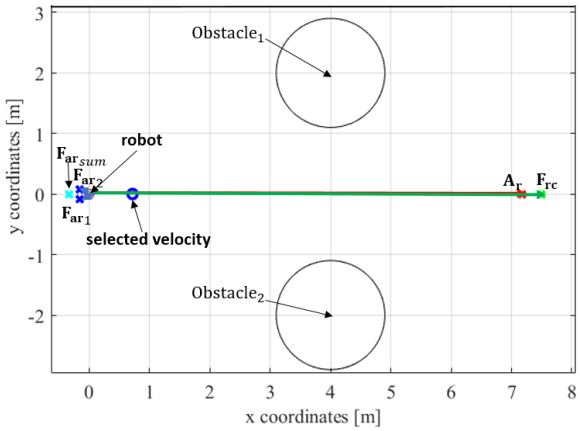

Fig. 4.4 First example: APF method

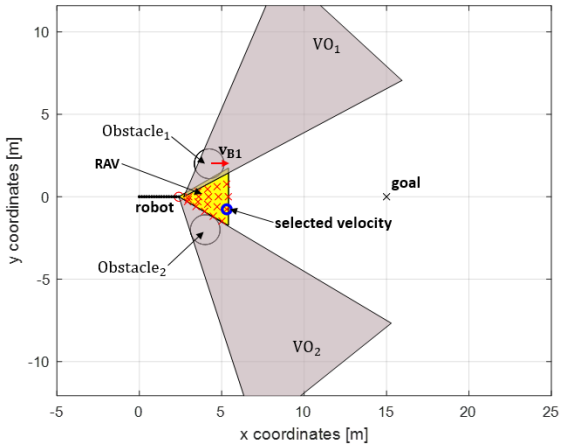

Fig. 4.5 Second example: Velocity selection

One static and one moving obstacles

In this example, the first obstacle is a moving obstacle and the second obstacle is a static obstacle. If the agent is far from the obstacles then it has the opportunity to select the velocity in the line to the goal position. After that, if it goes close to the obstacles it selects a velocity vector that results in a maneuver near to the static obstacle because its uncertainty is smaller. The results of the velocity selection in this case is depicted in Figure 4.5. The path of the robot is presented as a black line.

In that example, the uncertainties of the obstacles are not the same as in the previous case, the static obstacle has a smaller uncertainty during the whole motion as it is presented in Figure 4.6. It can be detected that in the first step there is not a huge difference between the uncertainties of the obstacles. It is generated because the moving obstacle has a small magnitude of the velocity and the distances between the obstacles and the robot are the same at the first step.

This example was also simulated with the APF method. Because the first obstacle has a nonzero velocity vector, this obstacle has a higher uncertainty when using the motion-planning algorithm. However, because the magnitude of the velocity vector is not a large value compared with the summation of the forces, the agent executes its motion almost directly in line with the target position, as presented in Figure 4.7. So, in this case, there is a difference between the result of the extended VO method and that of the APF method, but both of them can result in a collision-free motion between the agent and the environment, and using both methods, the agent can reach the target position. The uncertainties of the obstacles can also be calculated during the motion of the agent with the APF method, although the result will be slightly different from the result of the extended VO method. The changes of the uncertainty parameters can be seen in Figure 4.8.

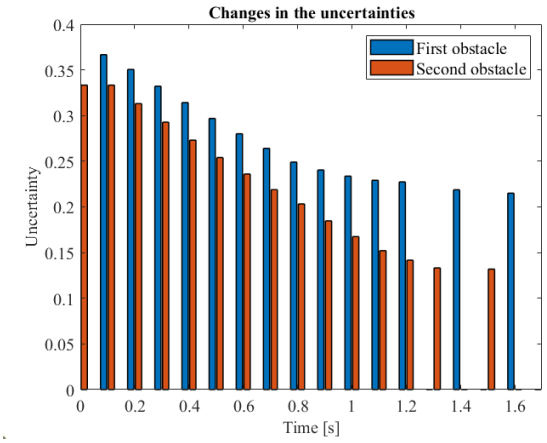

Fig. 4.6 Second example: the first obstacle is a moving obstacle, the second obstacle is a static obstacle; changing uncertainties during the motion

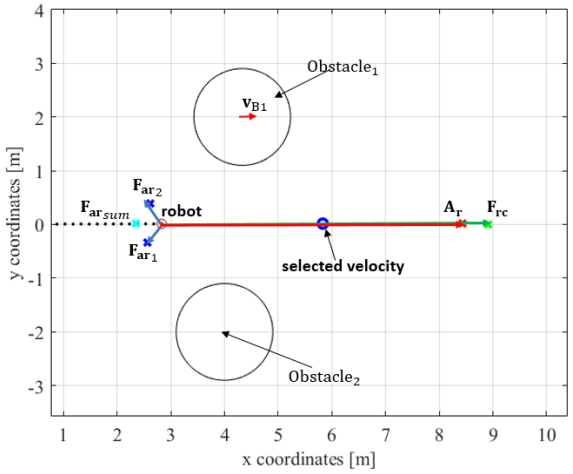

Fig. 4.7 Second example: Velocity selection based on the APF method.

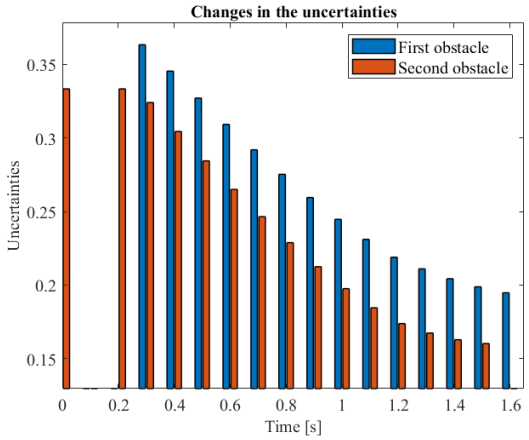

Fig. 4.8 Second example: One moving (first) and one static (second) obstacle; changing uncertainties during motion using the extended APF method.

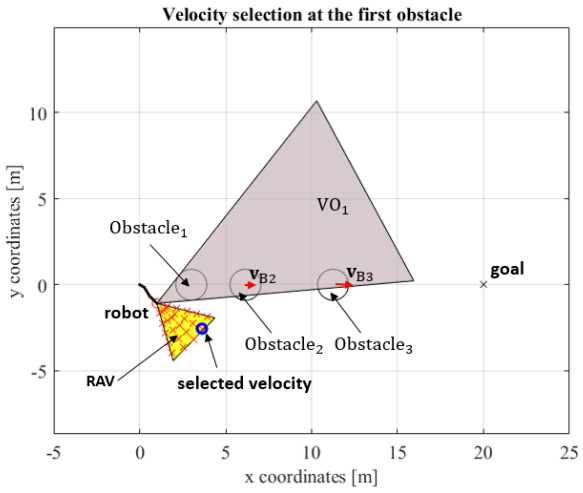

Fig. 4.9 Third example: Velocity selection at the first obstacle

Three obstacles in front of each other

In the following example, there are three obstacles in front of each other with different velocities (the first obstacle is a static obstacle, and the others have even higher velocity vectors). Figure 4.9 represents the velocity selection at the first obstacle and Figure 4.10 shows the velocity selection at the second obstacle. It can be detected that a further velocity vector will be selected at the second obstacle, hence it has a higher velocity vector.

The bigger the velocity of the obstacle is, the higher the uncertainty is for the obstacle as it is represented in Figure 4.11. After passing the obstacle, the uncertainty is reduced. This figure shows that not all the obstacles need to be considered throughout the motion, only those that influence the motion of the robot and which have fulfilled the precheck algorithm that was introduced in Section 3.1. The calculation of the reliability is not executed for those obstacles that should be not considered after the precheck method.

The β_h parameter plays a significant role in the cost function at the aspect of the target reaching strategy. In the previous examples, the value of β_h was 0.3. If this parameter has a bigger value, it has a higher impact on the motion than the uncertainties of the obstacles, as presented in Figure 4.12. In that case ($\beta_h = 0.6$), the agent executes the motion as close to the obstacle as the collision-free motion-planning algorithm allows.

The values of the parameters depend on the usage of the algorithm; different parameter values generate different results using the collision-free motion-planning algorithm. However, it is not possible to find a solution that takes into account every

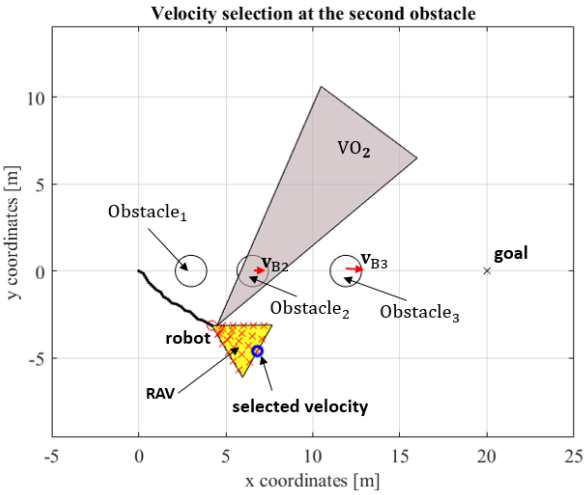

Fig. 4.10 Third example: Velocity selection at the second obstacle

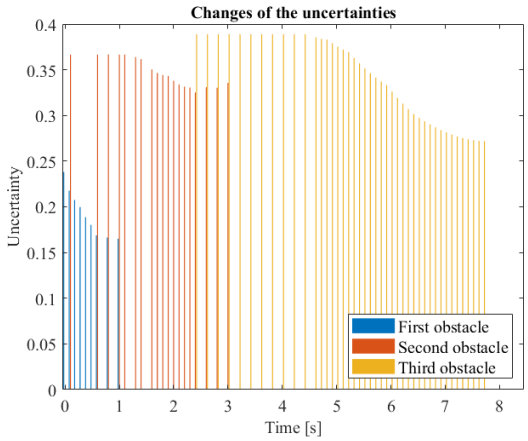

Fig. 4.11 Third example: three obstacles in front of each other with different velocities; changing uncertainties during the motion

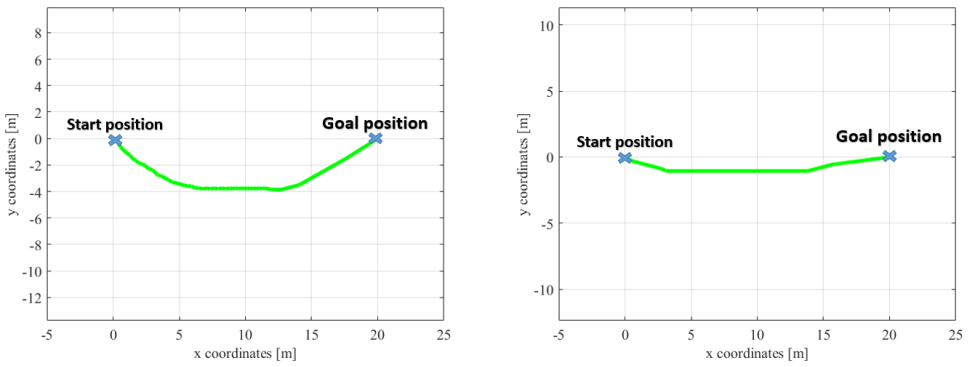

Fig. 4.12 The resulted paths of the motion of the robot with different heading parameters, in the first example $\beta_h = 0.3$, in the second example $\beta_h = 0.6$

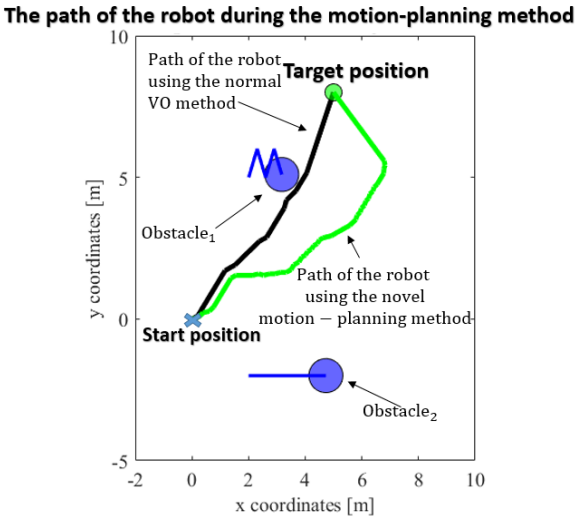

Fig. 4.13 Final paths of the robot using the normal VO method and the novel motion-planning algorithm.

aspect of the motion-planning problem. A sub-optimal solution must therefore always be calculated.

Using the APF method, only the first obstacle is considered in motion planning due to the precheck algorithm. The algorithm directs the agent towards the first obstacle, leading to a collision. The APF method doesn't ensure a path to the target.

Standard VO method and the novel motion-planning method

The introduced novel motion-planning algorithm was also compared with the original VO method because the basic concept of the motion-planning algorithm is based on this algorithm. Figure 4.13 presents the final paths of the robot using the normal VO method and the novel motion-planning algorithm. The comparison used the example of two moving obstacles in the robot's workspace. One of them has a changing velocity vector that results in a higher uncertainty in the motion of the robot. It can be seen that using the standard VO method, which provides the fastest target-reaching concept, the agent executes a tangential motion next to the first obstacle. However, if the uncertainties in the measurement data are also considered, the target-reaching method will be solved, generating a path that is relatively far from the first obstacle (with changing velocities).

4.2 Obstacle estimation with Kalman filter and Particle filter method

The previous motion planning algorithms used the position and velocity vectors of the obstacles. Now, different state perception methods are introduced that can perceive this information using an onboard LiDAR sensor.

4.2.1 Kalman filter method

In this section, the Kalman filter algorithm [95–99] is presented in a structured manner. The Kalman filter was introduced to estimate the state vector of a linear system considering noise as well. The methodology of the algorithm is divided into two key components: the time update and measurement update processes. These two components are crucial in achieving the desired state estimation for the system under consideration.

The system equation can be defined as:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \xi_k \tag{4.9}$$

where k is the actual discrete time step, $\mathbf{x}_k$ is the actual state, $\mathbf{A}$ is a square matrix, $\mathbf{B}$ is a column vector in the case, when the system has one input and $\mathbf{u}_{k-1}$ is the control input, ξ_k is the system noise.

The output can be calculated as:

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + z_k \tag{4.10}$$

where $\mathbf{y}_k$ is the output vector, $\mathbf{C}$ is a row vector or a matrix and z_k is the measurement noise.

The time update can be defined with the following equations:

$$\bar{\mathbf{x}}_k = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} \tag{4.11}$$

where $\bar{\mathbf{x}}_k$ is the a priori state estimation, $\hat{\mathbf{x}}_{k-1}$ is the previous state prediction using the Kalman filter.

After that, the error covariance $\mathbf{M}_k$ can be calculated.

$$\mathbf{M}_k = \mathbf{A}\Sigma_{k-1}\mathbf{A}^\intercal + \mathbf{R}_v \tag{4.12}$$

where Σ_{k-1} is the covariance matrix of the prior estimate and $\mathbf{R}_v$ is the covariance matrix of the process noise ξ_k.

The Kalman gain $\mathbf{K}_k$ can be calculated, which minimizes the a posteriori error covariance Σ_k.

$$\mathbf{K}_k = \mathbf{M}_k\mathbf{C}^\intercal(\mathbf{C}\mathbf{M}_k\mathbf{C}^\intercal + \mathbf{R}_z)^{-1} \tag{4.13}$$

where $\mathbf{R}_z$ is the covariance matrix of the measurement noise z_k.

The maximum speed and hence displacement of the obstacles was assumed, and this data was used to determine the covariance matrix of the process noise ($\mathbf{R}_v$), and the LiDAR data sheet was used to determine the measurement noise covariance matrix ($\mathbf{R}_z$).

Receiving a measurement from the sensor ($\mathbf{y}_k$), the measurement update can be calculated for the estimated state:

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\bar{\mathbf{x}}_k) \tag{4.14}$$

The a posteriori error covariance matrix is:

$$\Sigma_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{M}_k \tag{4.15}$$

It is important to note that before executing the Kalman filter algorithm, it must be initialized. The system matrices, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ must be known. Additionally, the values of $\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{R}_z, \mathbf{R}_v$ must be added.

This method was used primarily for state estimation of the agent but I will use it for perception of the obstacles occurring in the workspace in Section 4.2.3.

4.2.2 Particle filter method

In this Section, we provide a detailed description of the steps involved in implementing the Particle filter algorithm. The Particle filter is a method for approximating the posterior distribution of a system using a discrete density. One of the key advantages of the Particle filter over other recursive Bayesian filtering methods is its ability to handle nonlinear dynamic models, in addition to linear ones. Additionally, the Particle filter can be applied to systems with non-Gaussian noise.

In the Particle filter algorithm, a weighted set of points ($\mathbf{S}_k$) is utilized to approximate the posterior distribution. The set comprises of pairs of $< \mathbf{x}_k^{(i)}, w_k^{(i)} >$, where $\mathbf{x}_k^{(i)}$ represents a possible state of the i^{th} particle at time k, and $w_k^{(i)}$ denotes the likelihood (weight) of that state. The number of particles, represented by N, is not fixed and may

change during the iteration. The vector of weights, denoted as $\mathbf{w}_k$, is constrained by the requirement that the sum of its elements is equal to 1 ($\sum_{i=1}^{N} w_k^{(i)} = 1$).

Prior to implementing the Particle filter algorithm, it is necessary to initialize the filter. This includes defining the state transition function, the resampling strategy, and the initial number of particles, denoted as N. The state transition function is mathematically represented in equation (4.16).

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \xi_k \tag{4.16}$$

where f is a linear or nonlinear function, $\mathbf{u}_k$ means the actual control input and ξ_k is the system noise.

In the initialization step of the Particle filter algorithm, a random set of points, denoted as $\mathbf{x}_1^{(i)}$, is generated via sampling from the apriori distribution $P_{\mathbf{x}0}$. The initial weight assigned to each particle is $\mathrm{w}_1^{(i)}$, where the index 1 indicates that the current weights are calculated at time step $k = 1$ using information from the prior iteration. In the absence of available measurement data at the beginning of the algorithm, the volume of the weights is set to $1/N$. The steps of the Particle filter algorithm, as outlined in [115], involve the following:

1. Step of Measurement update

   The process of updating the weights of the particles in the Particle filter algorithm is critical in ensuring accurate state estimation. This is accomplished by incorporating sensor measurements ($\mathbf{z}_k$) into the weight update equation for each particle.

   $$\mathrm{w}_k^{(i)} = \frac{\mathrm{w}_{k-1}^{(i)}\, P(\mathbf{z}_k|\mathbf{x}_k^{(i)})}{\sum_{j=1}^{N} \mathrm{w}_{k-1}^{(j)}\, P(\mathbf{z}_k|\mathbf{x}_k^{(j)})} \tag{4.17}$$

   where $P(\mathbf{z}_k|\mathbf{x}_k^{(j)})$ means what is the probability that the measurement is $\mathbf{z}_k$ if the system is in $\mathbf{x}_k^{(j)}$.

2. Estimation

   The approximated state can be calculated:

   $$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} \mathrm{w}_k^{(i)}\, \mathbf{x}_k^{(i)} \tag{4.18}$$

3. Resampling

Usually, N samples will be selected from the set of the particles with replacement, considering the weights of the particles. Several resampling methods can be used that were investigated in my previous research [C6].

4. Time update

   In this step, prediction can be defined as:

$$\mathbf{x}_{k+1}^{(i)} = \mathbf{A}\mathbf{x}_{k}^{(i)} + \mathbf{B}\mathbf{u}_k + \gamma \tag{4.19}$$

   when the system is a linear model. If it is not, then (4.16) can be used in the time update step. $\mathrm{w}_{k+1}^{(i)}$ can be calculated using (4.17) with the new sample of particles after the resampling step. γ means the system noise in the equation.

4.2.3 Perception method of an Obstacle with the Kalman filter and Particle filter

LiDAR sensor simulation

This study introduces the utilization of a 2D LiDAR sensor in a simulated environment. The sensor has a maximum range of D_L (in my research, I used an RpLiDAR2 where the value is 12 meters) and a resolution range of R_L which is between $[0^\circ, 1^\circ]$ (in my case it is 0.5°). The measurement noise is also simulated in the introduced environment. Figure 4.14 depicts the LiDAR measurement data obtained. The position of obstacles in the workspace can be determined through segmentation of the measurement data and identification of the angles associated with said obstacles. It is assumed that all obstacles present in the workspace possess a disk-like shape (the VO algorithm uses disk-shaped obstacles as well). The position of the center point of the obstacle, represented by $(z_{p_x},\ z_{p_y})$, can then be calculated using the Least Square (LS) method (that is presented in Appendix A), which works also as a segmentation method because the obstacles are separated from the environment. The methodology of obstacle measurement employed in this study is previously detailed in my prior work [C7].

Perception method with the Kalman filter

Two different methods can be introduced in the Kalman filter solution that will be compared later:

- The perceived state is the position of the obstacle (the velocity vector is calculated from the position) which is the basic Kalman filter perception method $\hat{\mathbf{x}}_{k_2} = [\hat{p}_x,\ \hat{p}_y]^T$

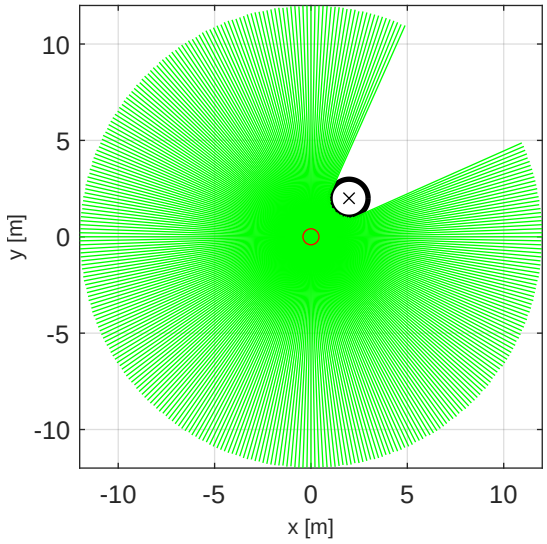

Fig. 4.14 Measurement data using the LiDAR sensor. The agent is in the origon position (red circle). The onboard sensor distance data can be seen with green lines. There is one obstacle in the workspace, represented by a black circle.

- The perceived state includes the position and the velocity vector of the obstacle which is the extended Kalman filter perception method $\hat{\mathbf{x}}_{k_4} = [\hat{p}_x,\ \hat{p}_y,\ \hat{v}_x,\ \hat{v}_y]^T$

In the first aspect, the time update can be formulated with the following equation:

$$\overline{\mathbf{x}}_{k_2} = \hat{\mathbf{x}}_{k_2-1} + (\hat{\mathbf{x}}_{k_2-1} - \hat{\mathbf{x}}_{k_2-2}) \tag{4.20}$$

where it is assumed that the velocity vector is constant between the time steps. It is important to note that in the aforementioned equation, the control input is not present. This is due to the fact that during the estimation of the obstacle, it is not possible to control its motion, and thus, the position of the obstacle can only be estimated using sensor data acquired from a LiDAR sensor. The state vector in this scenario comprises the p_x and p_y coordinates of the obstacle.

In addition to the position of the obstacle, the time update step of the state vector can also include the velocity vector. This allows for the definition of the time update as:

$$\overline{\mathbf{x}}_{k_4} = \mathbf{A}_4 \hat{\mathbf{x}}_{k_4-1} \tag{4.21}$$

where the state vector consists p_x, p_y, v_x, v_y of the obstacle and $\mathbf{A}_4$ is defined in (4.22) where T is the sampling time.

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

The equations of (4.12), (4.13), (4.15), (4.14) can be used in the same format as they were described in Section 4.2.1. The dimension of the covariance matrices have a size of 2×2 in the first case and 4×4 in the second case. Using the LiDAR sensor, the center point of the obstacle can be calculated using the Least Square method-based segmentation of the measurement data of the LiDAR sensor.

Perception method with the Particle filter

The Particle filter methodology is described here, which utilizes a weighted set of particles to represent the possible states of the system at a given sampling time. These states can be represented by the **particles** matrix, with dimensions of 2 rows and N columns, and the vector of weights, denoted by $\mathbf{w}$. At the Kalman filter, we can use $\overline{\mathbf{x}}_{k_2}$ and $\overline{\mathbf{x}}_{k_4}$, now, because the LiDAR sensor provides only position data, $\overline{\mathbf{x}}_{k_2}$ is used for position perception. The prediction of the states of the obstacles is accomplished utilizing measurement data obtained from the LiDAR sensor. The algorithm also accounts for the system and measurement noise. The methodology for executing the algorithm is outlined in Section 4.2.2, with the main steps being iteratively implemented.

Calculation of the weights of the particles

Assuming that the workspace only contains disk-shaped obstacles, we can compute the center point using the LiDAR sensor's measurement data. Here, an estimated x center (z_{p_x}) and estimated y center (z_{p_y}) are derived through a *Least square error* estimation, accounting for every data point measured by the sensor.

Each particle signifies the obstacle's center point (p_x and p_y). The error of the distance measurement noise (deviation) is represented by MN_d. Each particle contains data on the x position ($\hat{z}_{p_x}(i)$) and the y position ($\hat{z}_{p_y}(i)$).

Initially, the x position is taken into account when computing the weights:

$$w_{p_x}^{(i)} = \frac{1}{\sqrt{2\pi\, MN_d}} exp(\frac{-(z_{p_x} - \hat{z}_{p_x}(i))^2}{2MN_d}) \tag{4.23}$$

The x position weights should be normalized:

$$w_{p_x}^{(i)} = \frac{w_{p_x}^{(i)}}{\sum_{j=1}^{N} w_{p_x}^{(j)}} \tag{4.24}$$

In this instance, the total weight of the particles sums up to 1.

Subsequently, we can compute the weights while taking the y position into consideration:

$$w_{p_y}^{(i)} = \frac{1}{\sqrt{2\pi\, MN_d}} exp(\frac{-(z_{p_y} - \hat{z}_{p_y}(i))^2}{2MN_d}) \tag{4.25}$$

Normalization is also required for the y position weights:

$$w_{p_y}^{(i)} = \frac{w_{p_y}^{(i)}}{\sum_{j=1}^{N} w_{p_y}^{(j)}} \tag{4.26}$$

The ultimate weights of the particles can be deduced by using the x and y position weights:

$$w^{(i)} = w_{p_x}^{(i)} * w_{p_y}^{(i)} \tag{4.27}$$

The final weights should also be subjected to normalization:

$$w^{(i)} = \frac{w^{(i)}}{\sum_{j=1}^{N} w^{(j)}} \tag{4.28}$$

Hence here it is a normalization, (4.24) and (4.26) can be ignored as well.

The state perception algorithm

Upon establishing the weights of every particle, the approximated condition of the obstacle, particularly its location, needs to be deduced. There are multiple techniques to ascertain this perceived condition, one such approach is via the calculation of the particles' mean.

$$\hat{S}_k = \frac{\sum_{i=1}^{N} \mathbf{particles}_k^{(i)}}{N} \tag{4.29}$$

where $\hat{S}$ means the perceived state.

Another approach for estimating the perceived state involves incorporating the weights of the particles. In this scenario, equation (4.18) is applied.

$$\hat{S}_k = \sum_{i=1}^{N} \mathbf{particles}_k^{(i)} w_k^{(i)} \tag{4.30}$$

In the perception methodology, I employed equation (4.18) to discern the state of the obstacle. The state estimation includes the position. The velocity estimation of the obstacles is calculated from the estimated positions (because the sensor measurement data is available only for the positions of the obstacles).

Resampling algorithm and state transition method

In this step, the systematic resampling algorithm was employed for particle selection. The different resampling algorithms are presented in my previous research [C7].

The state of the dynamic obstacle is composed of two coordinates (p_x, p_y), indicating its position. Following the perception of the position, the velocity vector (v_x, v_y) can be deduced. The state transition model takes both the perceived position and velocity into account. It can be articulated as presented in equation (4.19).

At every time step, an uncertainty degree (α) can be also calculated which can be useful in the motion planning algorithm (3.12) for the mobile agent:

$$\alpha = \min\left(1, \max(std(\textbf{particles}))\right) \tag{4.31}$$

where std means the standard deviation which is calculated for both the x positions and y positions of the obstacles and the highest value will be selected for the uncertainty degree. The maximum value is saturated to 1. The bigger the value of the uncertainty degree is, the higher the uncertainty of the accuracy of the actual perception is.

4.2.4 Experiments and Results

In this section, a comparison was made between the proposed Kalman filter-based perception methods and the previously introduced Particle filter-based algorithm.

One moving obstacle with constant velocity vector

In this section, the investigation of a moving obstacle is presented. The initial measurement data with the LiDAR sensor can be seen in Figure 4.15, the deviation of measurement noise (MN_d) was set in this example to 0.05 m.

In this example, a single moving obstacle was present within the operational space of the agent, possessing a constant linear velocity vector ($[v_x = 0.25m/s, v_y = 0.1m/s]$). At the Particle filter algorithm, $N = 50000$ particles were used. The absolute error in the perception of the p_x and p_y positions, as represented in the left and right side of Figure 4.16, respectively, are illustrated. The red line illustrates the results obtained from the Particle filter-based solution, the blue line (Kalman filter basic), that provides the perception of $\overline{\mathbf{x}}_{k_2}$) illustrates the absolute error of the Kalman filter-based method

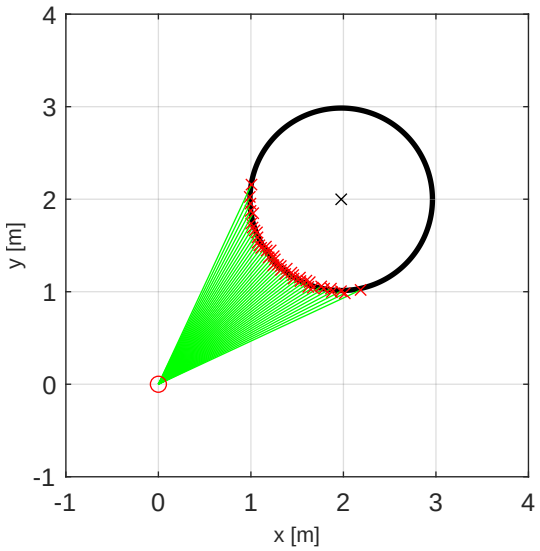

Fig. 4.15 Measurement with LiDAR sensor in simulation environment. The robot is in the origin, the center point of the obstacle is in [2;2] m. The red x-s are the measured points on the disk-shaped obstacle

when only the position is included in the state vector, and the black line (Kalman filter extended, that provides the perception of $\overline{\mathbf{x}}_{k_4}$) represents the results obtained from the Kalman filter-based method when the velocity vector is also included in the state vector. The simulation was conducted for a total of 100 iterations, with a sampling time of 0.1 seconds. The results obtained from the Kalman filter-based solutions and the Particle filter-based method were found to be similar. At the beginning of the simulation, the absolute error in position perception of the Particle filter-based solution was observed to be higher, however, after a few time steps, it was able to attain a more accurate solution in the perception. The average error of the p_x position perception was found to be 0.0228 meters for the Particle filter, 0.0245 meters for the Kalman filter basic, and 0.0218 meters for the Kalman filter extended solutions. The average absolute errors in the y position were found to be 0.0144 meters and 0.0195 meters and 0.0176 m, respectively.

The results of the v_x and v_y velocity perception using the introduced methods are presented in Figure 4.17 in the left and right side, respectively. In contrast to the results of the perception of the position vector, a notable difference can be observed between the Particle filter-based solution and the Kalman filter-based solutions. Specifically, the Particle filter-based solution demonstrates superior performance in terms of velocity perception, with an average absolute error of 0.0435 m/s for v_x and 0.0292 m/s for v_y. In comparison, the Kalman filter-based solutions yield significantly higher absolute errors of 0.1914 m/s and 0.1523 m/s at Kalman filter basic method for v_x and v_y respectively. The Kalman filter extended has an absolute error of 0.0818 m/s and

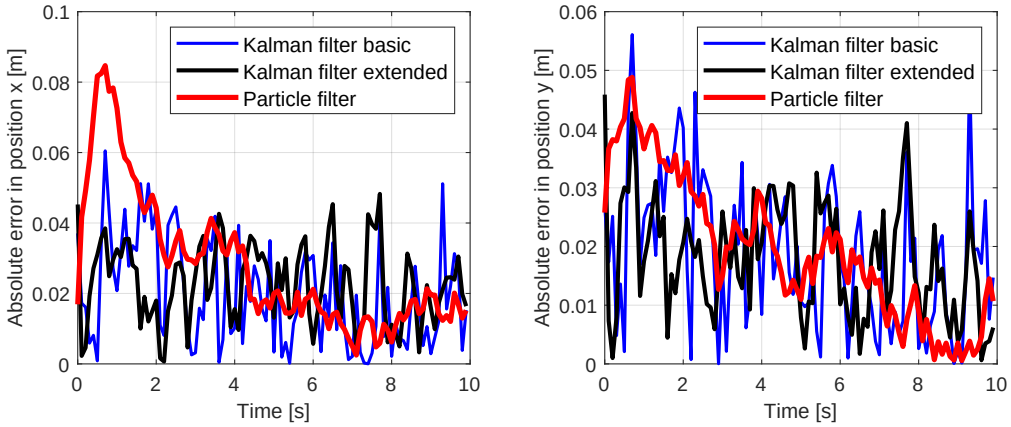

Fig. 4.16 Absolute error in the perception of the x (left) and y (right) position of the obstacle.

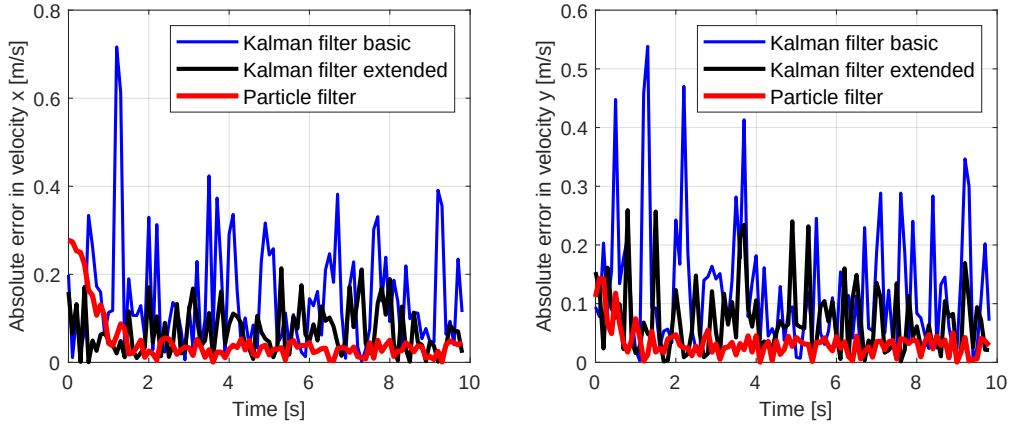

Fig. 4.17 Absolute error in the perception of the x (left) and y (right) velocities of the obstacle.

0.0905 m/s respectively. This suggests that the Particle filter-based method may be a more effective solution for velocity perception in this context.

One moving obstacle with changing velocity vector

In this scenario, a moving obstacle was present in the workspace of the agent, with a variable linear velocity vector. The results of the evaluation of the absolute error in the p_x (left side) and p_y (right side)position perception can be observed in Figure 4.18. The results considering the x-y directions are really similar. As previously observed, it can be inferred that the Particle filter-based solution initially generates a higher average error. However, it is worth noting that when the obstacle changes its velocity vector (at time 10 s), the Particle filter-based perception solution generates a higher average error for a brief period, subsequently reaching a better perception of the position than

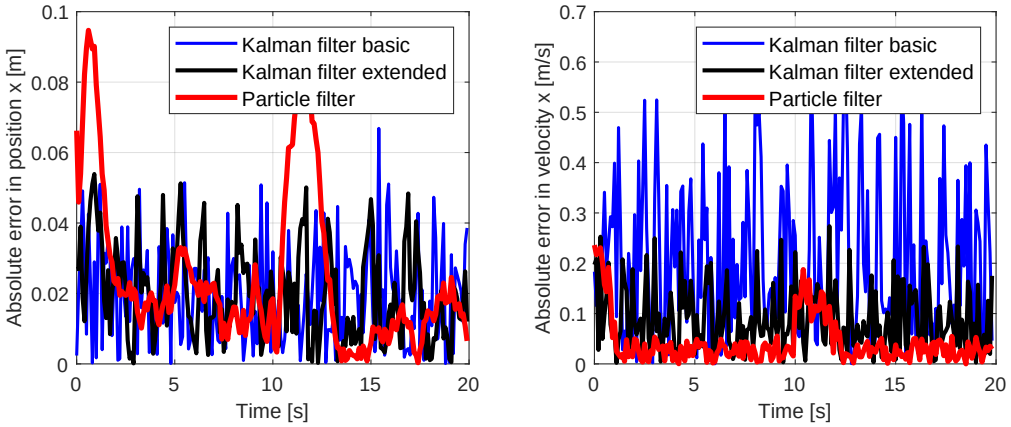

Fig. 4.18 Absolute error in perception of the x postion (left) and x velocity (right) of the obstacle in the second example.

the Kalman filter-based solutions. The Kalman filter-based solutions produce similar average errors throughout the motion of the obstacle.

Figure 4.18 illustrates the results of the average absolute error in the v_x (left side) and v_y (right side) velocity perception obtained using the different methods introduced. As in the previous case, the results of x-y directions are really similar. It is evident that the Particle filter-based solution demonstrates the best performance, with an average error of 0.069 m/s. The Kalman filter extended solution follows closely, with an average error of 0.0819 m/s. Conversely, the worst performance is observed in the Kalman filter basic solution, with an average error of 0.2023 m/s.

Based on the results presented in the previous examples, it can be inferred that, when velocity perception has significant importance, the Particle filter-based solution is a suitable choice as the perception method. The number of particles influences the accuracy of the estimation at the Particle filter (the higher the particle number is, the higher accuracy can be reached). Additionally, it is important to add that the system model is not perfect and the Kalman filter uses a linear model. It can be also a cause for the better performance of the Particle filter.

One static obstacle and one moving obstacle

In this example, there are two obstacles present in the workspace of the agent, one static and one moving obstacle. The perception of multiple obstacles can be approached in a variety of ways. One option is to employ different Particle or Kalman filters for each obstacle present in the agent's workspace. Another solution is to utilize a single Particle filter for the perception of the state of all obstacles. Given the previous results, this example presents only the Particle filter-based solution as the perception of the

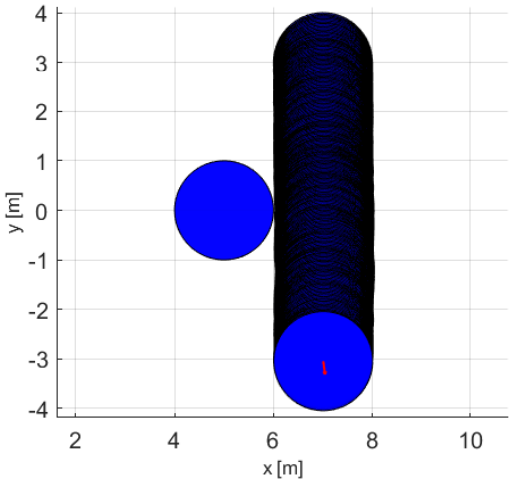

Fig. 4.19 Two obstacles are in the workspace of the agent. The moving obstacle executes its motion behind the static obstacle.

velocity vector is also deemed important. In this case, two Particle filters are used for the perception of the two obstacles.

As depicted in Figure 4.19, the path of the two obstacles in the workspace is represented. The moving obstacle is located behind the static obstacle for a period of time (while it is moving from up to down), during which sensor information from the moving obstacle is not available to the agent. In this scenario, the agent relies on previous estimations to predict the path of the moving obstacle. Once the moving obstacle emerges from the coverage of the static obstacle, the perception can be updated. The uncertainty degree, as introduced in equation (4.31), can be calculated at each time step during the motion of the obstacle.

In Figure 4.20, a comparison is presented between the actual, measured, and estimated positions of the obstacles. It is evident that the position of the static obstacle can be accurately measured and estimated. However, when the moving obstacle is obscured by the static obstacle, the measurement will be significantly incorrect. The Particle filter-based estimation method is capable of utilizing previous estimation results in the absence of new measurement data. As can be seen, the estimation error increases with the duration of the absence of measurement information. Upon receipt of information from the moving obstacle again, the estimation method can promptly predict its state with minimal absolute error.

Figure 4.21 depicts the temporal evolution of the uncertainty degree (α) throughout the motion of the static and moving obstacle. The graph illustrates that the uncertainty degree initially decreases at both obstacles, however, when the moving obstacle is obscured by the static obstacle, there is a significant increase in the uncertainty degree,

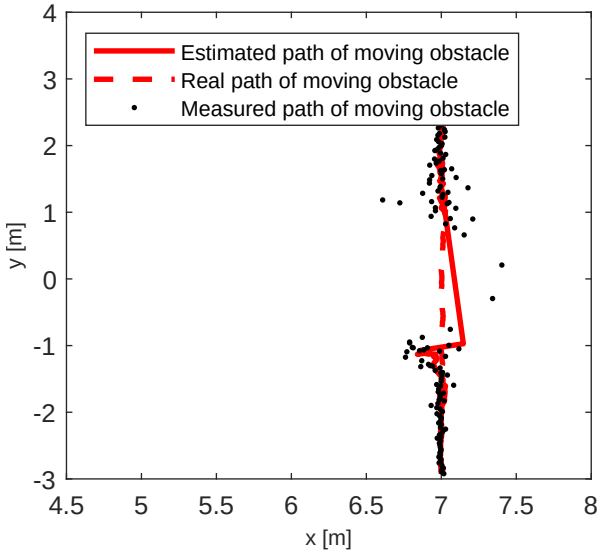

Fig. 4.20 Comparing the real, measured (LiDAR sensor-based), and estimated path of the obstacles

reaching a saturation value of 1. At this point, the absence of sensor information from the moving obstacle results in the dispersion of particles. In such instances, it is imperative for the robot to execute safe evasive maneuvers to avoid collisions. Considering the uncertainty degree of the static obstacle, it is constant after a while because there is always available sensor information in every time step that can be received from the LiDAR sensor. In that case, the uncertainty degree of the static obstacle is not 0 because of the measurement noise of the LiDAR sensor. The uncertainty degree can also be utilized in conjunction with cost function-based motion planning algorithms, which will be presented in detail in Section 4.3.

4.3 Particle Filter Velocity Obstacle (PFVO) method

A novel concept of uncertainty estimation, the Particle Filter Velocity Obstacle method was also introduced for the calculation of the uncertainty degree, the Particle Filter Velocity Obstacle method. In this case, the uncertainty degree can be calculated from the Particle filter algorithm, using the standard deviation of the particles and the changes in the velocity vector.

At every time step, an uncertainty degree (α_v) can be also calculated which can be useful in the motion planning algorithm for the mobile agent. First, the change of the velocity must be calculated:

$$\Delta v_i = ||\mathbf{v}_{Bi}(k) - \mathbf{v}_{Bi}(k-1)|| \tag{4.32}$$

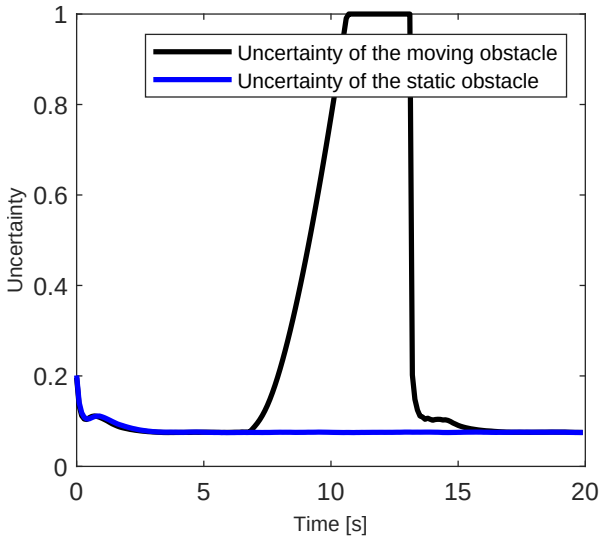

Fig. 4.21 Uncertainty degree in every time step for static and the moving obstacle

where k means the actual time step, $k-1$ is the previous time step, and i represents the i^{th} obstacle in the workspace of the agent.

The saturated deviation of the particles can be determined:

$$\alpha_i = \min\left(1, \max(std(\mathbf{particles}_i))\right) \tag{4.33}$$

Finally, the uncertainty degree (α) can be calculated:

$$\alpha_{v,i} = \max\left(\Delta v_i, \alpha_i\right) \tag{4.34}$$

A larger value for the uncertainty degree indicates a higher degree of uncertainty regarding the precision of the current perception. The uncertainty degree is calculated for every obstacle but only the maximum value from these uncertainties is used in the velocity selection:

$$\alpha_v = \max_i(\alpha_{v,i}), \tag{4.35}$$

which can be used in the objective function later (4.36).

4.3.1 Collision avoidance with Particle Filter Velocity Obstacles method

In the implementation of the Particle Filter Velocity Obstacle approach, the velocity decision for the agent is based on two key elements: speed and safety (uncertainty). Both these factors are simultaneously taken into account by employing a weighted

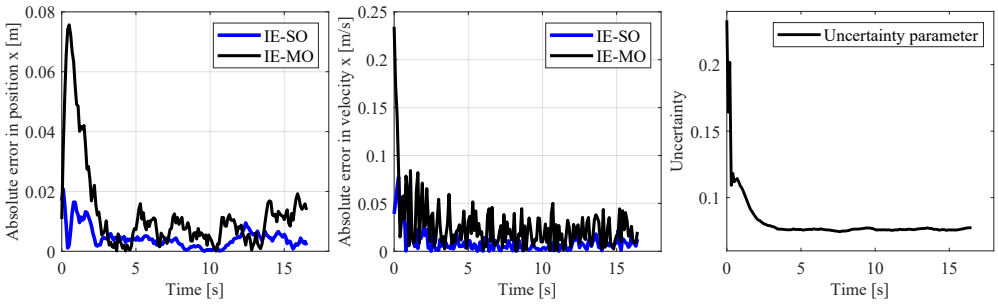

Fig. 4.22 Result of the first example. The left and center figures represent the absolute error in the position and the velocity and the right figure shows the uncertainty degree.

objective function that is changed depending on the present uncertainty degree at the specific sampling time.

Ultimately, the speed and safety components of the fitness function ($f(\mathbf{v}_i)$) are using a weighted mean, in which the parameter α_v denotes the previously calculated uncertainty degree. In scenarios where the uncertainty increases, the significance of the safety component within the objective function must be correspondingly amplified. The primary objective of the algorithm is to identify the velocity vector for the agent that maximizes the value of the objective function. The objective function can be used similarly as it was introduced in (3.20).

$$f(\mathbf{v}_i) = \begin{cases} \alpha_v \, SA(\mathbf{v}_i) + (1-\alpha_v) \, GO(\mathbf{v}_i) & \text{if } \mathbf{v}_i \in RAV \\ 0 & \text{otherwise} \end{cases} \tag{4.36}$$

4.3.2 Experiments and Results

In this section, the simulation and experimental results are presented using the Particle Filter Velocity Obstacle method.

Simulation results

All of the presented examples were compared with the fastest solution (VOTG method), which was presented in Section 2.1, and with one of our previously introduced motion planning algorithms (SVO method) (Section 3.1), where there is a predefined safety parameter.

First example: the obstacles are observable

In the initial illustrative case, the agent's workspace includes both stationary and dynamic obstacles (the perception was made by using different particle filters for different obstacles). Sensor information is sent to the robot at each sampling time.

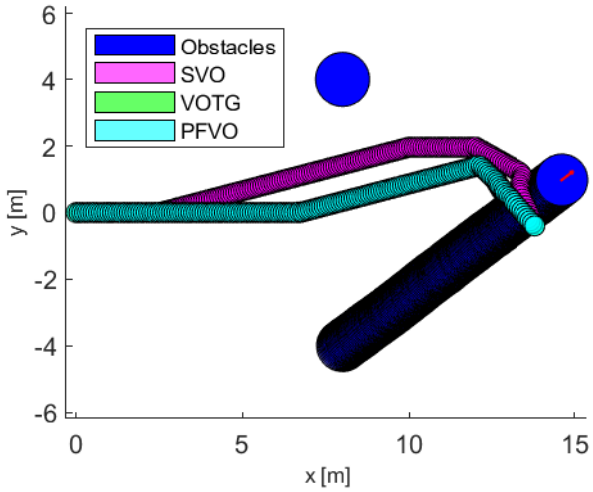

Fig. 4.23 Path of the mobile agent to the goal in the first example at the PFVO method

Figure 4.22 presents the absolute error pertaining to position and velocities for both categories of obstacles, in addition to the fluctuation of the uncertainty degree throughout the movement. In the referenced figures, 'IE-SO' symbolizes Inertial Estimation for stationary obstacles, while 'IE-MO' signifies Inertial Estimation for the moving obstacle. It is observed that the estimation accuracy concerning obstacle positioning (0.02 m) and velocity estimation (0.04 m/s) using N=10000 particles; nonetheless, velocity estimation maintains a substantial degree of precision and usability. The estimation accuracy for stationary obstacles exhibits a marginal superiority over that for the dynamic obstacle.

In Figure 4.23, the trajectory of the autonomous agent towards the target position is depicted. Under these conditions, the trajectory outcomes derived from both the VOTG and Particle Filter Velocity Obstacle (PFVO) methodologies align closely, attributable to the minimal uncertainty degree, which in turn, is a consequence of the consistent availability of measurement data at each sampling time. Conversely, the Safety Velocity Obstacle (SVO) approach navigates towards the target along a safer trajectory, a direct result of an elevated safety parameter.

Second example: the velocity of the moving obstacle is changed

In the second experiment, I considered a scenario with two moving obstacles. One obstacle maintains a constant velocity vector, while the other changes the velocity during its movement in some sampling time. The variability in the uncertainty degree, as presented in Figure 4.24, is noteworthy. The parameter experiences an increase

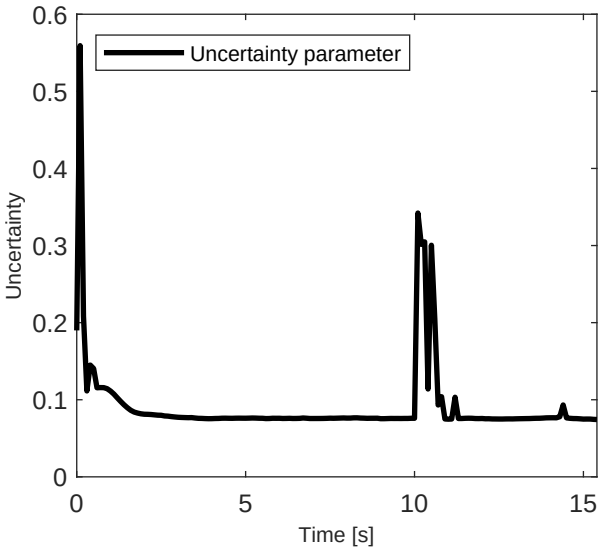

Fig. 4.24 The changes of the uncertainty degree in the second example at the PFVO method

when the obstacle alters its velocity vector. However, once the agent assimilates this new information over time, the uncertainty degree subsequently decreases.

One obstacle starts its motion in the position of [4; -3] and has a velocity vector of [0.8; 0] m/s until 10 s when it changes the velocity vector to the opposite direction and continues the motion with the velocity of [-2; 0] m/s. The agent's path to the goal position is illustrated in Figure 4.25 Analyzing the agent's paths under different strategies reveals that even in the face of escalating uncertainty, the PFVO and VOTG strategies results in quite similar path. This leads to the fastest goal attainment as none of the moving obstacles impact the robot's trajectory. Conversely, the SVO strategy (with parameter $\alpha = 0.5$), due to its higher constant safety parameter, results in a more evasive path, offering a wider maneuver from the obstacle that exhibits a change in velocity during its motion.

Third example: disappearing obstacle

In the third experiment, the environment contains a stationary obstacle and a dynamic obstacle that, during its trajectory, becomes covered by the former. As the dynamic agent disappears from detection, the autonomous agent is ignored for any sensory input regarding it. Consequently, the agent has to estimate the obstacle's position and velocity vector solely based on historical data.

In Figure 4.26, the left segment illustrates the absolute error in estimating the position within this scenario (in this scenario, the errors are presented considering the x coordinates but the result is similar in y coordinates). 'IE-SO' and 'IE-MO' retain

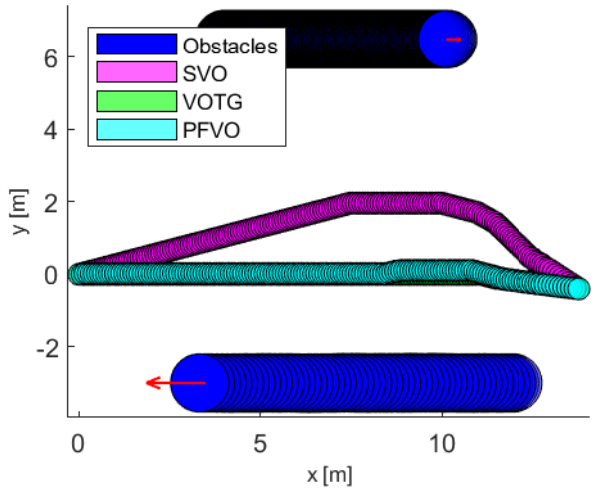

Fig. 4.25 Result of the motion of the agent in the second example at the PFVO method

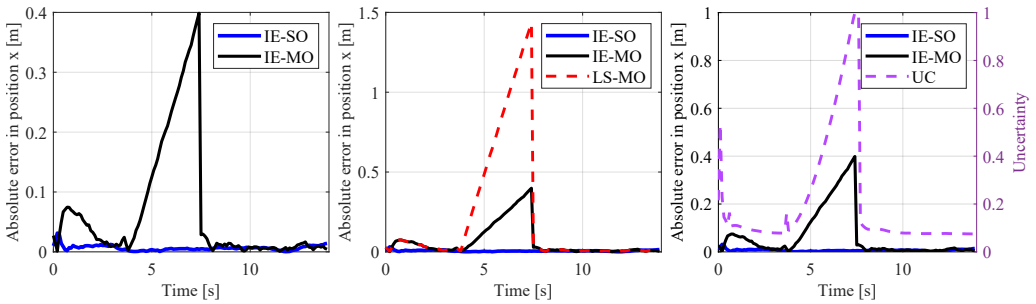

Fig. 4.26 Result of the third example. The left and the center figures represent the absolute error of the positions and the right figure shows the absolute errors and uncertainty degree at the PFVO method

their meanings as outlined in the preceding examples (the maximum error in position estimation is 0.4 m). It becomes evident that as long as the LiDAR sensor provides the agent with sensor measurement data, the error in estimation decreases. However, when the moving obstacle is covered by the static one, a rise in the estimation error for the position is observable, and this continues to increase until the agent can reestablish sensor information.

This escalating error can largely be connected with the missing phase of the resampling method in the Particle Filter Velocity Obstacle (PFVO) algorithm. In such instances, the particles disperse, influencing the estimation, and the algorithm can only rely on the data from the latest perception. Despite the growing estimation error during the period of sensor information unavailability, it is noteworthy that the error remains comparatively smaller than the scenario where the obstacle is assumed to cease motion once it becomes hidden behind the static obstacle. This is illustrated in the center segment of Figure 4.26., where 'LS-MO' denotes the moving obstacle as last seen, implying the case when the obstacle halts upon disappearing.

The uncertainty degree (UC) is depicted alongside the absolute error in the right segment of Figure 4.26. A clear correlation is observed between the uncertainty degree and the estimation error. The uncertainty degree starts its increase when the sensor information becomes unavailable, and as depicted, it reaches its peak value (1) during this period. It begins to decline as the sensor information is restored.

Figure 4.27 visually represents the temporal transformation in the spread of particles. At the beginning of the motion, the particles are evenly distributed (using uniform distribution) across the entire workspace, as depicted in the figure's left segment. The particles are symbolized by red x-marks, while blue x-marks indicate the resampled particles. The black circle marks the estimated current center point of the obstacle.

As the agent begins to receive sensor information regarding the obstacle, a contraction in the particle distribution becomes apparent, as illustrated in the figure's center segment. However, upon the obstacle's disappearance, the particle distribution begins to expand once more, an effect clearly observable in the figure's right segment. During this period, owing to the absence of LiDAR measurements, resampled particles are non-existent.

Figure 4.28 illustrates the spatial relationship between the agent and obstacles in terms of distance. The VOTG strategy results in the fastest solution as it always chooses the quickest velocity vector, despite the potential risks of collision this may pose for the agent during motion. As evidenced by the figure, this strategy generates a trajectory that reaches the obstacles tangentially (where R_r represents the radius of the obstacles), reducing the distance to null. Notably, however, the uncertainty inherent in the sensor measurements makes this tangential motion result in a collision in almost

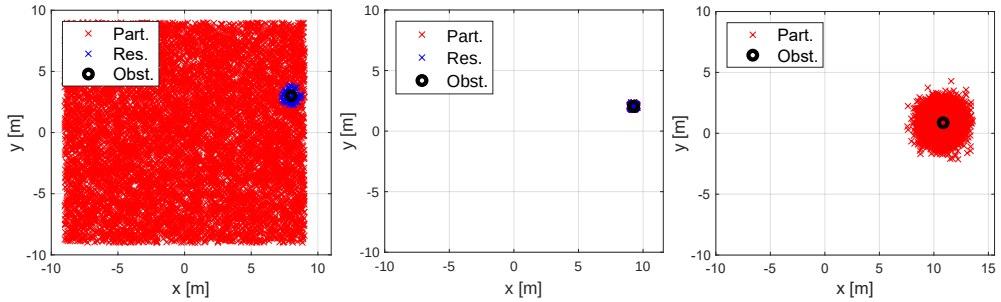

Fig. 4.27 Changes in the distribution of the particles during the motion of the agent at the PFVO method

all cases, thereby undermining its robustness as a solution for collision-free motion planning.

In the case of the Particle Filter Velocity Obstacle (PFVO) strategy, the agent-obstacle distance decreases quickly as long as sensor information is available. Yet, during periods of data unavailability (around the 6-second mark), the expanding uncertainty degree prompts a more cautious trajectory that distances the agent from the nearest obstacle. Once sensor data is reacquired (around the 9-second mark), the agent quickly converges on the goal, achieving it slightly faster than the Safety Velocity Obstacle (SVO) strategy.

The SVO strategy, on the other hand, consistently prioritizes safety during motion planning due to its constant safety parameter ($\alpha = 0.5$). Consequently, it formulates a smoothly safe path, unaffected by the change of sensor data. However, it requires the most time and traverses the longest path to the goal.

Figure 4.29 presents the resultant paths for each strategy.

Table 4.1 represents the relationship between the number of particles, corresponding running times, and the average position error that occurred during execution. It is noteworthy that an increase in particle count corresponds to an increase in running time and a decrease in average error. Notably, with particle numbers 100 and 500, the average error seems excessively high (1.25m and 0.45m respectively). Beyond the particle count of 100000, the running time exceeds 0.1s (thereby transcending the boundary for real-time feasibility). An appropriate trade-off between running time and the average error is achieved by selecting a particle number within the range of 1000 to 50000. In my presented scenarios, I adopt a particle count of 10000, an approach that effectively reduces the average error while maintaining a fast computational process.

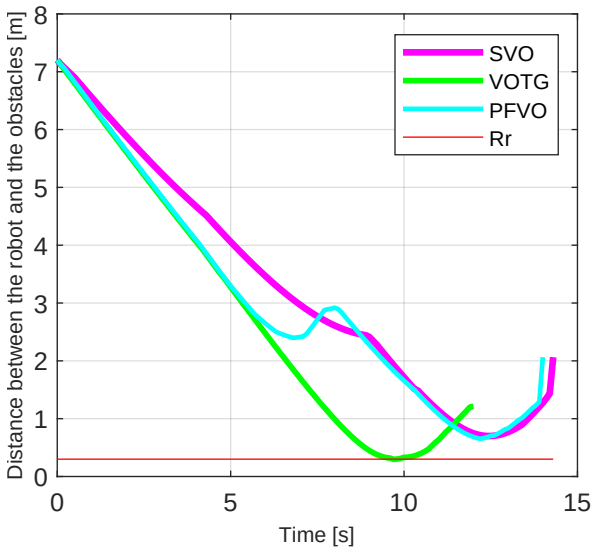

Fig. 4.28 Minimum distances between the agent and the obstacles considering the different strategies at the PFVO method

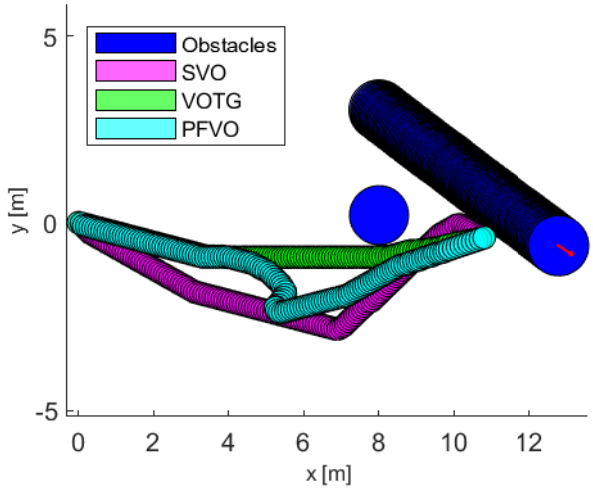

Fig. 4.29 Result of the motion of the agent in the third example at the PFVO method

Table 4.1 Running times of the PFVO algorithm considering the number of the particles

Number of Particles (N)	Running time [s]	Average error in position [m]
100	0.006	1.25
500	0.0077	0.45
1000	0.0078	0.0503
5000	0.0135	0.0385
10000	0.0212	0.0294
50000	0.0647	0.0209
100000	0.1642	0.0119

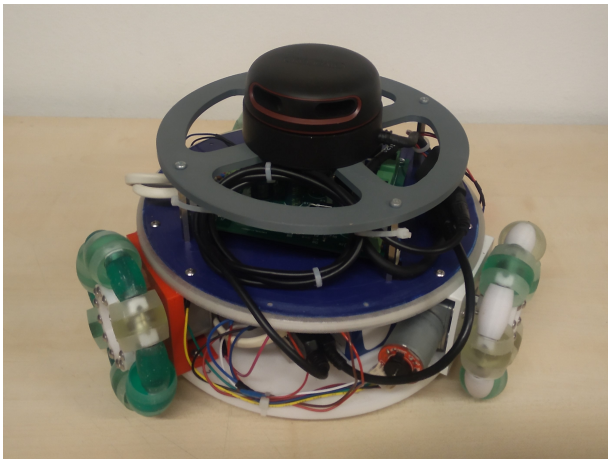

Fig. 4.30 Omnidirectional mobile robot

Experimental result

An omnidirectional mobile robot equipped with a LiDAR was built for the demonstration of the introduced algorithm that can be seen in Figure 4.30. The setup of the agent is presented in Appendix B.

The Particle Filter Velocity Obstacle (PFVO) algorithm was subjected to real-world testing using the introduced robot. As an obstacle within the robot's workspace, a differential-driven robot was employed. This obstacle was programmed to follow a black line on the ground. An extension, colloquially termed a 'hat,' was added to the differential-driven robot to ensure the LiDAR sensor's laser measurements were reflected appropriately; initially, the robot's height was insufficient for this purpose. This modification enabled the agent to consistently obtain information about the obstacle at every time step, thus facilitating accurate estimation of the obstacle's position and velocity vector and ensuring collision-free goal-reaching. To distinguish

Fig. 4.31 Real test at PFVO algorithm

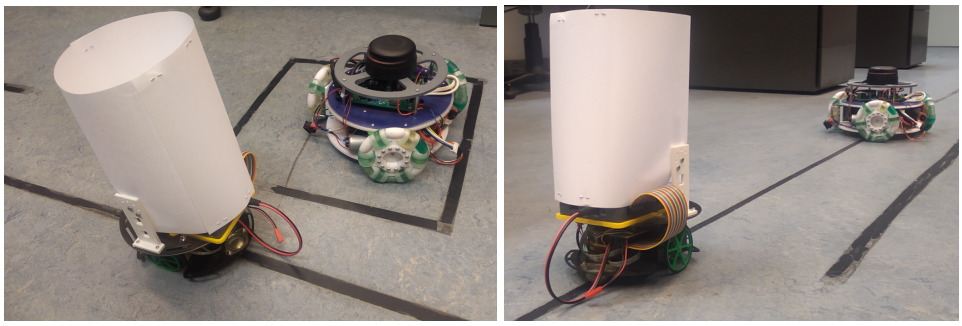

Fig. 4.32 Collision avoidance maneuver at PFVO algorithm

the obstacle from the environment, the segmentation method previously described was used, which effectively differentiates the obstacle from its surroundings.

Figure 4.31. illustrates both the obstacle and the agent's initial position in the workspace. The agent can execute the collision-free motion planning close to the obstacle while it has continuous sensor measurement data reaching the target position as can be seen in Figure 4.32.

4.4 Discussion

Throughout the course of my research, I have developed and compared various perception methods that utilize LiDAR measurement data for estimating the position and velocity vectors of obstacles within the agent's workspace. The Particle filter-based method, in particular, demonstrated superior performance in velocity perception compared to the Kalman filter-based method. I also proposed a method for calculating an uncertainty degree based on current sensor information, which could be integrated into motion planning tasks for mobile robots.

I introduced novel motion-planning method called the Uncertainty Velocity Obstacle method, based on the principles of Velocity Obstacles (VO). Later on, I compared it with Artificial Potential Fields (APF) algorithm. By calculating the changing uncertainties of the obstacles, which depend on the magnitudes of the velocity vectors of the obstacles, the distances between the obstacles and the robot, and the changes in

the obstacles' velocities, the mobile robot was able to execute collision-free motion planning. The VO-based method, in particular, was able to guarantee target-reaching solutions in situations where the APF method fell short.

Furthermore, I successfully introduced a novel method, the Particle Filter Velocity Obstacle mehtod. This method has proven its capability to accurately perceive the position and velocity vectors of obstacles within the agent's workspace. The calculated uncertainty degree, derived from the distribution of particles and changes in velocity vectors, has proven effective when applied in an objective-function-based velocity selection method. Testing through various simulations and experiments has validated the robustness and efficiency of the proposed algorithm.

In summary, my research has made a contribution in the domain of state estimation algorithms and collision-free motion planning for mobile robots in dynamic environments. I integrated the Particle filter-based perception method into motion planning tasks for mobile robots, and I introduced uncertainty degree in a cost-function-based velocity selection method.

Thesis group II.

I introduced a novel navigation algorithm capable of perceiving both the position and velocity vectors of obstacles in the workspace by utilizing LiDAR measurement data. This approach further calculates a changing uncertainty degree, providing an effective solution for collision avoidance challenges. The introduced method can reach safer in motion planning than the APF or VO algorithms considering the collision-free target-reaching task.

Thesis II.1

I developed the Uncertainty Velocity Obstacle method, a novel approach for addressing path planning and collision avoidance tasks in dynamic environments. This method introduces a changing uncertainty degree into the cost-function-based optimization, taking into account alterations in the relative position and velocity vectors of obstacles. The introduced method showed safer path planning in the motion planning task than the APF and VO methods.

Thesis II.2

I introduced and analyzed novel state perception methodologies for obstacles utilizing both Kalman and Particle filters. These techniques are adept at discerning the position

and velocity vectors of obstacles within the robot's workspace. Based on the results, the Particle filter method can generate better state perception accuracy for the obstacles than the Kalman filter method.

Thesis II.3

I introduced the Particle Filter Velocity Obstacle method, a comprehensive strategy where the uncertainty degree is instantly calculated using the Particle Filter-based state perception approach. This method effectively unifies the state perception and motion planning components, creating a seamless and integrated navigational framework. The introduced algorithm can reach safer results than those planning algorithms that use constant uncertainty degrees in motion planning.

Journal publications to the thesis [J]: [4], [5]
Conference publications to the thesis [C]: [5], [6], [7], [8]
In progress [IP]: [2]

Chapter 5

Motion planning in a structured environment with human presence

Until now, the main goal was to reach the goal position resulting in a collision-free motion. Now, a structured environment is established where different aspects should be considered. First, lanes are constructed using Bezier curves, and different methods the lane-keeping algorithm and traffic regulation-based methods are introduced. Another aspect of a structured environment is the human presence. The agent has to ensure the collision-free motion if it is possible or generate the least damaging collision because human life is the most important! In human presence, the human-tracking task can also be a challenging task, because the agent must follow the desired person in a dynamic environment.

5.1 Traffic Regulation Velocity Obstacle (TRVO) and Lane Keeping Velocity Obstacle (LKVO) method

As it was seen in Section 1.2, there are even more autonomous robots in factories and warehouses. In the future, maybe they will also adapt the rules in the motion planning algorithms in these scenarios, which are already introduced in the real autonomous car industry, like traffic rules, lane keeping algorithms, or emergency breaking in extreme situations where the probability of a possible collision is high. The main goal of my research was to introduce novel algorithms for mobile agents that can consider these constraints and rules in a structured environment.

5.1.1 Traffic Regulation Velocity Obstacle (TRVO) method

The main idea of the Traffic Regulation Velocity Obstacles (TRVO) method is to choose a velocity for the agent that will satisfy the basic rules of the Traffic Regulation. This method was inspired by [129, 130], where COLREGS (International Regulations for Preventing Collisions at Sea) were used for Unmanned Surface Vehicles (USV).

The *TRVO* method considers four rules for the motion planning:

- Crossing from the left
- Crossing from the right
- Overtaking
- Head-on

The main concept of this method is to select the velocity areas from the *RAV* that will ensure compliance with the rules. These areas can be denoted for every obstacle by S_r, S_f, and S_d. Using a velocity vector for the agent from these areas will result in different maneuvers for the robot during its motion. r has a meaning of rear maneuver, d is the divergent and f is the front maneuver in consideration of the obstacle and the robot. The different subsets are presented in Figure 5.1.

For a given $\mathbf{v}_A$, if it is not parallel to $\mathbf{v}_{Bi}$ the intersection point of the paths of A and B_i can be determined as:

$$\mathbf{p}_A + \mathbf{v}_A t_A = \mathbf{p}_{Bi} + \mathbf{v}_{Bi} t_{Bi} = \mathbf{p}_x \tag{5.1}$$

where $\mathbf{p}_x$ is a point in the workspace where the obstacle and the robot would intersect their path during their motion in the future or in the past.

The different subsets of *RAV* can be defined using the value of t_A and t_{Bi} as:

- $0 < t_A < t_{Bi} \Rightarrow \mathbf{v}_A \in S_f$
- $0 < t_{Bi} < t_A \Rightarrow \mathbf{v}_A \in S_r$
- $min\ (t_A,\ t_{Bi}) < 0 \Rightarrow \mathbf{v}_A \in S_d$
- $t_A = t_B$ and $min(t_A,\ t_{Bi}) > 0 \Rightarrow$ collision

where S_f means that selecting a velocity vector from this set of velocity vectors, will result in a maneuver that the agent will go in front of the obstacle so it reaches the $\mathbf{p}_x$ position (5.1) earlier. The opposite situation is when the velocity vector is selected from S_r, in this situation the obstacle reaches the $\mathbf{p}_x$ position earlier. In this case, the

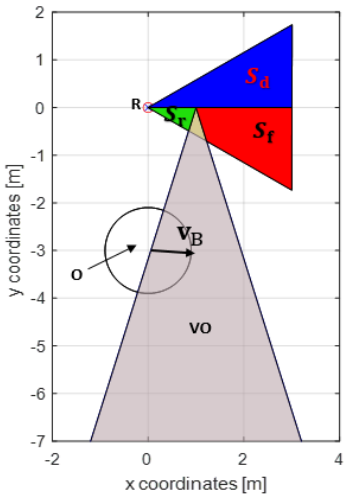

Fig. 5.1 Subsets of RAV

obstacle reaches a position earlier than the agent. Selecting a velocity vector from S_d means that the agent and the obstacle do not cross the path of each other in the future.

Figure 5.1 shows a situation where the velocities can be divided into four subsets. The red-colored subset represents the velocity vectors of the agent that would result in a front maneuver to the obstacles. The green area consists of the velocities resulting in a rear maneuver. The blue area means the divergent velocity vectors of the agent and the grey area is the VO.

In every sampling time, these subsets must be constructed. With the knowledge of the measured position and velocities of the agent and the obstacles, the actual rule can be constructed. At Crossing from the right situation (the obstacle comes from the right direction towards the agent and the agent must give priority), the velocity can be chosen from the union of S_r and S_d. At Crossing from the left situation, the velocity can be chosen from the union of S_r, S_f. and S_d, so form the whole RAV. In this situation, the agent has priority, but it can give it up as well. In a Head-on situation, every velocity vector can be chosen that will result in a right maneuver (considering the angles of the velocity vectors) to the corresponding obstacle. In the Overtaking situation, those velocities can be selected that will contribute a left maneuver to the corresponding obstacle. (The algorithm is applied in right-hand traffic situations).

5.1.2 Lane Keeping Velocity Obstacles (LKVO) method

The Velocity Obstacles method and the Safety Velocity Obstacles method can be used even if there is no lane inside of the workspace of the agent. However, if there is a lane, then the Lane Keeping Velocity Obstacles (LKVO) method can ensure an appropriate

solution. The structure of the lanes is constructed in a general way using Bezier splines [131]. Previously, I also investigated motion planning in straight lanes [C11].

The basic of Bezier splines

For creating the lanes, splines can be used as an acceptable solution opportunity. Different types of splines are usable, e.g: Bezier [132], B-splines [133], Catmull-Rom [134]. The spline can be divided into segments, and each segment is an n degree polynomial. The Bezier spline was chosen for constructing the lane. The lane is fixed during the motion, and as an assumption, the control points ($\mathbf{P}_i$) of the spline are known at the beginning of the motion planning. The Bezier splines can be created as:

$$\mathbf{Bez}(t) = \sum_{i=0}^{n} b_{i,n}(t)\, \mathbf{P}_i \tag{5.2}$$

where $t = 0...1$, $\mathbf{P}_i$ is a two-dimensional control point of a segment, there are $n+1$ control points in each segment and $\mathbf{Bez}(t) : [0..1] \subset \mathbb{R} \rightarrow \mathbb{R}^2$. The Bernstein polynomial can be calculated as:

$$b_{i,n}(t) = \binom{n}{i} t^i\, (1-t)^{n-i} \tag{5.3}$$

where $i = 0...n$ and

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} \tag{5.4}$$

Structure of the lanes

The borders of the lanes are calculated using second degree Bezier splines. First of all, one side of the borders shall be calculated, the other border can be defined by using an offset for the spline. For a second degree Bezier spline three control points have to be used in every segment ($\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$). Every control point has two coordinates (x, y). The second degree Bezier spline is represented in (5.5), by using (5.2) with $n = 2$ substitution:

$$\mathbf{Bez}(t) = \mathbf{P}_0\, (1-t)^2 + 2\, \mathbf{P}_1\, t\, (1-t) + \mathbf{P}_2\, t^2 \tag{5.5}$$

As an expectation, the border has to ensure the zero and first order continuity. For the zero order continuity the first control point of the next segment must be the same as the last control point of the previous segment. If $\mathbf{Bez}_1$ means the Bezier curves to the first segment (with control points $\mathbf{P}_0, \mathbf{P}_1$ and $\mathbf{P}_2$) and $\mathbf{Bez}_2$ means the Bezier curves to the second segment (with control points $\mathbf{P}_2, \mathbf{P}_3$ and $\mathbf{P}_4$) then for the first order

continuity, the next equations must be fulfilled:

$$\mathbf{Bez}_1'(1) = \mathbf{Bez}_2'(0) \tag{5.6}$$

Where $\mathbf{Bez}_1'(1)$ means the time-derivation of the $\mathbf{Bez}_1(t)$ and substituted the value of 1 into the derivated equation.

$\mathbf{Bez}_1'(t)$ can be calculated as:

$$\mathbf{Bez}_1'(t) = 2\,\mathbf{P}_0\,t - 2\,\mathbf{P}_0 + 2\,\mathbf{P}_1 - 4\,\mathbf{P}_1\,t + 2\,\mathbf{P}_2\,t \tag{5.7}$$

$\mathbf{Bez}_2'(t)$ can be calculated as:

$$\mathbf{Bez}_2'(t) = 2\,\mathbf{P}_2\,t - 2\,\mathbf{P}_2 + 2\,\mathbf{P}_3 - 4\,\mathbf{P}_3\,t + 2\,\mathbf{P}_4\,t \tag{5.8}$$

After the substitution the result is:

$$\mathbf{P}_3 = -\mathbf{P}_1 + 2\,\mathbf{P}_2 \tag{5.9}$$

If all the points are known that the spline has to contain then the Bezier spline can be already calculated using (5.5) and (5.9).

Figure 5.2 illustrates a Bezier spline with two segments. The second control point of the second segment ($\mathbf{P}_3$) is calculated using (5.9) ensuring the first order continuity. If the Bezier spline has more than two segments, the control points of the segments can be calculated with the same algorithm.

After all of the control points of all segments have been defined, the whole Bezier spline can be established. To get the other side of the corridor, another Bezier spline is needed that always has the same distance from the previous spline. The resulted Bezier splines that generate the lanes are presented in Figure 5.2. The control points are marked with green shaped x-s, and the borders of the lanes are shown with blue color.

Steps of LKVO

The main concept of the *LKVO* method is to select a velocity vector for the robot with that the agent will stay inside of the lane if it is possible or reach the boundary of the lane in the furthest time.

To detect, when the robot would reach the boundary of the lane, the intersection of the line of the velocity and second-order Bezier spline has to be calculated. Suppose the line has the (normal vector) equation.

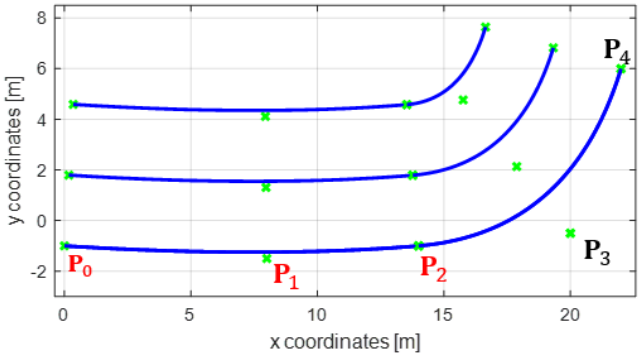

Fig. 5.2 Lanes with the control points

$$ax + by = c \tag{5.10}$$

In vector form:

$$\mathbf{A}^T \cdot \mathbf{X} = c \tag{5.11}$$

Now $\mathbf{A}^T = (a,b)$, $\mathbf{X}(t) = \mathbf{Bez}(t)$ from (5.5) (which belongs to the right lane or left lane depending on the investigation of the direction) because for the border of the lanes, second order Bezier spline is used. After the substitution of (5.5) into (5.11), the result is:

$$(1-t)^2 \left(\mathbf{A}^T \cdot \mathbf{P}_0\right) + 2\,t\,(1-t)\left(\mathbf{A}^T \cdot \mathbf{P}_1\right) + t^2 \left(\mathbf{A}^T \cdot \mathbf{P}_2\right) - c = 0 \tag{5.12}$$

Using (5.12) for every investigated velocity vector from the corresponding subset that satisfies the TRVO algorithm, every intersection point can be defined. It is possible that for a selected velocity of the robot, there are more intersection points on the Bezier spline. In that case, the algorithm has to choose the closest intersection point to the position of the robot with the right orientation. For every velocity $\mathbf{v}_A$, it has to be calculated when the robot would reach the right $(t_R(\mathbf{v}_A))$ and the left boundary $(t_L(\mathbf{v}_A))$ of the lane and use the minimum from them.

The cost value of the *LKVO* method can be defined after a normalization

$$C_{LK}(\mathbf{v}_\mathrm{A}) = 1 - \frac{\min(t_R(\mathbf{v}_A),\, t_L(\mathbf{v}_A), t_H)}{t_H} \tag{5.13}$$

where t_H is a given time horizon, the cost value is even smaller if the robot will reach the boundary of the lane in a further time. If both of $t_R(\mathbf{v}_A)$ and $t_L(\mathbf{v}_A)$ are infinite numbers, then the robot will never reach the boundary, it will move in the lane. In that

case t_H must be used in the cost function instead of $\min(t_R(\mathbf{v}_A),\ t_L(\mathbf{v}_A))$. (5.13) should be used if the robot is inside of the lane. A new logical variable can be introduced (*inLane*) that has a value of 1 if the robot is inside of the lane and the value of the variable is 0 if the agent is outside of the lane. As an assumption the robot leaves the lane in the left side. So the extended cost value for this method can be defined as:

$$C_{LK}(\mathbf{v}_\mathrm{A}) = inLane\ \left(1 - \frac{\min(t_R(\mathbf{v}_A),\ t_L(\mathbf{v}_A)), t_H)}{t_H}\right) + (1 - inLane)\ \frac{t_R(\mathbf{v}_A)}{t_H} \quad (5.14)$$

because if the robot is outside of the lane, it has to select a velocity vector that will result the lane entering back to the right as fast as it is possible (if $t_R(\mathbf{v}_A)$ exists). If there is no opportunity to select a velocity vector resulting lane reaching, the same strategy must be used as in the case when the robot is inside of the lane.

As in SVO method, at this algorithm it is also a weakness that using only the *LKVO* method the agent has a slow motion inside of the lane because the main goal is to keep the lane if it is possible. The target position reaching has in this algorithm no effect.

So an appropriate solution idea is using a similar cost function as it was used at the SVO method.

$$Cost(\mathbf{v}_\mathrm{A}) = \gamma\, C_{LK}(\mathbf{v}_\mathrm{A}) + \beta\ C_G(\mathbf{v}_\mathrm{A}) \quad (5.15)$$

where $\gamma \geq 0$. In that case, the motion planning can be influenced by the target reaching and the lane keeping at the same time.

So the steps of the motion planning algorithm are:

- Calculate the VO sets for every obstacle (Section 2.1).
- Calculate the subsets that satisfy the Traffic Regulation rules (Section 5.1.1).
- Make a grid from the investigated velocity vectors.
- Calculate the cost value for every grid point in the *RAV* set (after calculating every part of the cost function described in Section 5.1.3).
- Select the velocity vector that has minimal cost value.

5.1.3 Combination of the different methods

A cost function can be constructed that contains every above-presented strategy (LKVO, TRVO, SVO):

$$Cost_{total}(\mathbf{v}_A) = \alpha\, C_S(\mathbf{v}_A) + \beta\ C_G(\mathbf{v}_A) + \gamma\, C_{LK}(\mathbf{v}_A) \quad (5.16)$$

where every part of the cost function is the same as it was introduced previously and in Section 3.1 and in Section 5.1.2. The α, β, and γ parameters will influence which strategy will play a higher role during the motion planning at a specific sampling time. These parameters are given at the beginning of the motion planning algorithm, and they have the same value during the whole motion (or they can also be changeable considering the DSVO, UCVO, or PFVO methods). The exact values of the parameters can be specified considering the expected solution strategy using the empirical parameter tuning methodology with the experimental results.

A cost value can be calculated for the velocity vectors of the appropriate subsets of the *RAV* (see Section 5.1.1). The best option can be selected using an optimization method as it was seen in Section 3.2 too.

5.1.4 Experiments and Results

In this section several simulation results are presented. Because of the low calculation cost, a grid-based solution is introduced. The $Cost_{total}(\mathbf{v}_A)$ value is calculated in every grid point. The optimal solution can be generated selecting the velocity vector from the grid that has the minimal cost value hence ensures a collision-free motion for the robot using the traffic rules and the lane-keeping algorithm.

There are two obstacles in the workspace of the agent: the first obstacle is a moving obstacle that approaches to the agent in the opposite lane; the second obstacle is a static obstacle located in front of the robot in the same lane where the agent is (at the start). The parameters have been determined using empirical results. These examples show the differences between the introduced strategies and the result of the strategy where every part plays a role in the cost function. In every example, (5.16) is used with different parameter values considering the desired strategy. As an assumption, the *TRVO* method is used in every example.

The fastest solution

To get fastest solution for the target reaching the parameters must be set as:

- $\alpha = 0$
- $\beta = 1$
- $\gamma = 0$

In this case the agent will select a velocity vector resulting the fastest motion as it is represented in Figure 5.3, where the grid is represented using the little red x-s, the black line shows the previous path of the robot and the other notations are the same as

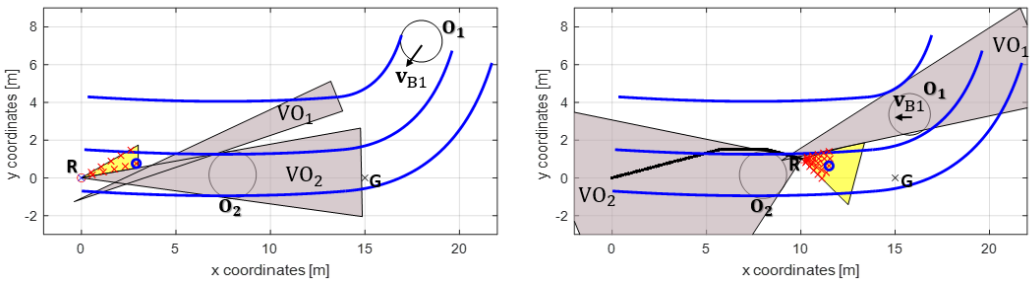

Fig. 5.3 Selecting the velocity vector for the robot using the fastest solution

previously. It can be recognized that the robot has a tangential situation with the static obstacle during the motion. The robot executed the overtaking maneuver before the moving obstacle crosses the path of the robot, ensuring the target reaching as fast as it is possible. The soft-landing algorithm is used during the method. In that case, the closer the robot is to the goal the smaller velocity vector will be selected.

Lane keeping algorithm

To execute the lane keeping algorithm the parameters must be set as:

- $\alpha = 0$
- $\beta = 0$
- $\gamma = 1$

In that case the robot will select a velocity vector that will result to stay inside of the lane for the longest time. The result of this method is presented in Figure 5.4. As it is shown the moving obstacle has already gone before the agent would have started the overtaking maneuver.

On the left side of Figure 5.5, the lane-keeping algorithm is presented in the situation when the robot is outside of the lane. In that case, if the robot has the opportunity, then a velocity vector is selected that results in the lane entering in the next time interval.

On the right side of Figure 5.5, the final path of the motion is presented. If the robot is outside of the lane, then the domination of the lane-entering maneuver is remarkable. **S** means the start position of the robot.

Combination of every method

To use all of the introduced methods together, the parameters must be set as:

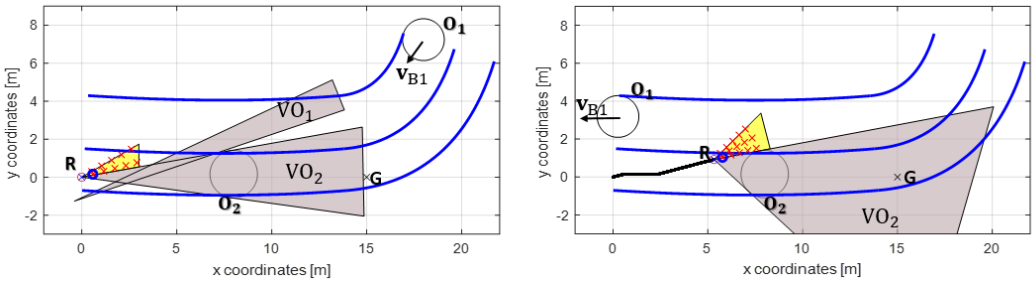

Fig. 5.4 Selecting the velocity vector for the robot using LKVO

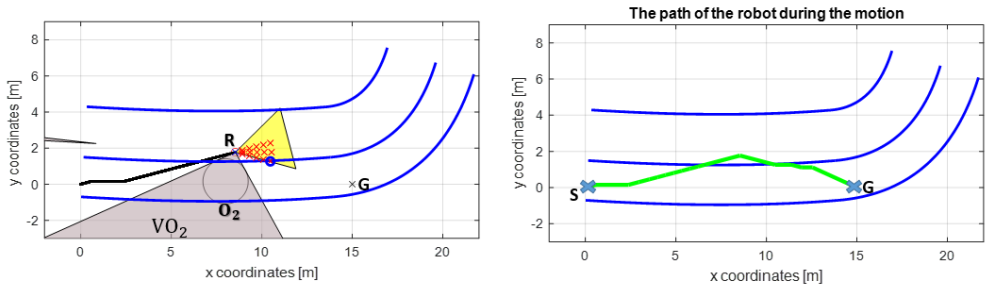

Fig. 5.5 Selecting the velocity vector for the robot using LKVO outside of the lane and path

- $\alpha = 1$
- $\beta = 1$
- $\gamma = 1$

In the beginning of the motion, the *LKVO* method plays a higher role. The velocity vector with the minimal cost value ensures the lane keeping as it is presented in Figure 5.6 ($t = 1s$).

As the agent nears to the static obstacle, it has a small velocity until the moving obstacles execute its motion in the next lane. After that, the agent starts the overtaking maneuver. Velocity is selected, resulting in a safe motion for the robot because the static obstacle is close to the robot. So at this moment, the *SVO* method has a higher impact on the cost ($t = 8s$). If the robot is out of the lane and it has the chance to come back, then it will execute this maneuver immediately.

After the robot passed the static obstacle, it reaches the target position as fast as it is possible ($t = 10s$ and $t = 12s$). This example illustrated if the combined cost function is used with all components presented in (5.16), how the location of the obstacles and their velocities influence which component of the cost function will dominate at the velocity selection.

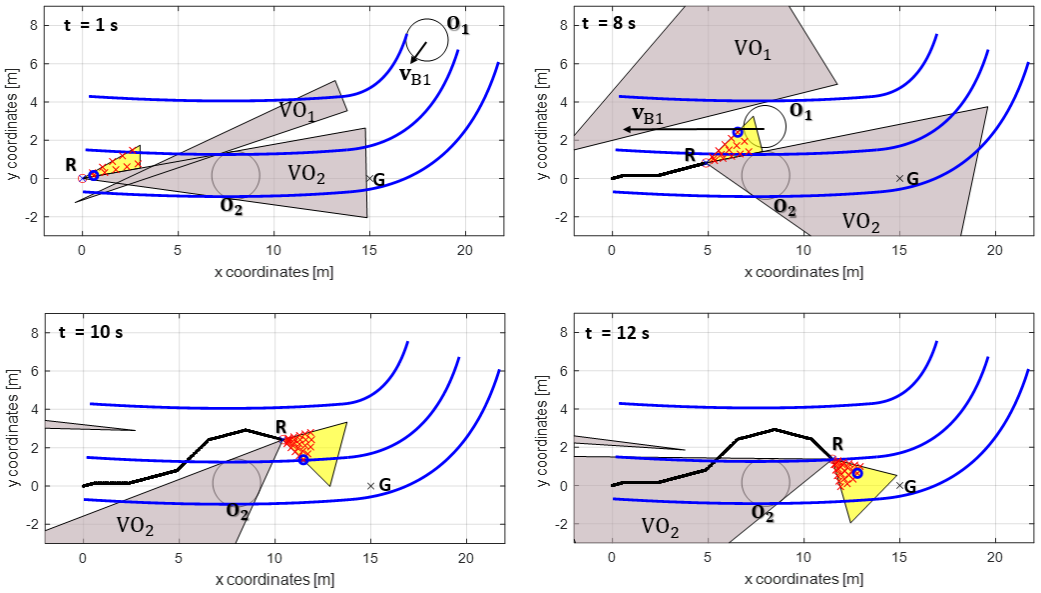

Fig. 5.6 The result of the motion planning using the combined cost function

Simulation in CoppeliaSim

CoppeliaSim (V-REP) [135] provides an appropriate solution opportunity to test mobile and non-mobile robots using motion planning algorithms. Several types of robots can be used for test cases. The result of the defined algorithm was also tested in the CoppeliaSim simulation environment. The motion planning algorithm was implemented in MATLAB, and there is the opportunity to connect the V-REP simulation environment with the MATLAB.

In the simulation, an omnidirectional robot was used (blue colored). In that case, the direction of the movement can be changed in every sampling time if it is necessary. In the first example, there is a static obstacle (presented with a grey colored cylinder) between the robot and the target point. The goal will be reached using the fastest solution without the TRVO method. The video of the solution in CoppeliaSim can be checked in YO[2]. In the second example, the TRVO algorithm is also considered. The workspace of the robot is presented in Figure 5.7. There are a static ($\mathbf{O}_1$ presented with grey colored cylinder), and two moving obstacles ($\mathbf{O}_2$, $\mathbf{O}_3$- presented with grey colored differential driven robots) in the environment of the robot. In consideration of the $\mathbf{O}_3$, an overtaking maneuver is presented, ensuring an evasive maneuver. $\mathbf{O}_2$ is crossing from the left, so the agent can execute its motion to the goal position without giving priority to the obstacle. The video of the motion is presented in YO[3].

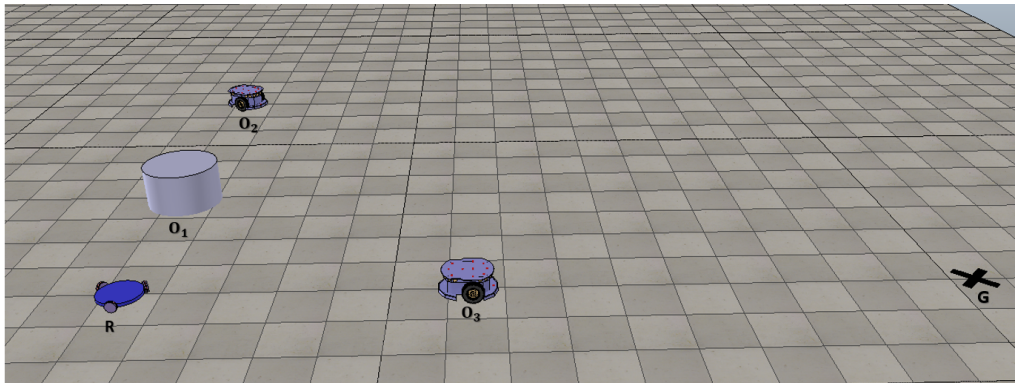

Fig. 5.7 CoppeliaSim simulation

5.2 Collidable Velocity Obstacle (CVO) method

As it was presented previously, if there is an opportunity then the main goal for the agent is to execute its motion to the target position, generating a collision-free motion. On the other hand, there are also situations when there is no option to select a collision-free velocity vector for the agent. At that time, most of the motion planning algorithms return with error because they cannot select an available velocity vector that would cause collision-free target reaching. In that case, with autonomous driving, the only opportunity is to use the emergency braking function [136, 137] which can cause huge damage between the robot and the environment.

Using the reactive motion planning methods, as it was introduced previously, it is a challenging task to generate evasive maneuvers that can ensure a safe motion for the environment and the agent. The task is more difficult if there is no available velocity vector that would cause evasive motion for the robot in the workspace in the future. I introduced a novel heuristic motion planning algorithm that can select the velocity vector that would cause the least damage in the future time between the agent and the obstacles. If there are reachable avoidance velocity vectors then the algorithm selects the velocity vector for the robot using a cost function-based algorithm.

5.2.1 The novel motion planning algorithm of CVO

Because the obstacle is a passive element in the workspace, its velocity vector is supposed to be constant during a sampling time, the velocity vector of the agent will be changed considering the new information that will be received at the next sampling time. If there is no velocity vector that would be in *RAV* at an actual sampling time, then the previous presented motion planning algorithms generate an error message and cannot generate a solution for the motion planning task (in autonomous car systems, the emergency braking is the only applicable method in that case). Now, the Collidable

Velocity Obstacles (CVO) method is introduced to solve this problem. We find the B_{i_c} obstacle that is reached at the latest time, it is ignored to determine the RAV. From the resulting RAV, the velocity vector is selected that will cause the least damage if a collision occurs with B_{i_c}.

In the first step, because there are no velocity vectors in the *RAV*, it has to be selected, which obstacle influences less the motion of the robot so which VO could not be considered from the workspace of the agent at the sampling time. This can be calculated using the introduced Precheck algorithm (Section 3.1).

Later on, that VO_{ic} must not be considered at the velocity selection algorithm whose $t_{min_{A,B_i}}$ is maximum. That obstacle will not be taken into account which will result in a collision in the latest time (in this case, maybe it will also change its velocity vector until the collision).

$$i_c = \arg\max_i (t_{min_{A,B_i}}) \tag{5.17}$$

After that, it must be checked whether there is an available velocity vector in *RAV*. If not, then the previous process must be continued until the *RAV* will exist.

For generating the least damage, that velocity vector will be chosen for the robot where the difference between the velocity of the robot and the B_{i_c} is minimum. As an assumption, the smaller is the difference between the vectors, the smaller will be the damage after the collision.

The velocity vector $(\mathbf{v}_{\mathrm{Bic}})$ of the obstacle (B_{i_c}), whose VO_{ic} set was eliminated, must be subtracted from the velocity vectors $(\mathbf{v}_{\mathrm{Aj}})$ of the robot from the RAV. The smallest distance is looked for as it can be seen in the following equation:

$$\mathbf{v}_A = \arg\min_{\mathbf{v}_{Aj}} ||\mathbf{v}_{Aj} - \mathbf{v}_{B_{ic}}|| \tag{5.18}$$

where $\mathbf{v}_{Aj}$ means the velocity vector of the robot from the RAV set and $\mathbf{v}_{B_{ic}}$ is the velocity vector of the obstacle. (If more obstacles VO must be eliminated, then the absolute difference of the velocities must be calculated for every obstacle). The least damage between the robot and the investigated obstacle(s) will be resulted in using the velocity $\mathbf{v}_A$ generating the smallest difference between the vectors.

The whole algorithm is presented in Algorithm 1, where K_{grid} means the number of the velocity vectors in the grid and $\mathbf{v}_{grid}(k)$ means the k^{th}velocity vector for the agent on the grid.

If there is no opportunity for the agent to miss the collision with an obstacle, then it must be calculated what will be the velocity vectors of the agent and the obstacle after a perfectly elastic collision.

First, the Law of Conservation of Momentum can be used:

$$m_A\mathbf{v}_A + m_B\mathbf{v}_B = m_A\mathbf{v}_{A_{new}} + m_B\mathbf{v}_{B_{new}}, \tag{5.19}$$

where m_A and m_B present the masses of the robot and the obstacle respectively and $\mathbf{v}_{A_{new}}$ is the new velocity vector of the agent after the collision, $\mathbf{v}_{B_{new}}$ is the new velocity vector of the obstacle after collision with the robot. After that, in a closed, compact system, the law of conservation of kinetic energy can also be used:

$$\frac{1}{2}m_A\mathbf{v}_A^2 + \frac{1}{2}m_B\mathbf{v}_B^2 = \frac{1}{2}m_A\mathbf{v}_{A_{new}}^2 + \frac{1}{2}m_B\mathbf{v}_{B_{new}}^2 \tag{5.20}$$

In the collision situation, using (5.19) and (5.20), the new velocity vectors for the agent and the obstacle can be determined.

Input: RV, RAV, VO, $\mathbf{p}_\mathrm{A}$, $\mathbf{v}_\mathrm{A}$, $\mathbf{p}_\mathrm{B}$, $\mathbf{v}_\mathrm{B}$ $\mathbf{v}_{grid}$
Output: $\mathbf{v}_A$
for i=*1...m* **do**
  $t_{min_{A,Bi}} = \frac{-(\mathbf{p}_\mathrm{A}-\mathbf{p}_\mathrm{Bi})(\mathbf{v}_\mathrm{A}-\mathbf{v}_\mathrm{Bi})}{||\mathbf{v}_\mathrm{A}-\mathbf{v}_\mathrm{Bi}||}$;
  $d_{min_{A,Bi}} = ||(\mathbf{p}_\mathrm{A} + \mathbf{v}_\mathrm{A}t_{min_{A,Bi}}) - (\mathbf{p}_\mathrm{Bi} + \mathbf{v}_\mathrm{Bi}t_{min_{A,Bi}})||$;
end
$VO_{skip} = \emptyset$;
while $RAV == \emptyset$ **do**
  $i_c = \arg\max_i(t_{min_{A,B_i}})$;
  $VO_{skip} = VO_{skip} \cup VO_{ic}$;
  $RAV = RV$ - (VO - VO_{skip});
end
if $VO_{skip} \neq 0$ **then**
  $\mathbf{v}_A = \mathbf{v}_{grid}(1)$;
  $||\Delta\mathbf{v}_{min}|| = ||\mathbf{v}_{grid}(1) - \mathbf{v}_{B_{ic}}||$;
  for k = *2...*K_{grid} **do**
    $||\Delta\mathbf{v}|| = ||\mathbf{v}_{grid}(k) - \mathbf{v}_{B_{ic}}||$;
    if $||\Delta\mathbf{v}|| \leq ||\Delta\mathbf{v}_{min}||$ **then**
      $||\Delta\mathbf{v}_{min}|| = ||\Delta\mathbf{v}||$;
      $\mathbf{v}_A = \mathbf{v}_{grid}(k)$;
    end
  end
end
else
  $\mathbf{v}_A$= argmin $Cost(\mathbf{v}_{grid})$;
end

Algorithm 1: The CVO algorithm

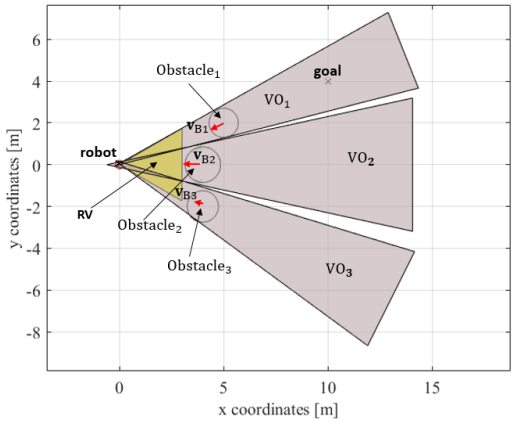

Fig. 5.8 There is no velocity vector which is reachable and available

where $\mathbf{p}_B$ consists position of every obstacle and $\mathbf{v}_B$ consists velocity of every obstacle.

It is important that the *RV* set always consists the velocity vector that would result in the emergency braking, so the *CVO* algorithm could generate the emergency braking solution too, if there is not better solution (generating less damage).

If the RAV is not empty and there are more velocity vectors from which the robot could choose, then the velocity can be selected using the Safety Velocity Obstacles method which was introduced in Section 3.1.

5.2.2 Experiments and Results

In this section, the simulation results of the CVO algorithm are presented. Every situation is compared with the Emergency braking solution as well.

The robot can avoid the collision using the CVO method

In Figure 5.8, a situation is shown where the RAV set is empty. There are three moving obstacles in the workspace of the agent. It can be seen that at this sampling time, the robot has no opportunity to select a velocity vector that would cause a collision-free motion in a future time.

In that case, the CVO algorithm can be used that was presented in Section 5.2.1. As a result, the VO set, belonging to the first obstacles, can be eliminated because this obstacle has the biggest $t_{min_{A,Bi}}$ time when the obstacle would be at the nearest distance to the agent (collision case). In that case, there will be a RAV set from which the velocity vector for the robot can be selected using (5.18), the minimum Euclidean distance of the velocity vector of the first obstacle, and the available velocity vectors of the robot from the RAV set. The result of the velocity selection can be seen in Figure

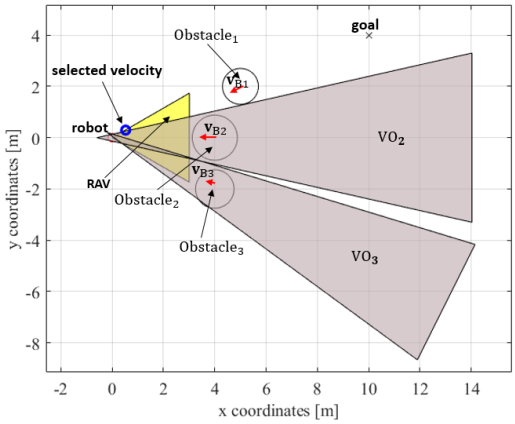

Fig. 5.9 Velocity selection using CVO method

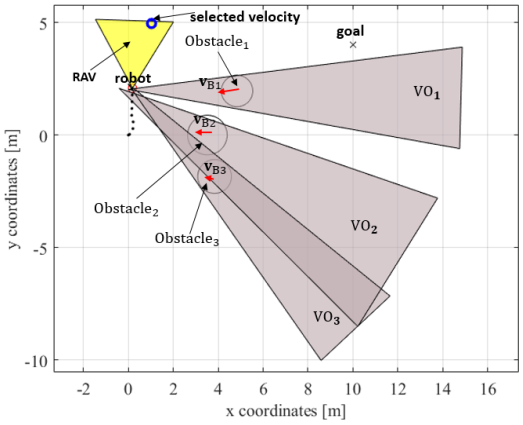

Fig. 5.10 Velocity selection using SVO algorithm

5.9. The blue circle represents the selected velocity vector. The other markings are the same as were presented in the previous Figure. The movement of the agent can be seen in a video as well YO[4].

The CVO method must be used in every sampling time if the RAV set is empty. If it is not empty, then the cost function-based velocity selection method can be used. Figure 5.10 represents an example of when the *RAV* set is not empty, the cost-function-based method (based on Section 3.1) can be used. In this example, the $\alpha_i = 1$ for every obstacle and $\beta = 0.8$.

Using the Emergency braking algorithm, the robot cannot select a velocity vector that would result in a collision-free motion in the workspace at the first step, that is why it stays in the origin position until the obstacle will cause a head-on collision, generating huge damage in the robot. The collision can be seen in Figure 5.11 where the obstacles are presented with different blue colored circles, the robot is shown as a green circle. In comparison, using the CVO method, the algorithm can generate

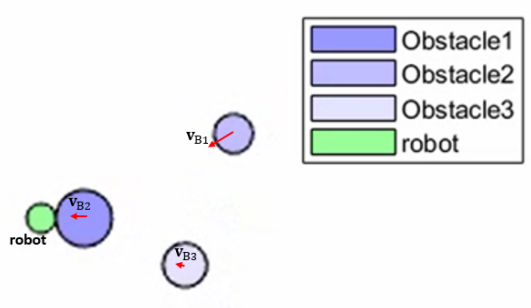

Fig. 5.11 Collision between the robot and the obstacle using the Emergency braking method

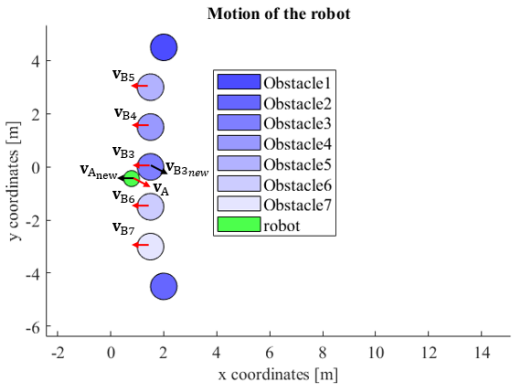

Fig. 5.12 The collision is inevitable between the agent and the obstacle

a collision-free trajectory for the agent even at the first sampling time there was no velocity vector in the *RAV* set. On the other hand, at the Emergency braking method, there is a head-on collision between the robot and the obstacle. The result of the movement of the robot can be seen in a video as well YO[5].

The collision is inevitable

In this example, there are so many obstacles in the workspace of the agent, and the collision is inevitable. Based on Algorithm 1, a velocity vector will be selected for the agent that will cause the least damage. At the moment of the collision, using (5.19) and (5.20), considering the actual velocity vector of the agent and the obstacle $(\mathbf{v}_{\mathrm{A}}, \mathbf{v}_{\mathrm{B3}})$ and the masses (m_A, m_{B3}), the velocity vectors after the collision can be calculated $(\mathbf{v}_{\mathrm{Anew}}, \mathbf{v}_{\mathrm{B3new}})$ as it is presented in Figure 5.12. The motion of the robot can be seen in a video as well YO[6].

On the other hand, if the Emergency braking method is used, then at the first sampling time, it selects a velocity vector for the agent, but after that the robot must slow down because every velocity will result in a collision, so there will be a head-on

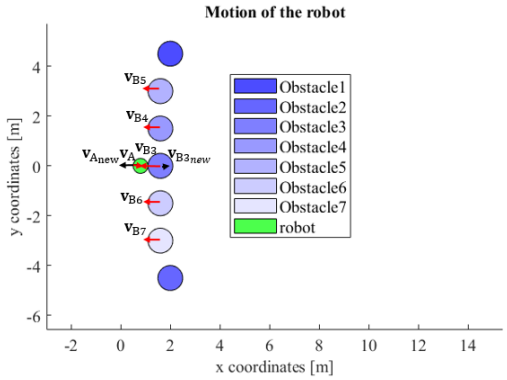

Fig. 5.13 Collision between the robot and the obstacle using the Emergency braking

collision again between the robot and the obstacle which can be seen in Figure 5.13. Comparing the CVO method and Emergency braking, it can be seen that in the CVO algorithm, the collision is not a head-on collision, so the damage is less than at the situation of the Emergency braking method as it can be seen in video YO[7].

5.3 Human Tracking Velocity Obstacle (HTVO) method

A novel motion planning algorithm was introduced that can follow a moving goal (human) so the main motivation is not to reach a fixed goal position. To reach this novel approach, the VO method was combined with the Directive Circle (DC) algorithm. The DC method designs a collision-free, near-optimal path for a mobile robot to chase another mobile robot by avoiding moving obstacles. Finding the shortest path to the another mobile robot is the first step at the algorithm. Then, the collision-free directions of the robot are computed using the velocity vectors and the Directive Circle [138]. The collision-free path closest to the optimal path is then found. These subtasks are repeated one after the other until the two objects meet. The introduced method has a better solution than a normal tracking method because next to the tracking task it can solve the collision avoidance at the same time and checking the visibility as well (the tracking method is combined with SVO reactive motion planning alorithm). The human tracking aspect can be used in different scenarios in the future. It can be used in hospitals [139], helping elderly people [140], or also in the army [21]. The movement of pedestrians between autonomous vehicles plays also an influential role in the research field. Several techniques consider an adversarial relationship between the vehicle and pedestrians [141, 142]. In several previous studies, pedestrians were assumed to adhere to predetermined routes or utilized crosswalks for traversing [143–145].

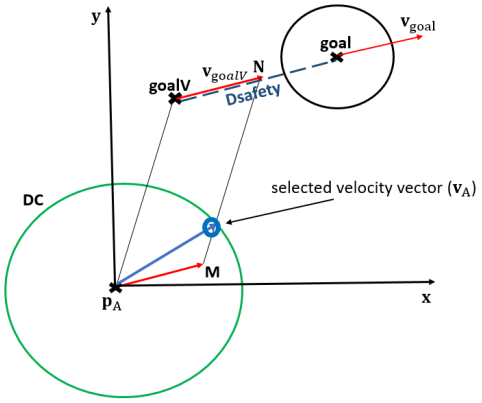

Fig. 5.14 Virtual target and Directive Circle in HTVO

In this Section, the Human Tracking Velocity Obstacles (HTVO) motion planning algorithm is introduced where the Directive Circle and the Velocity Obstacles methods are combined and extended. Using this novel method, the human tracking and obstacle avoidance aspects can be considered at the same time, generating a collision-free motion for the agent in the dynamic workspace.

5.3.1 Velocity selection for the robot

Both the obstacles and the human are represented with circles. In that case, a virtual target must be introduced which is always behind the real target position (the human that the agent follows). The distance between the real and the virtual target position is always constant. In Figure 5.14, **goal** presents the real goal position, $\mathbf{v}_{\text{goal}}$ is the velocity vector of the human and **goalV** is the virtual target position (which is next to the human with a distance of D_{safety} using the opposite direction of the velocity vector) with $\mathbf{v}_{\text{goalV}}$ velocity vector with an end point of N which is the same velocity vector as the human has ($\mathbf{v}_{\text{goalV}} = \mathbf{v}_{\text{goal}}$). The distance between the human and the virtual target position is the constant *Dsafety*. The robot is in position $\mathbf{p}_{\text{A}}$. The DC circle is presented with green color which represents a circle with a radius of the maximum velocity v_{max} of the agent. M denotes the end point of the velocity $\mathbf{v}_{\text{goalV}}$ if it starts from $\mathbf{p}_{\text{A}}$. Where MN section intersects the DC circle, it will be the velocity vector of the robot which will generate a motion for the agent resulting in reaching the virtual target position. This velocity vector must be selected for the robot if it is in the RAV set, considering the RV and VO sets in the workspace.

A grid-based velocity map is introduced to the RV set so if the desired velocity vector is not in the RAV set, then a grid-based velocity selection can be used which can be seen in Figure 5.15.

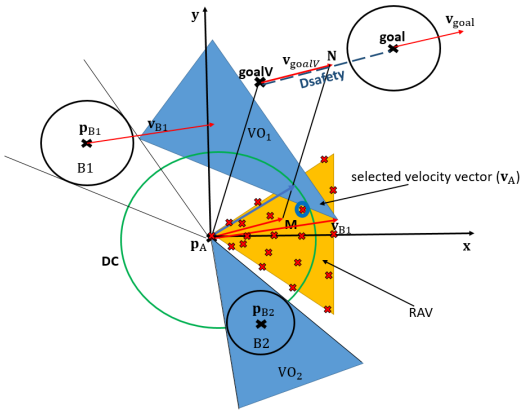

Fig. 5.15 VOs for HTVO

There are several situations that can be considered in the velocity selection method (similarly to [138]):

- If there are no intersection point between the *DC* and the *MN* section then the human tracking task is not solvable because the robot cannot select a velocity vector that is large enough to reach the virtual target position

$$DC \cap MN = \emptyset \tag{5.21}$$

- If there is one intersection point between the *DC* and the *MN* section ($\mathbf{p}_1$), then it is the solution for the velocity selection task.

$$DC \cap MN = \mathbf{p}_1 \tag{5.22}$$

$$\mathbf{v}_{\mathrm{DC}} = \mathbf{p}_1 \tag{5.23}$$

  where $\mathbf{v}_{\mathrm{DC}}$ is the ideal velocity, calculated with the DC method.

- If there are two intersection points between the *DC* and the *MN* section ($\mathbf{p}_1$ and $\mathbf{p}_2$), then that point must be selected which is closer to point *N*:

$$DC \cap MN = \{\mathbf{p}_1, \mathbf{p}_2\} \tag{5.24}$$

$$\mathbf{v}_{\mathrm{A}} = \arg\min_{\mathbf{p}_i} ||\mathbf{N} - \mathbf{p}_i|| \tag{5.25}$$

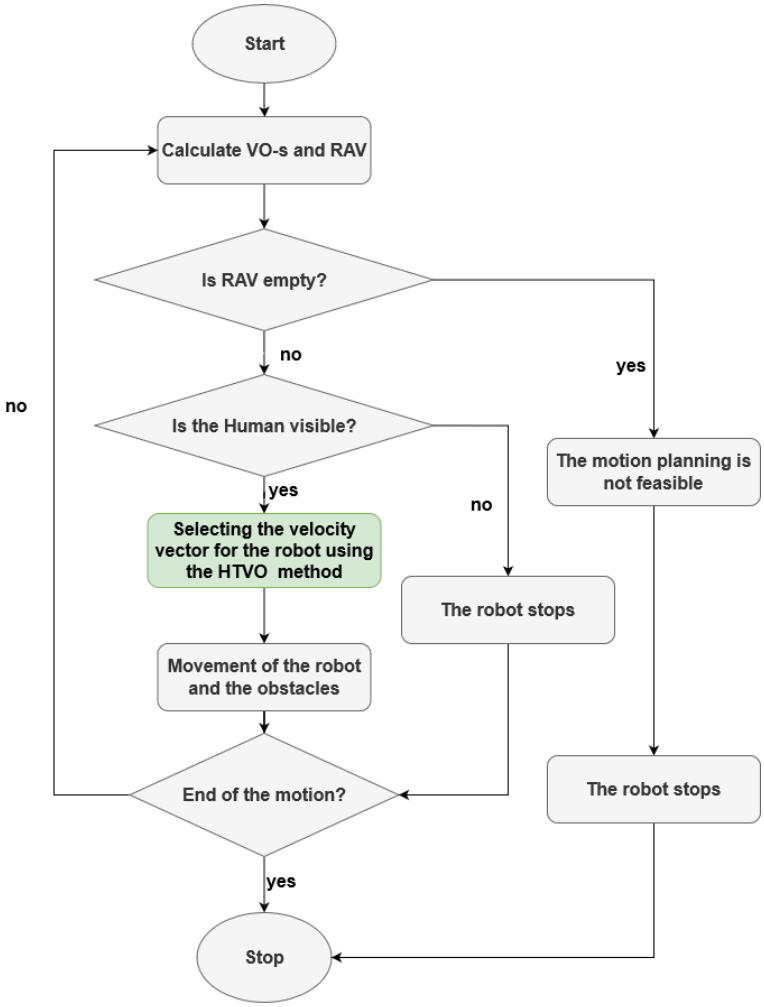

Fig. 5.16 Steps of the HTVO algorithm

- If there are two intersection points between the DC and the MN section ($\mathbf{p}_1$ and $\mathbf{p}_2$), and N is situated between these two points, then N must be selected as the velocity vector of the agent. This situation can occur when the robot will reach the virtual target position in the next time step:

$$DC \cap MN = \{\mathbf{p}_1, \mathbf{p}_2\} \ \ AND \ \ \mathbf{N} \in \mathbf{p}_1\mathbf{p}_2 \tag{5.26}$$

$$\mathbf{v}_\mathrm{A} = \mathbf{N} \tag{5.27}$$

- If the human stands in a position and it has no velocity vector, then the virtual goal is also a static point. In that case, the robot has to reach the virtual goal position as fast as it is possible. The velocity vector can be calculated as:

$$\mathbf{v}_\mathrm{A} = \frac{\mathbf{goalV} - \mathbf{p}_A}{||\mathbf{goalV} - \mathbf{p}_A||} \cdot \min(v_{max}, ||\mathbf{goalV} - \mathbf{p}_A||), \tag{5.28}$$

- If the desired velocity vector that would cause an appropriate virtual target reaching is not in the *RAV* set, then the nearest velocity vector must be selected from the introduced grid.

$$\mathbf{v}_{DC} \notin RAV \tag{5.29}$$

$$\mathbf{v}_\mathrm{A} = \arg\min_{\mathbf{v}_\mathrm{grid}(i)} ||(DC \cap MN) - \mathbf{v}_\mathrm{grid}(i)||, \tag{5.30}$$

  where every grid point must be investigated.

In Figure 5.15, a situation is represented where the velocity vector which would result the best human tracking solution is not reachable, so the nearest grid point will be selected for the agent from the *RAV* set.

It is important to note, that the HTVO method is not human-specific, it can be used to track other objects if the position and velocity vector of the object are available at the sampling time. In the current method, both the agent and the object to be tracked (human) are represented by disk-shaped objects. If the shape of the object to be tracked is different from this, then the method introduced must be extended to provide a suitable solution.

5.3.2 Visibility check

In every sampling time, it must be investigated whether the human that the robot has to follow is visible or not. If it is visible than the velocity selection method can be used that was introduced in Section 5.3.1. The human is visible if minimum one point is observable from itself using a sensor (e.g.: LiDAR). If the human is not visible, the agent stops until the robot receives information from the human again.

In Section 5.3.3, there will be an example, where the human is not visible.

In Figure 5.16, the steps of the HTVO algorithm can be seen. After calculating the VO-s and *RAV* sets, it must be checked whether there are available velocity vectors in the RAV or not. If no, then the motion planning for the mobile robot is not executable and the agent stops or the CVO algorithm can be used which was introduced in Section 5.2. If the *RAV* set is not empty, then the visibility check is the next step. If the Human is not visible, then the agent must stop, on the other hand, if it is visible, then the velocity vector for the robot can be calculated using the HTVO method. Using this

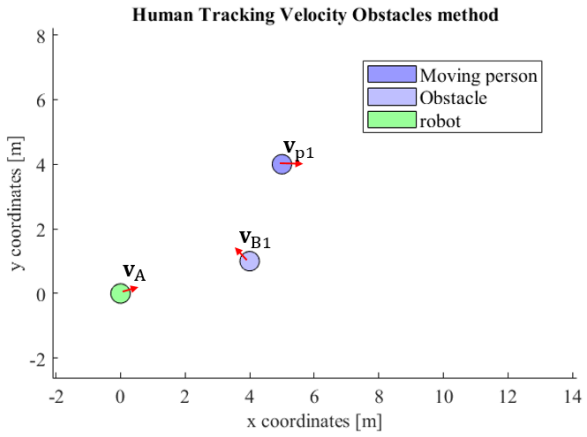

Fig. 5.17 Simulation result in MATLAB, there is one obstacle in the workspace

velocity vector for one time step, the whole method should execute again until the end of the motion is not reached.

5.3.3 Experiments and Results

In this section, the simulation results of the Human Tracking Velocity Obstacles are presented. In every simulation, the motion planning of an omnidirectional robot is presented, which follows the human.

MATLAB simulation

The motion planning algorithm was implemented first in the MATLAB environment and in the C++ environment. In every simulation, the obstacles have constant velocity vectors. In Figure 5.17, there is one obstacle in the workspace of the robot which crosses the line between the robot and the human. The obstacle is presented with blue circle with velocity vector $\mathbf{v}_{B1}$ and the human (moving person) is a lila circle with velocity vector $\mathbf{v}_{p1}$. The robot is shown as a green circle with velocity vector $\mathbf{v}_A$. The robot has always the opportunity to select the velocity vector which will result in the human tracking method. In the middle of the motion, the human is not visible because of the obstacle. In that case, the robot slows down and continues the motion only if the human is visible again. The average running time is 0.0016 s in MATLAB simulation, so the introduced method can be also used in real-world scenarios.

The motion of the robot can be seen in the following video YO[8].

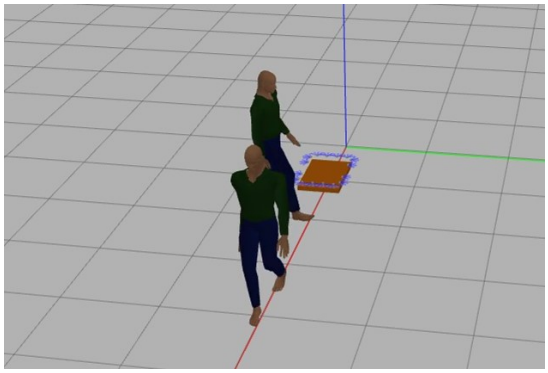

Fig. 5.18 The robot follows the person while another person passes the path

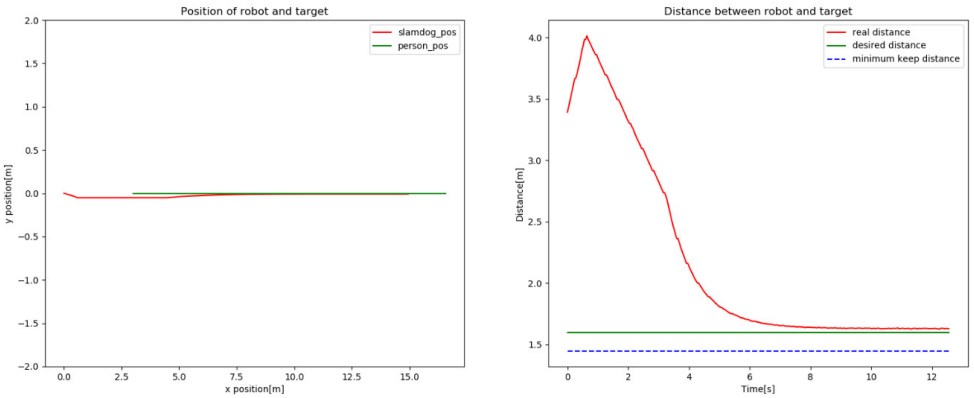

Fig. 5.19 The path and the distance of the robot and the target

ROS-Gazebo simulation

Robot simulation is a necessary tool in testing the results of the motion planning algorithms. Well-designed simulators can quickly test algorithms, design models, perform tests, and train AI systems using real-world scenarios. Gazebo provides the ability to accurately and efficiently simulate robotic populations in complex indoor and outdoor environments. To integrate ROS with stand-alone Gazebo, a set of ROS packages offer wrappers for Stand-alone Gazebo. They provide the essential interfaces to simulate robots in Gazebo with ROS messages, services and dynamic reconfiguration.

Two scenarios were tested in ROS-Gazebo environment.

1. The mobile robot follows a person who walks along a straight-line trajectory, while another person (as an obstacle) passes by, which makes the goal invisible to the mobile robot. In this case, the robot would stop moving according to the introduced algorithm until it could receive the information of its goal again after the obstacle moved away. The presented example can be seen in Figure

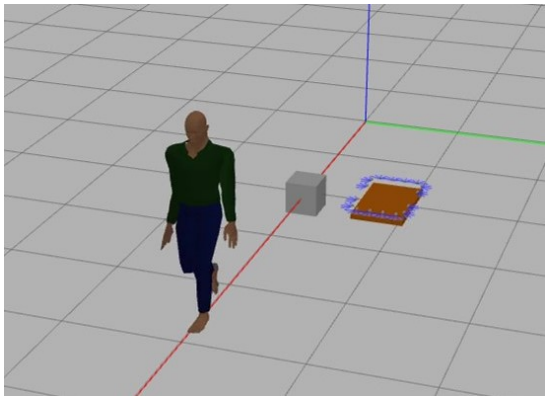

Fig. 5.20 The robot follows the person while a static obstacle is in the workspace

5.18. Additionally, Figure 5.19 left side represents the path of the robot and the human that the agent follows. Later, in the video, it can be seen that the robot stops when the other person crosses the line and after that follows the target person. The right side of the picture shows the distance between the human and the robot, and the desired distance and the minimum keeping distance. It can be seen that during the motion, the distance will be even smaller until the desired distance is reached. There is a little offset between the desired and the actual distance in the end of the motion, it is because the actuators and the delays of the system but it does not influence the solution of the motion planning method. The motion of the robot can be seen in the following video as well YO[9].

2. The mobile robot follows a goal person who walks along a straight-line trajectory, while a static obstacle stays between the target and robot. The goal person is still visible to robot although this obstacle exists. In this case the robot should manage to get around the obstacle generating an evasive maneuver and continues to track its target.

   This case can be seen in Figure 5.20.

   The path of the the motion can be seen in Figure 5.21, where an evasive maneuver is generated considering the static obstacle and after that, the target person is followed. On the right side of this figure, the distance can be seen between the agent and the person, which reaches the desired distance during the motion of the agent. In both examples, the human tracking task can be reached while the collision avoidance task is solved. The result of the motion can be seen in the following video YO[10].

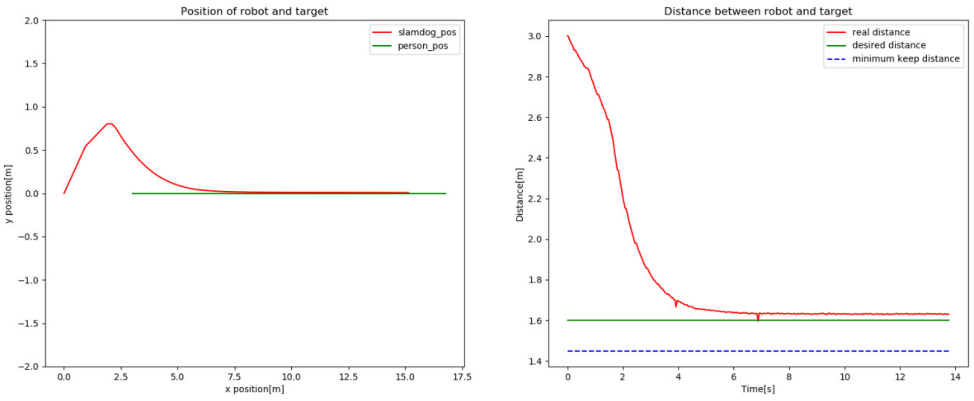

Fig. 5.21 The path and the distance of the robot and the target in the case if there is a static obstacle between the robot and the human

5.4 Discussion

In this thesis group, I have delved into various motion planning algorithms for mobile robots operating in dynamic environments, focusing on safety, efficiency, and adaptability and rule-based velocity selection.

Firstly, I introduced a novel motion planning algorithm for AGVs that can generate collision-free motion in a structured environment. The traffic regulations and lane-keeping methods were integrated into this motion planning strategy, adding to its practicality. A major innovation of this algorithm is its capacity to utilize different aspects of motion planning simultaneously to produce a sub-optimal collision-free path for the robot. I evaluated the algorithm in the CoppeliaSim simulation environment, considering only the kinematic constraints of the robot, and the results demonstrated that it efficiently generated feasible solutions. For future work, I plan to implement this algorithm on an actual robotic system. One avenue for further development is to make the parameters of the cost function adaptable in real-world scenarios, to account for sensor measurement uncertainties considering the algorithm that was introduced in Thesis group II and to expand the cost function with additional aspects of motion planning.

Next, I introduced the Collidable Velocity Obstacles (CVO) method as a specialized motion planning algorithm. This algorithm comes into play when, at the sampling time, no reachable avoidance velocity vector exists that would ensure collision-free motion. However, if there is an opportunity to select a velocity vector that would avoid a collision at a future time, a cost function is used to select the robot's velocity. I compared the CVO method to the Emergency Braking method, which is the current standard in autonomous vehicles for collision scenarios. The results showed that the

CVO method resulted in less damage to the agent compared to Emergency Braking. This suggests that the CVO algorithm holds promise for applications in factories with dense, dynamic environments. My future work in this area includes considering other aspects in scenarios where no avoidance velocity vector is available and evaluating the change in kinetic energy during a collision to select the velocity vector that minimizes damage.

Lastly, I introduced the Human Tracking Velocity Obstacles (HTVO) method, a novel motion planning algorithm that combines human tracking with collision avoidance. Through this combination, the robot can maintain a specified distance from a target person while ensuring collision-free movement in a dynamic environment. I tested and validated this method in different environments and scenarios. My future research will involve comparing it against a human tracking algorithm based on Model Predictive Control (MPC). This method has potential applications in real-world settings such as hospitals and airports, where it could significantly improve daily human life.

In summary, this thesis group represents my contribution to three novel motion planning algorithms, with an emphasis on ensuring safe and efficient navigation for mobile robots in dynamic environments. The algorithms present novel approaches to address the challenges of collision avoidance, human tracking, and reactive planning.

Thesis group III.

I devised novel motion planning algorithms in a structured environment for automated guided vehicles that take into account different scenarios, like adhering to traffic rules or lanes and introducing novel operational roles for mobile robots, such as human-following tasks.

Thesis III.1

I introduced the Traffic Regulation Velocity Obstacle method, a novel strategy allowing mobile robots to incorporate basic traffic rules and the Lane Keeping Velocity Obstacle method, a strategy that enables mobile robots to maintain lane discipline while performing target-reaching and collision avoidance tasks.

Thesis III.2

I introduced the Collidable Velocity Obstacle method which can select the velocity vector for the agent in every sampling time that results in the least damage in every

situation where the collision between the agent and the obstacles is inevitable. The algorithm had superior performance comparing with the Emergency braking algorithm.

Thesis III.3

I proposed the Human Tracking Velocity Obstacles method, a novel algorithm that simultaneously handles tracking and collision avoidance tasks, resulting in a navigation strategy that avoids collisions with the environment while effectively following the human subject.

Journal publication to the thesis [J]: [2]
Conference publications to the thesis [C]: [4], [9], [10], [11], [12], [13], [14]
In progress [IP]: [1]

Chapter 6

Conclusion

In my research, I have developed and explored a variety of novel motion planning algorithms for mobile robots operating in dynamic environments. These algorithms, which focus on safety, efficiency, adaptability, and rule-based velocity selection, have shown promising results in various testing scenarios.

Figure 6.1 represents the steps of the motion planning algorithm. As can be seen, after the robot initialization, three steps are executed in every sampling time for the mobile agent until reaching the goal position:

- State perception of the environment
- Velocity selection using a Reactive motion planning method
- Motion control

During my research, I focused on the first two steps. Considering the first step, I introduced a Kalman filter and Particle filter-based state perception methods to estimate the velocity and position vector of the obstacles occurring in the workspace of the agent. I also connected the Particle filter-based state perception with a reactive motion planning method calculating an uncertainty degree. My research mainly focused on the second step, introducing novel motion planning methods for selecting velocity vectors for the agent resulting in collision-free motion planning to the goal position. I introduced the SVO and GAVO methods, where safety and speed can be considered at the same time. Later on, I introduced the UCVO and PFVO methods where the uncertainty of the sensor measurements can be taken into account during the motion planning. I introduced the LKVO and TRVO methods for a structured environment, where the lane-keeping algorithm and the main traffic rules can be used. A human presence in the environment influences the motion of the agent during motion planning. The CVO method was introduced for situations where the collision is inevitable and

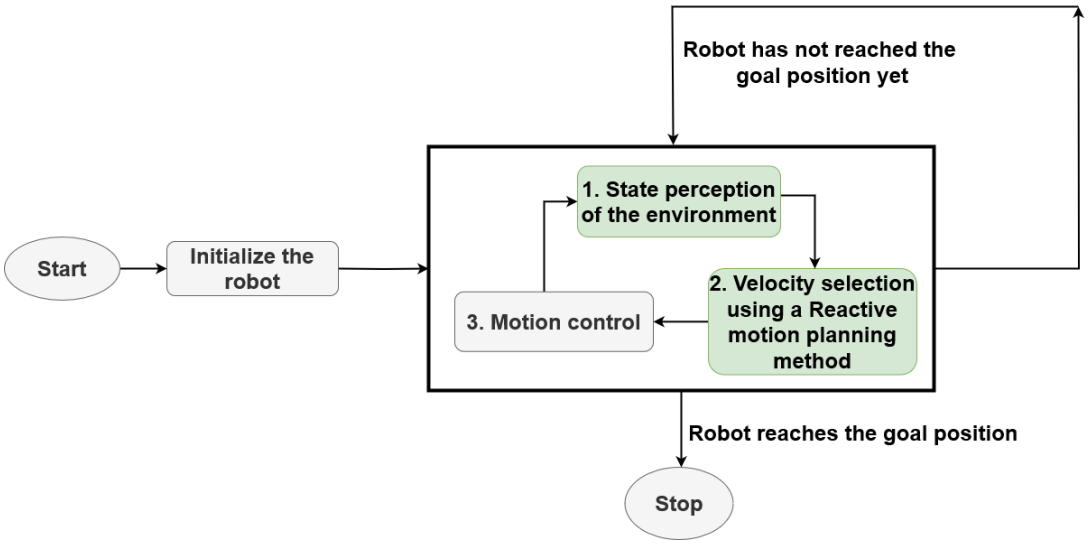

Fig. 6.1 Steps of the motion planning algorithm

minimal damage should be reached between the agent and the environment. Finally, the HTVO method was introduced for executing human tracking and collision avoidance at the same time.

Looking forward, I plan to continue refining these algorithms and exploring new avenues for their application. My future plans include extending the algorithm for differential-driven mobile robots and multi-agent scenarios and implementing these algorithms on actual robotic systems. I would like to investigate in the future, whether artificial intelligence-based motion planning methods could be combined with well-known motion planning methods to reach superior performance in the collision avoidance task. The Collidable Velocity Obstacle method could be used with the combinations of reinforcement learning methods. The potential for these algorithms to improve the safety and efficiency of mobile robots in dynamic environments is vast, and I look forward to continuing to contribute to this exciting field.

In conclusion, my research has made novel results in state estimation algorithms and collision-free motion planning for mobile robots in dynamic environments. The algorithms I have introduced and tested have shown promising results, and I believe they hold great potential for future applications in real-world settings such as factories, hospitals, and airports. In the future, it may be possible to combine the PFVO method with the results presented in Thesis Group III. Several aspects need to be considered when introducing this combination:

- Ensuring a real-time solution: in order to achieve the desired operation, attention must be paid to the implementation of a running time, including the estimation of the environment state and the implementation of the motion planning algorithm.

- PFVO - HTVO combination: in this case, the position of the person has to be determined based on LiDAR data. A solution is available in the literature that can determine the feet and the direction of the people based on the LiDAR sensor measurement data, thus the current position of the person to be followed could be determined [146–148].

In future research directions, other sensors, like cameras and sonar can be used for the motion planning algorithms in addition to the LiDAR sensor. Sensor fusion can be implemented for multiple sensors to ensure an adequate solution. For the PFVO method, an estimated position and variance are required based on the sensor data. Any sensor that provides these data can be used.

Acknowledgements

In this journey of scientific pursuit, it is with profound gratitude that I acknowledge the Almighty God, whose unwavering strength and patience have guided my research. A journey such as this is not traversed alone, and I am immensely indebted to my parents Zoltán and Ilona, and my sister, Julianna, whose constant support and encouragement have been my steadfast pillars throughout the years.

I would like to express my deepest appreciation here to my supervisor, Emese Gincsainé Dr. Szádeczky-Kardoss. Her unrelenting guidance from my bachelor's degree through to my doctoral studies has been invaluable. Her insightful advice, uplifting encouragement, and constant assistance have been instrumental in achieving the results presented in this dissertation.

My heartfelt gratitude also extends to my special fiancée, Zsuzsanna Jákob, perhaps the only physiotherapist in Hungary to spend her weekends delving into the world of my robotics papers (especially those planned for submission). Her praises and encouragement have been a sustaining force in the completion of this research.

A special acknowledgment goes to Dr. Björn Hein and Dr. Ilshat Mamaev, my supervisors during the DAAD Short-Term research scholarship in Karlsruhe. Their guidance has undoubtedly enriched this dissertation. I am also extremely thankful to Dr. Ladislau Bölöni, my supervisor in Orlando during the Fulbright and Rosztoczy scholarships, whose expertise and mentorship were invaluable.

I extend my sincerest thanks to the Department of Control Engineering and Information Technology at the Budapest University of Technology and Economics, to Dr. Bálint Kiss, the head of the department, and all of my colleagues, for providing a conducive environment for this research.

In conclusion, I would like to share my ars poetica which guided me throughout this journey: We must learn from the past, live in the present, and plan for the future. It is this philosophy that will continue to guide my academic and personal journey, and I hope it will inspire others on their paths.

Appendix A

Least square circle fitting using LiDAR sensor data

"Using the LiDAR sensor data, the least square method can be used for the circle fitting, calculating the center point of the obstacle [149].

Given a finite set of points in $\mathbb{R}^2$, say $\{(x_i, y_i) \,|\, 0 \leq i < N\}$, we want to find the circle that “best" (in a least squares sense) fits the points. Define

$$\bar{x} = \frac{1}{N}\sum_i x_i \quad \text{and} \quad \bar{y} = \frac{1}{N}\sum_i y_i \tag{A.1}$$

and let

$$u_i = x_i - \bar{x}, \quad v_i = y_i - \bar{y} \quad \text{for} \quad 0 \leq i < N. \tag{A.2}$$

We solve the problem first in (u,v) coordinates, and then transform back to (x,y). Let the circle have center (u_c, v_c) and radius R. We want to minimize

$$S = \sum_i (g(u_i, v_i))^2 \tag{A.3}$$

where

$$g(u,v) = (u-u_c)^2 + (v-v_c)^2 - \alpha, \quad \text{and where} \quad \alpha = R^2. \tag{A.4}$$

To do that, we differentiate $S(\alpha, u_c, v_c)$ with respect to α, u_c, and v_c respectively:

$$\frac{\partial S}{\partial \alpha} = -2\sum_i g(u_i, v_i) \tag{A.5}$$

Thus $\frac{\partial S}{\partial \alpha} = 0$ iff

$$\sum_i g(u_i, v_i) = 0 \tag{A.6}$$

Continuing, we have

$$\frac{\partial S}{\partial u_c} = -4\sum_i (u_i - u_c) g(u_i, v_i) \tag{A.7}$$

$$= -4\sum_i u_i g(u_i, v_i) + 4u_c \sum_i g(u_i, v_i) \tag{A.8}$$

$$= -4\sum_i u_i g(u_i, v_i) \quad \text{by (A.6))} \tag{A.9}$$

Thus, in the presence of (A.6), $\frac{\partial S}{\partial u_c} = 0$ holds iff

$$\sum_i u_i g(u_i, v_i) = 0 \tag{A.10}$$

Similarly, requiring $\frac{\partial S}{\partial v_c} = 0$ gives

$$\sum_i v_i g(u_i, v_i) = 0 \tag{A.11}$$

Expanding (A.10) gives

$$\sum_i u_i \left(u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 - \alpha\right) = 0 \tag{A.12}$$

Defining $S_u = \sum_i u_i$, $S_{uu} = \sum_i u_i^2$, etc., we can rewrite this as

$$S_{uuu} - 2u_c S_{uu} + u_c^2 S_u + S_{uvv} - 2v_c S_{uv} + v_c^2 S_u - \alpha S_u = 0 \tag{A.13}$$

Since $S_u = 0$, this simplifies to

$$u_c S_{uu} + v_c S_{uv} = \frac{1}{2}(S_{uuu} + S_{uvv}) \tag{A.14}$$

In a similar fashion, expanding (A.11) and using $S_v = 0$ gives

$$u_c S_{uv} + v_c S_{vv} = \frac{1}{2}(S_{vvv} + S_{vuu}) \tag{A.15}$$

Solving (A.14) and (A.15) simultaneously gives (u_c, v_c). Then the center (x_c, y_c) of the circle in the original coordinate system is $(x_c, y_c) = (u_c, v_c) + (\bar{x}, \bar{y})$.

To find the radius R, expand (A.6):

$$\sum_i \left(u_i^2 - 2u_i u_c + u_c^2 + v_i^2 - 2v_i v_c + v_c^2 - \alpha\right) = 0 \tag{A.16}$$

Using $S_u = S_v = 0$ again, we get

$$N(u_c^2 + v_c^2 - \alpha) + S_{uu} + S_{vv} = 0 \tag{A.17}$$

Thus

$$\alpha = u_c^2 + v_c^2 + \frac{S_{uu} + S_{vv}}{N} \quad \text{Eq. 6} \tag{A.18}$$

and, of course, $R = \sqrt{\alpha}$."

Appendix B

Application - Robot setup

An omnidirectional mobile robot was built that can be used to test the motion planning algorithms.

The different parts and the connections of the mobile agent can be seen in Figure B.1. The main component is the Raspberry PI 3B which is connected to the three DC motors, the motor driver, the LiDAR, and the power supply. As it can be seen a Battery management system was created and a ROS2 (Robot Operating System) is running on a Linux system on the Raspberry.

ROS2 [150] is a flexible and distributed framework for building robotic systems that provides a set of tools and libraries for efficient communication and development of complex robot software. The Slamtec company provides a simplified BSD-licensed ROS2 package alongside the purchased LiDAR.

The ROS2 network consists of three nodes as can be seen in Figure B.2. The node named "*sllidar_node*" handles the LiDAR by controlling its motor and forwarding incoming data to the selected communication interface through the "scan" topic. I used the provided C++ file in its original state without modification for this project. The scan topic delivers messages of type "*sensor_msgs/LaserScan*" to the

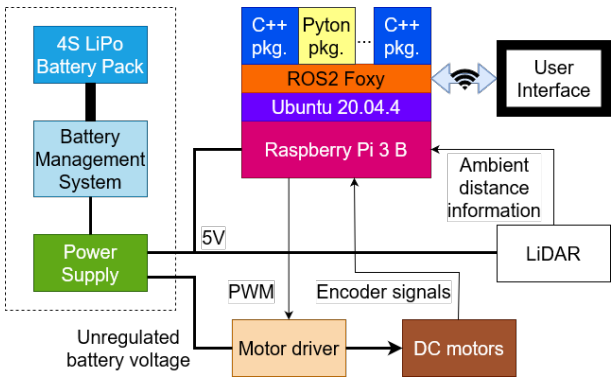

Fig. B.1 Structure of the mobile agent

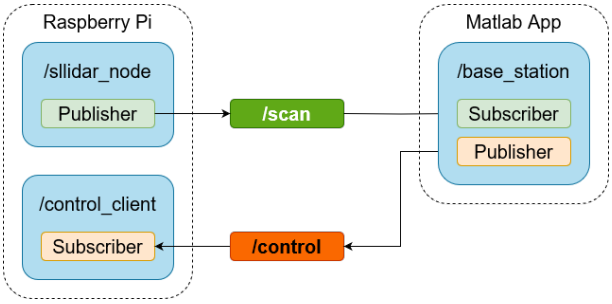

Fig. B.2 Connection of the ROS2 nodes

subscribers. It can be observed that the orientations are mapped to the $(-\pi;\pi]$ rad range and stored in a separate array along with the corresponding distance values. The node called "*base_station*" subscribes to the scan topic and is created under the MATLAB-based user interface. The message received from the scan topic contains the minimum and maximum angles, the size of the increments between them, and the distance values. Each distance value corresponds to a specific angle. Using this data, elementary mathematical operations can be used to generate Cartesian coordinates in the Descartes coordinate system. Due to the placement of the laser scanner on the robot, corrections, rotations, and reflections were required to ensure that the resulting image has the appropriate orientation in the coordinate system used in the application, where the forward direction corresponds to the positive direction of the y-axis. The "*base_station*" node sends pulse width values for controlling the motors to the "*control_client*" node through the "control" topic. I transmitted the published data in the "*std_msgs/UInt8MultiArray*" format. The MultiArray message type is quite flexible, allowing the developer to freely determine the type and amount of data contained in the homogeneous array. In this case, the data package contains an eight-element, unsigned, eight-bit integer array, specifying the movement of the three motors in different directions. The "*control_client*" node receives control signals from the user interface and forwards them to the motor controllers through the Raspberry Pi GPIO connectors. Iused the PIGPIO library to access the GPIO pins. It was important to ensure that if the connection between the robot and the controlling computer is interrupted due to any malfunction, the system should stop quickly, within 200ms in this case. I achieved this using software-based timed interrupts. Launch files assist in the configuration and launch of the nodes. These files can be in Python, XML, or YAML format. The configuration file was created for this project based on the project launch files published by Slamtec mentioned earlier. In this file, I parameterized the "*sllidar_node*" to send the LiDAR data to the scan topic via UDP port, and Ialso initialized the "*control_client*" node.

References

[1] Francisco J Perez-Grau, J Ramiro Martinez-de Dios, Julio L Paneque, J Joaquin Acevedo, Arturo Torres-González, Antidio Viguria, Juan R Astorga, and Anibal Ollero. Introducing autonomous aerial robots in industrial manufacturing. *Journal of Manufacturing Systems*, 60:312–324, 2021.

[2] Mohd Javaid, Abid Haleem, Ravi Pratap Singh, and Rajiv Suman. Substantial capabilities of robotics in enhancing industry 4.0 implementation. *Cognitive Robotics*, 1:58–75, 2021.

[3] Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. A survey of robots in healthcare. *Technologies*, 9(1):8, 2021.

[4] Jane Holland, Liz Kingston, Conor McCarthy, Eddie Armstrong, Peter O'Dwyer, Fionn Merz, and Mark McConnell. Service robots in the healthcare sector. *Robotics*, 10(1):47, 2021.

[5] Shaohua Wan, Zonghua Gu, and Qiang Ni. Cognitive computing and wireless communications on the edge for healthcare service robots. *Computer Communications*, 149:99–106, 2020.

[6] Baichang Zhong and Liying Xia. A systematic review on exploring the potential of educational robotics in mathematics education. *International Journal of Science and Mathematics Education*, 18:79–101, 2020.

[7] Ashraf Alam. Educational robotics and computer programming in early childhood education: a conceptual framework for assessing elementary school students' computational thinking for designing powerful educational scenarios. In *2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN)*, pages 1–7, 2022.

[8] Lina Van Aerschot and Jaana Parviainen. Robots responding to care needs? a multitasking care robot pursued for 25 years, available products offer simple entertainment and instrumental assistance. *Ethics and Information Technology*, 22(3):247–256, 2020.

[9] Laurens De Gauquier, Malaika Brengman, Kim Willems, Hoang-Long Cao, and Bram Vanderborght. In or out? a field observational study on the placement of entertaining robots in retailing. *International Journal of Retail & Distribution Management*, 49(7):846–874, 2021.

[10] Lihui Wang, Robert Gao, József Váncza, Jörg Krüger, Xi Vincent Wang, Sotiris Makris, and George Chryssolouris. Symbiotic human-robot collaborative assembly. *CIRP annals*, 68(2):701–726, 2019.

[11] Mohammad Hosein Hadian, Melika Salehian, Iman Sharifi, and Mohammad Bagher Menhaj. Utilizing function approximation technique and neural network controllers on a 2-dof painter robot. In *2022 10th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 202–208, 2022.

[12] Lei Yang, Yanhong Liu, and Jinzhu Peng. Advances techniques of the structured light sensing in intelligent welding robots: a review. *The International Journal of Advanced Manufacturing Technology*, 110:1027–1046, 2020.

[13] Tresna Dewi, Pola Risma, and Yurni Oktarina. Fruit sorting robot based on color and size for an agricultural product packaging system. *Bulletin of Electrical Engineering and Informatics*, 9(4):1438–1445, 2020.

[14] Jesse Fox and Andrew Gambino. Relationship development with humanoid social robots: Applying interpersonal theories to human–robot interaction. *Cyberpsychology, Behavior, and Social Networking*, 24(5):294–299, 2021.

[15] Michelle ME Van Pinxteren, Ruud WH Wetzels, Jessica Rüger, Mark Pluymaekers, and Martin Wetzels. Trust in humanoid robots: implications for services marketing. *Journal of Services Marketing*, 33(4):507–518, 2019.

[16] Andre Luiz Gioia Morrell, Alexander Charles Morrell-Junior, Allan Gioia Morrell, Jose Mendes, Mauritio Freitas, Francesco Tustumi, Alexander Morrell, et al. The history of robotic surgery and its evolution: when illusion becomes reality. *Revista do Colégio Brasileiro de Cirurgiões*, 48:e20202798, 2021.

[17] Hassan M Qassim and WZ Wan Hasan. A review on upper limb rehabilitation robots. *Applied Sciences*, 10(19):6976, 2020.

[18] Atheer Awad, Sarah J Trenfield, Thomas D Pollard, Jun Jie Ong, Moe Elbadawi, Laura E McCoubrey, Alvaro Goyanes, Simon Gaisford, and Abdul W Basit. Connected healthcare: Improving patient care using digital health technologies. *Advanced Drug Delivery Reviews*, 178:113958, 2021.

[19] Songsong Tang, Fangyu Zhang, Hua Gong, Fanan Wei, Jia Zhuang, Emil Karshalev, Berta Esteban-Fernández de Ávila, Chuying Huang, Zhidong Zhou, Zhengxing Li, et al. Enzyme-powered janus platelet cell robots for active and targeted drug delivery. *Science Robotics*, 5(43):eaba6137, 2020.

[20] Asher Elmquist, Radu Serban, and Dan Negrut. A sensor simulation framework for training and testing robots and autonomous vehicles. *Journal of Autonomous Vehicles and Systems*, 1(2):021001, 2021.

[21] Deepak Patil, Munsaf Ansari, Dilisha Tendulkar, Ritesh Bhatlekar, Vijaykumar Naik Pawar, and Shailendra Aswale. A survey on autonomous military service robot. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–7, 2020.

[22] Vinh Nhat Lu, Jochen Wirtz, Werner H Kunz, Stefanie Paluch, Thorsten Gruber, Antje Martins, and Paul G Patterson. Service robots, customers and service employees: what can we learn from the academic literature and where are the gaps? *Journal of Service Theory and Practice*, 30(3):361–391, 2020.

[23] Cheng Chen, Emrah Demir, Yuan Huang, and Rongzu Qiu. The adoption of self-driving delivery robots in last mile logistics. *Transportation research part E: logistics and transportation review*, 146:102214, 2021.

[24] Alex Junho Lee, Wonho Song, Byeongho Yu, Duckyu Choi, Christian Tirtawardhana, and Hyun Myung. Survey of robotics technologies for civil infrastructure inspection. *Journal of Infrastructure Intelligence and Resilience*, 2(1):100018, 2023.

[25] Toby Howison, Simon Hauser, Josie Hughes, and Fumiya Iida. Reality-assisted evolution of soft robots through large-scale physical experimentation: a review. *Artificial Life*, 26(4):484–506, 2020.

[26] Qin Yang and Ramviyas Parasuraman. Needs-driven heterogeneous multi-robot cooperation in rescue missions. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 252–259, 2020.

[27] Timothy A Vincent, Yuxin Xing, Marina Cole, and Julian W Gardner. Investigation of the response of high-bandwidth mox sensors to gas plumes for application on a mobile robot in hazardous environments. *Sensors and Actuators B: Chemical*, 279:351–360, 2019.

[28] Peng Li, Hongjiu Yang, Hongbo Li, and Shiqing Liang. Nonlinear eso-based tracking control for warehouse mobile robots with detachable loads. *Robotics and Autonomous Systems*, 149:103965, 2022.

[29] Kathrin Dörfler, Norman Hack, Timothy Sandy, Markus Giftthaler, Manuel Lussi, Alexander N Walzer, Jonas Buchli, Fabio Gramazio, and Matthias Kohler. Mobile robotic fabrication beyond factory conditions: Case study mesh mould wall of the dfab house. *Construction robotics*, 3:53–67, 2019.

[30] Anna Henschel, Guy Laban, and Emily S Cross. What makes a robot social? a review of social robots from science fiction to a home or hospital near you. *Current Robotics Reports*, 2:9–19, 2021.

[31] Abel Gawel, Hermann Blum, Johannes Pankert, Koen Krämer, Luca Bartolomei, Selen Ercan, Farbod Farshidian, Margarita Chli, Fabio Gramazio, Roland Siegwart, et al. A fully-integrated sensing and control system for high-accuracy mobile robotic building construction. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2300–2307, 2019.

[32] Jun Hu, Zidong Wang, Guo-Ping Liu, and Hongxu Zhang. Variance-constrained recursive state estimation for time-varying complex networks with quantized measurements and uncertain inner coupling. *IEEE Transactions on Neural Networks and Learning Systems*, 31(6):1955–1967, 2019.

[33] Ross Hartley, Maani Ghaffari, Ryan M Eustice, and Jessy W Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *The International Journal of Robotics Research*, 39(4):402–430, 2020.

[34] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Daniel Riordan, and Joseph Walsh. Adaptive multimodal localisation techniques for mobile robots in unstructured environments: A review. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 799–804, 2019.

[35] Mary B Alatise and Gerhard P Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.

[36] Jun Chen and Philip Dames. Collision-free distributed multi-target tracking using teams of mobile robots with localization uncertainty. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6968–6974, 2020.

[37] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.

[38] Julien Moreau, Pierre Melchior, Stéphane Victor, Louis Cassany, Mathieu Moze, François Aioun, and Franck Guillemard. Reactive path planning in intersection for autonomous vehicle. *IFAC-PapersOnLine*, 52(5):109–114, 2019.

[39] Mahmood Reza Azizi, Alireza Rastegarpanah, and Rustam Stolkin. Motion planning and control of an omnidirectional mobile robot in dynamic environments. *Robotics*, 10(1):48, 2021.

[40] Aisha Muhammad, Mohammed AH Ali, Sherzod Turaev, Ibrahim Haruna Shanono, Fadhl Hujainah, Mohd Nashrul Mohd Zubir, Muhammad Khairi Faiz, Erma Rahayu Mohd Faizal, and Rawad Abdulghafor. Novel algorithm for mobile robot path planning in constrained environment. *Comput Mater Contin*, 71(2):2697–2719, 2022.

[41] Lu Chang, Liang Shan, Chao Jiang, and Yuewei Dai. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Autonomous Robots*, 45:51–76, 2021.

[42] Saroj Kumar, Dayal R Parhi, Manoj Kumar Muni, and Krishna Kant Pandey. Optimal path search and control of mobile robot using hybridized sine-cosine algorithm and ant colony optimization technique. *Industrial Robot: the international journal of robotics research and application*, 47(4):535–545, 2020.

[43] Janko Petereit, Thomas Emter, Christian W Frey, Thomas Kopfstedt, and Andreas Beutel. Application of hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments. In *Proc. of ROBOTIK-12; 7th German Conference on Robotics*, pages 227–232, Munich, Germany, 2012 May 21-22 2012.

[44] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. Tech. Rep. TR 98-11, Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, 1998.

[45] Iram Noreen, Amna Khan, Hyejeong Ryu, Nakju Lett Doh, and Zulfiqar Habib. Optimal path planning in cluttered environment using RRT*-AB. *Intelligent Service Robotics*, 11(1):41–52, 2018.

[46] Yiqun Dong, Efe Camci, and Erdal Kayacan. Faster RRT-based nonholonomic path planning in 2D building environments using skeleton-constrained path biasing. *Journal of Intelligent & Robotic Systems*, 89(3):387–401, 2018.

[47] John H Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999.

[48] Guanghui Li, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. Effective improved artificial potential field-based regression search method for autonomous mobile robot path planning. *International Journal of Mechatronics and Automation*, 3(3):141–170, 2013.

[49] Bence Kovács, Géza Szayer, Ferenc Tajti, Mauricio Burdelis, and Péter Korondi. A novel potential field method for path planning of mobile robots by adapting animal motion attributes. *Robotics and Autonomous Systems*, 82:24–34, 2016.

[50] Cao Qixin, Huang Yanwen, and Zhou Jingliang. An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS-06)*, pages 3331–3336, 2006.

[51] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[52] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-99)*, volume 1, pages 341–346, 1999.

[53] Bin Wu, Xiaonan Chi, Congcong Zhao, Wei Zhang, Yi Lu, and Di Jiang. Dynamic path planning for forklift agv based on smoothing a* and improved dwa hybrid algorithm. *Sensors*, 22(18):7079, 2022.

[54] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.

[55] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. Generalized velocity obstacles. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS-09)*, pages 5573–5578, 2009.

[56] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS-10)*, pages 4584–4589, 2010.

[57] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-08)*, pages 1928–1935. IEEE, 2008.

[58] Yamin Huang, Linying Chen, and PHAJM Van Gelder. Generalized velocity obstacle algorithm for preventing ship collisions at sea. *Ocean Engineering*, 173:142–156, 2019.

[59] Thomas Battisti and Riccardo Muradore. A velocity obstacles approach for autonomous landing and teleoperated robots. *Autonomous Robots*, 44:217–232, 2020.

[60] Jur Van Den Berg, Jamie Snape, Stephen J Guy, and Dinesh Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-11)*, pages 3475–3482, 2011.

[61] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011.

[62] James A Douthwaite, Shiyu Zhao, and Lyudmila S Mihaylova. Velocity obstacle approaches for multi-agent collision avoidance. *Unmanned Systems*, 7(01):55–64, 2019.

[63] Federico Vesentini, Nicola Piccinelli, and Riccardo Muradore. Velocity obstacle-based trajectory planner for anthropomorphic arms. *European Journal of Control*, page 100901, 2023.

[64] Stefanie Manzinger, Christian Pek, and Matthias Althoff. Using reachable sets for trajectory planning of automated vehicles. *IEEE Transactions on Intelligent Vehicles*, 6(2):232–248, 2020.

[65] Javier Alonso-Mora, Paul Beardsley, and Roland Siegwart. Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics*, 34(2):404–420, 2018.

[66] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[67] Thierry Fraichard and Hajime Asama. Inevitable collision states - a step towards safer robots. *Advanced Robotics*, 18:1001–1024, 2004.

[68] Luis Martinez-Gomez and Thierry Fraichard. Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-09)*, pages 100–105, May 2009.

[69] Jianhua Li, Jianfeng Sun, Liqun Liu, and Jiasheng Xu. Model predictive control for the tracking of autonomous mobile robot combined with a local path planning. *Measurement and Control*, 54(9-10):1319–1325, 2021.

[70] Damion D Dunlap, Charmane V Caldwell, Emmanuel G Collins, Oscar Chuy, et al. Motion planning for mobile robots via sampling-based model predictive optimization. *Recent advances in mobile robotics*, 1, 2011.

[71] Narges Eslami and Roya Amiadifard. Moving target tracking and obstacle avoidance for a mobile robot using mpc. In *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, pages 1163–1169, 2019.

[72] Puyong Xu, Ning Wang, Shi-Lu Dai, and Lei Zuo. Motion planning for mobile robot with modified bit* and mpc. *Applied Sciences*, 11(1):426, 2021.

[73] Chang Liu and J Karl Hedrick. Model predictive control-based target search and tracking using autonomous mobile robot with limited sensing domain. In *2017 American control conference (ACC)*, pages 2937–2942, 2017.

[74] I. Ashiru and C. Czarnecki. Optimal motion planning for mobile robots using genetic algorithms. In *Proc. of Intl. Conf. on Industrial Automation and Control*, pages 297–300, 1995.

[75] Maram Alajlan, Anis Koubaa, Imen Chaari, Hachemi Bennaceur, and Adel Ammar. Global path planning for mobile robots in large-scale grid environments using genetic algorithms. In *Proc. of Intl. Conf. on Individual and Collective Behaviors in Robotics (ICBR-13)*, pages 1–8, 2013.

[76] Robert Martin C Santiago, Anton Louise De Ocampo, Aristotle T Ubando, Argel A Bandala, and Elmer P Dadios. Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In *Proc. of IEEE Intl. Conf. on Humanoid, Nanotechnology, Information Technology, communication and control, environment and management (HNICEM-17)*, pages 1–5, 2017.

[77] Yang Xue. Mobile robot path planning with a non-dominated sorting genetic algorithm. *Applied Sciences*, 8(11), 2018.

[78] Rajat Kumar Panda and BB Choudhury. An effective path planning of mobile robot using genetic algorithm. In *Proc. of IEEE Intl. Conf. on Computational Intelligence & Communication Technology*, pages 287–291, 2015.

[79] Jesus Savage, Stalin Munoz, Mauricio Matamoros, and Roman Osorio. Obstacle avoidance behaviors for mobile robots using genetic algorithms and recurrent neural networks. *IFAC Proceedings Volumes*, 46(24):141–146, 2013.

[80] Adem Tuncer and Mehmet Yildirim. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38(6):1564–1572, 2012.

[81] Qing Li, Wei Zhang, Yixin Yin, Zhiliang Wang, and Guangjun Liu. An improved genetic algorithm of optimum path planning for mobile robots. In *Proc. of Intl. Conf. on Intelligent Systems Design and Applications*, volume 2, pages 637–642, 2006.

[82] Ryo Terasawa, Yuka Ariki, Takuya Narihira, Toshimitsu Tsuboi, and Kenichiro Nagasaka. 3d-cnn based heuristic guided task-space planner for faster motion planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9548–9554, 2020.

[83] Hiroki Kanayama, Taishi Ueda, Hiroshi Ito, and Kenjiro Yamamoto. Two-mode mapless visual navigation of indoor autonomous mobile robot using deep convolutional neural network. In *2020 IEEE/SICE International Symposium on System Integration (SII)*, pages 536–541, 2020.

[84] M Saravanan, P Satheesh Kumar, and Amit Sharma. Iot enabled indoor autonomous mobile robot using cnn and q-learning. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 7–13, 2019.

[85] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[86] Leonid Butyrev, Thorsten Edelhäußer, and Christopher Mutschler. Deep reinforcement learning for motion planning of mobile robots. *arXiv preprint arXiv:1912.09260*, 2019.

[87] Huihui Sun, Weijie Zhang, Runxiang Yu, and Yujie Zhang. Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review. *IEEE Access*, 9:69061–69081, 2021.

[88] Junli Gao, Weijie Ye, Jing Guo, and Zhongjuan Li. Deep reinforcement learning for indoor mobile robot path planning. *Sensors*, 20(19):5493, 2020.

[89] Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022.

[90] Bharath Gopalakrishnan, Arun Kumar Singh, Meha Kaushik, K Madhava Krishna, and Dinesh Manocha. Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS-17)*, pages 1089–1096, 2017.

[91] Aleksandar Timcenko and Peter Allen. Probability-driven motion planning for mobile robots. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 2784–2789, 1994.

[92] Luís BP Nascimento, Dennis Barrios-Aranibar, Vitor G Santos, Diego S Pereira, William C Ribeiro, and Pablo J Alsina. Safe path planning algorithms for mobile robots based on probabilistic foam. *Sensors*, 21(12):4156, 2021.

[93] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *Proc. of American Control Conference*, pages 7–pp, 2006.

[94] Simon Thompson, Takehiro Horiuchi, and Satoshi Kagami. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *Proc. of 4th International Conference on Autonomous Robots and Agents*, pages 663–668, 2009.

[95] Stergios I Roumeliotis and George A Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-00))*, volume 3, pages 2985–2992, 2000.

[96] Hamzah Ahmad and Toru Namerikawa. Extended kalman filter-based mobile robot localization with intermittent measurements. *Systems Science & Control Engineering: An Open Access Journal*, 1(1):113–126, 2013.

[97] SY Chen. Kalman filter for robot vision: a survey. *IEEE Transactions on industrial electronics*, 59(11):4409–4420, 2011.

[98] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*, 82:35–45, 1960.

[99] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Journal of Fluids Engineering, Transactions of the ASME*, 83:95–108, 1961.

[100] Neil Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear and linear bayesian state estimation, 1993.

[101] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. Methods for bayesian filtering. *Statistics and Computing*, pages 197–208, 2000.

[102] Zhiwen Zhong, Huadong Meng, and Xiqin Wang. Extended target tracking using an IMM based rao-blackwellised unscented kalman filter. *Proc. of Intl. Conf. on Signal Processing Proceedings (ICSP-08)*, pages 2409–2412, 2008.

[103] Daniel Claes, Daniel Hennes, Karl Tuyls, and Wim Meeussen. Collision avoidance under bounded localization uncertainty. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (IROS-12)*, pages 1192–1198, 2012.

[104] Pieter M Blok, Koen van Boheemen, Frits K van Evert, Joris IJsselmuiden, and Gook-Hwan Kim. Robot navigation in orchards with localization based on particle filter and kalman filter. *Computers and electronics in agriculture*, 157:261–269, 2019.

[105] Leopoldo Jetto, Sauro Longhi, and Giuseppe Venturini. Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 15(2):219–229, 1999.

[106] Evgeni Kiriy and Martin Buehler. Three-state extended kalman filter for mobile robot localization. *McGill University., Montreal, Canada, Tech. Rep. TR-CIM*, 5:23, 2002.

[107] Ling Chen, Huosheng Hu, and Klaus McDonald-Maier. Ekf based mobile robot localization. In *Proc. of Intl. Conf. on Emerging Security Technologies*, pages 149–154, 2012.

[108] Jakub Simanek, Michal Reinstein, and Vladimir Kubelka. Evaluation of the ekf-based estimation architectures for data fusion in mobile robots. *IEEE/ASME Transactions on Mechatronics*, 20(2):985–990, 2014.

[109] Luka Teslić, Igor Škrjanc, and Gregor Klančar. Ekf-based localization of a wheeled mobile robot in structured environments. *Journal of Intelligent & Robotic Systems*, 62:187–203, 2011.

[110] Luigi D'Alfonso, Walter Lucia, Pietro Muraca, and Paolo Pugliese. Mobile robot localization via ekf and ukf: A comparison based on real data. *Robotics and Autonomous Systems*, 74:122–127, 2015.

[111] Francesco Martinelli. Robot localization: comparable performance of EKF and UKF in some interesting indoor settings. In *Proc. IEEE of Mediterranean Conference on Control and Automation*, pages 499–504, 2008.

[112] Prashant G Medewar, Manish Yadav, and Hirenkumar G Patel. A comparison between nonlinear estimation based algorithms for mobile robot localizations. In *Proc. of IEEE Intl. Conf. on 1st International Conference on Energy, Systems and Information Processing (ICESIP-19)*, pages 1–6, 2019.

[113] Nak Yong Ko and Tae-Yong Kuc. Fusing range measurements from ultrasonic beacons and a laser range finder for localization of a mobile robot. *Sensors*, 15(5):11050–11075, 2015.

[114] Marshall N. Rosenbluth and Arianna W. Rosenbluth. Monte carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, 23(2):356–359, 1955.

[115] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25:53–81, 2010.

[116] Jur Van Den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. *Springer Tracts in Advanced Robotics*, 70:3–19, 2011.

[117] Yongbo Chen, Shoudong Huang, and Robert Fitch. Active slam for mobile robots with area coverage and obstacle avoidance. *IEEE/ASME Transactions on Mechatronics*, 25(3):1182–1192, 2020.

[118] Supun Kamburugamuve, Hengjing He, Geoffrey Fox, and David Crandall. Cloud-based parallel implementation of slam for mobile robots. In *Proc. of IEEE Intl. Conf. on Internet of things and Cloud Computing*, pages 1–7, 2016.

[119] Yi Kiat Tee and Yi Chiew Han. Lidar-based 2d slam for mobile robot in an indoor environment: A review. In *Proc. of IEEE Intl. Conf. on Green Energy, Computing and Sustainable Technology (GECOST-21)*, pages 1–7, 2021.

[120] Abdullah Yusefi, Akif Durdu, Muhammet Fatih Aslan, and Cemil Sungur. Lstm and filter based comparison analysis for indoor global localization in uavs. *IEEE Access*, 9:10054–10069, 2021.

[121] Feng Zhang, Siqi Li, Shuai Yuan, Enze Sun, and Languang Zhao. Algorithms analysis of mobile robot slam based on kalman and particle filter. In *Proc. of IEEE Intl. Conf. on Modelling, Identification and Control (ICMIC-17)*, pages 1050–1055, 2017.

[122] David Törnqvist, Thomas B Schön, Rickard Karlsson, and Fredrik Gustafsson. Particle filter slam with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 55:249–266, 2009.

[123] Daniel Hennes, Daniel Claes, Wim Meeussen, and Karl Tuyls. Multi-robot collision avoidance with localization uncertainty. In *Proc. of Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-12)*, volume 2, pages 672–679, 2012.

[124] SLAMTEC. Rplidar a2, 2016. Accessed: 09.09.2023.

[125] Hao-Tien Chiang, Nick Malone, Kendra Lesser, Meeko Oishi, and Lydia Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *Proc. of IEEE Intl. Conf. on Robotics and Automation (ICRA-15)*, pages 2347–2354, 2015.

[126] Bence Kovács, Géza Szayer, Ferenc Tajti, Mauricio Burdelis, and Péter Korondi. A novel potential field method for path planning of mobile robots by adapting animal motion attributes. *Robotics and Autonomous Systems*, 82:24–34, 2016.

[127] Yong Yuan, You Shi, Song Yue, Shanliang Xue, Changyan Yi, and Bing Chen. A dynamic obstacle avoidance method for agv based on improved speed barriers. *Electronics*, 11(24), 2022.

[128] James E Baker. Reducing bias and inefficiency in the selection algorithm. In *Proc. of Second International Conference on Genetic Algorithms*, volume 206, pages 14–21, 1987.

[129] Jia Yuan Zhuang, Yu Min Su, Yu Lei Liao, and Han Bing Sun. Motion planning of USV based on marine rules. *Procedia Engineering*, 15:269–276, 2011.

[130] Yoshiaki Kuwata, Michael T. Wolf, Dimitri Zarzhitsky, and Terrance L. Huntsberger. Safe maritime autonomous navigation with colregs, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39:110–119, 2014.

[131] Helmut Pottmann and Gerald Farin. Developable rational b'ezier and b-spline surfaces. *Computer Aided Geometric Design*, 12:513–531, 1995.

[132] Xiaoyun Huang, Feng Gao, Guoyan Xu, Nenggen Ding, Liujun Li, and Yao Cai. Extended-search, b'ezier curve-based lane detection and reconstruction system for an intelligent vehicle. *International Journal of Advanced Robotic Systems*, 40:1459–1470, 2016.

[133] Heidi Loose and Uwe Franke. B-spline-based road model for 3d lane recognition. In *Proc. of IEEE Intl. Conf. on Intelligent Transportation Systems ITSC*, pages 91–98, 2010.

[134] Yue Wang, Dinggang Shen, and Eam Khwang Teoh. Lane detection using spline model. *Pattern Recognition Letters*, 21:677–689, 2000.

[135] CoppeliaSim. Coppeliasim. https://www.coppeliarobotics.com/.

[136] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4):1135–1145, 2015.

[137] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on control systems technology*, 17(5):1105–1118, 2009.

[138] Ellips Masehian and Yalda Katebi. Robot motion planning in dynamic environments with moving obstacles and target. *International Journal of Mechanical Systems Science and Engineering*, 1(1):20–25, 2007.

[139] Andres Vasquez, Marina Kollmitz, Andreas Eitel, and Wolfram Burgard. Deep detection of people and their mobility aids for a hospital robot. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2017.

[140] Amari Tomoya, Satoru Nakayama, Atsushi Hoshina, and Midori Sugaya. A mobile robot for following, watching and detecting falls for elderly care. *Procedia computer science*, 112:1994–2003, 2017.

[141] Ioannis Exarchos, Panagiotis Tsiotras, and Meir Pachter. UAV collision avoidance based on the solution of the suicidal pedestrian differential game. In *Proc. of AIAA Guidance, Navigation, and Control Conf.*, Jan. 2016.

[142] Nick Malone, Hao-Tien Chiang, Kendra Lesser, Meeko Oishi, and Lydia Tapia. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Transactions on Robotics*, 33(5):1124–1138, Oct. 2017.

[143] Sarah Bonnin, Thomas H Weisswange, Franz Kummert, and Jens Schmüdderich. Pedestrian crossing prediction using multiple context-based models. In *Proc. of IEEE 17th Intl. Conf. on Intelligent Transportation Systems (ITSC-14)*, pages 378–385, 2014.

[144] Tufail Ahmed, Mehdi Moeinaddini, Meshal Almoshaogeh, Arshad Jamal, Imran Nawaz, and Fawaz Alharbi. A new pedestrian crossing level of service (PCLOS) method for promoting safe pedestrian crossing in urban areas. *International Journal of Environmental Research and Public Health*, 18(16):8813, 2021.

[145] T.S. Bhatia, G. Solmaz, D. Turgut, and L. Bölöni. Two algorithms for the movements of robotic bodyguard teams. In *Proc. of Workshop on Knowledge, Skill, and Behavior Transfer in Autonomous Robots*, pages 2–8, Jan. 2015.

[146] Donggeun Cha and Woojin Chung. Human-leg detection in 3d feature space for a person-following mobile robot using 2d lidars. *International Journal of Precision Engineering and Manufacturing*, 21:1299–1307, 2020.

[147] Tapio Taipalus and Juhana Ahtiainen. Human detection and tracking with knee-high mobile 2d lidar. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 1672–1677. IEEE, 2011.

[148] Jiannan Chen, Ping Ye, and Zhipeng Sun. Pedestrian detection and tracking based on 2d lidar. In *2019 6th International Conference on Systems and Informatics (ICSAI)*, pages 421–426. IEEE, 2019.

[149] Randy Bullock. Least-squares circle fit. *Developmental testbed center*, 3, 2006.

[150] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

Own Journal Papers

[J1] Emese Gincsainé Szádeczky-Kardoss and Zoltán Gyenes. Velocity obstacles for car-like mobile robots: Determination of colliding velocity and curvature pairs. *Advances in Science, Technology and Engineering Systems Journal*, 3(1):225–233, 2018.

[J2] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Novel motion planning method for mobile robots using velocity obstacle. *Acta Polytechnica Hungarica, Q2, IF: 1.806*, 17(9):221–240, 2020.

[J3] Zoltán Gyenes, Ladislau Bölöni, and Emese Gincsainé Szádeczky-Kardoss. Can genetic algorithms be used for real-time obstacle avoidance for lidar-equipped mobile robots? *Sensors, Q2, IF: 3.846*, 23(6):3039, 2023.

[J4] Zoltán Bálint Gyenes and Emese Gincsainé Szádeczky-Kardoss. Uncertain estimation-based motion-planning algorithms for mobile robots. *Acta IMEKO, Q3*, 10(3):51–60, 2021.

[J5] Zoltán Gyenes, Ladislau Bölöni, and Emese Gincsainé Szádeczky-Kardoss. Exploring the use of particle and kalman filters for obstacle detection in mobile robots. *Periodica Polytechnica Electrical Engineering and Computer Science, Q3*, 67:384–393, 2023.

Own Conference Papers

[C1] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Motion planning for mobile robots using the safety velocity obstacles method. In *Proc. of 19th IEEE International Carpathian Control Conference (ICCC)*, pages 389–394, 2018.

[C2] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Mobilis robotok mozgástervezése az SVO és APF módszerekkel. In *Proc. of Tavaszi Szél 2019 Konferencia = Spring Wind 2019: Konferenciakötet III.*, pages 42–55, 2019.

[C3] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Velocity selection for mobile robots using different strategies and probability. In *Proc. of the Workshop on the Advances of Information Technology (WAIT)*, pages 142–147, 2020.

[C4] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Rule-based velocity selection for mobile robots under uncertainties. In *Proc. of 24th IEEE International Conference on Intelligent Engineering Systems (INES)*, pages 127–132, 2020.

[C5] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Motion planning for mobile robots using uncertain estimations about the environment. In *Proc. of 23rd IEEE International Symposium on Measurement and Control in Robotics (ISMCR)*, pages 1–6, 2020.

[C6] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Particle filter based obstacle's position estimation using lidar measurement data. In *Proc. of the Workshop on the Advances of Information Technology (WAIT)*, pages 97–102, 2021.

[C7] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Particle filter-based perception method for obstacles in dynamic environment of a mobile robot. In *Proc. of 25th IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 97–102, 2021.

[C8] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Perception of obstacles at mobile robot's motion planning algorithm using kalman filter and particle filter. In *Proc. of the Workshop on the Advances of Information Technology (WAIT)*, pages 86–91, 2023.

[C9] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Motion planning for mobile robots using traffic regulation. In *Proc. of the Workshop on the Advances of Information Technology (WAIT)*, pages 92–97, 2019.

[C10] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Traffic regulation velocity obstacles method. In *Proc. of 20th IEEE International Carpathian Control Conference (ICCC)*, Paper: 8766055 , 6 p., 2019.

[C11] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. A novel concept of lane-keeping algorithms for mobile robots. In *Proc. of 17th IEEE International Symposium on Intelligent Systems and Informatics (SISY)*, pages 47–52, 2019.

[C12] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Collidible velocity obstacles method. In *Proc. of the Workshop on the Advances of Information Technology (WAIT)*, pages 134–139, 2022.

[C13] Zoltán Gyenes and Emese Gincsainé Szádeczky-Kardoss. Motion planning for mobile robots using the collidable velocity obstacles method. In *Proc. of 30th IEEE Mediterranean Conference on Control and Automation (MED)*, pages 737–742, 2022.

[C14] Zoltán Gyenes, Ilshat Mamaev, Dongxu Yang, E Gincsainé Szádeczky-Kardoss, and Björn Hein. Motion planning for mobile robots using the human tracking velocity obstacles method. In *Proc. of 19th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 484–491, 2022.

Own Papers in progress

[IP1] János Szőts, Zoltán Gyenes, Ladislau Bölöni, Emese Gincsainé Szádeczky-Kardoss, and István Harmati. The emergency braking game - a game theoretic approach for manuevering in a dense crowd of pedestrians (accepted). *Robomech, Q3*, 2023.

[IP2] Zoltán Gyenes, Barnabás Pajkos, Ladislau Bölöni, and Emese Gincsainé Szádeczky-Kardoss. Motion planning for mobile robots using uncertain obstacle's estimation considering lidar sensor data (submitted). *IEEE Access, Q1*, 2023.

Youtube videos

[YO1] Zoltan Gyenes. Different value safety velocity obstacle method. https://www.youtube.com/watch?v=22NYgpzjPOk&feature=youtu.be.

[YO2] Zoltan Gyenes. Fastest solution. https://youtu.be/86qM-XcqZJo.

[YO3] Zoltan Gyenes. Test 2 in coppeliasim. https://youtu.be/Mg0b9ZqB0go.

[YO4] Zoltan Gyenes. Collidable velocity obstacles method, 1st example. https://youtu.be/5frL38eGyCY, 2022.

[YO5] Zoltan Gyenes. Emergency braking, 1st example. https://youtu.be/j-quD1ENXmM, 2022.

[YO6] Zoltan Gyenes. Collidable velocity obstacles method, 2nd example. https://youtu.be/8FIyLHnqdEc, 2022.

[YO7] Zoltan Gyenes. Emergency braking, 1st example. https://youtu.be/aio_9q5daWY, 2022.

[YO8] Zoltan Gyenes. Human tracking velocity obstacles method 1. https://www.youtube.com/watch?v=SKNefY399Co&ab_channel=Zolt%C3%A1nGyenes, 2022.

[YO9] Zoltan Gyenes. Human tracking velocity obstacles method 2. https://www.youtube.com/watch?v=FZFnf7xBhAk&ab_channel=Zolt%C3%A1nGyenes, 2022.

[YO10] Zoltan Gyenes. Human tracking velocity obstacles method 3. https://www.youtube.com/watch?v=1xABzIMZ3uc&ab_channel=Zolt%C3%A1nGyenes, 2022.