sensors

MDPI

Article

Coordinated Multi-Robotic Vehicles Navigation and Control in Shop Floor Automation

Gregor Klančar [1,*] and Marija Seder [2]

[1] Faculty of Electrical Engineering, University of Ljubljana , Tržaška 25, SI-1000 Ljubljana, Slovenia
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia; marija.seder@fer.hr
* Correspondence: gregor.klancar@fe.uni-lj.si

Abstract: In this paper, we propose a global navigation function applied to model predictive control (MPC) for autonomous mobile robots, with application to warehouse automation. The approach considers static and dynamic obstacles and generates smooth, collision-free trajectories. The navigation function is based on a potential field derived from an E* graph search algorithm on a discrete occupancy grid and by bicubic interpolation. It has convergent behavior from anywhere to the target and is computed in advance to increase computational efficiency. The novel optimization strategy used in MPC combines a discrete set of velocity candidates with randomly perturbed candidates from particle swarm optimization. Adaptive horizon length is used to improve performance. The efficiency of the proposed approaches is validated using simulations and experimental results.

Keywords: navigation; model predictive control; path planing; mobile robots; warehouse automation

Citation: Klančar, G.; Seder, M. Coordinated Multi-Robotic Vehicles Navigation and Control in Shop Floor Automation. *Sensors* **2022**, *22*, 1455. https://doi.org/10.3390/s22041455

Academic Editor: Jesús Ureña

Received: 16 January 2022
Accepted: 31 January 2022
Published: 14 February 2022

1. Introduction

Mobile robotic platforms have found numerous applications over the past decade, a significant portion of which can be attributed to intralogistics and transportation in modern manufacturing, warehousing, and utility markets [1–3]. Although numerous sites have been automated by either automated guided vehicles (AGV) or autonomous mobile robots (AMR) with impressive deployments, the market is expected to grow by about 30% over the next five years [4,5].

Since the first AGV was built in 1953 [3], AGVs have evolved into today's solution, which is the standard in automating internal logistics. Typically, AGVs move along predefined paths and can only deliver to fixed points along the path. This makes these transportation systems simpler and more robust. Since movement is limited to fixed paths, the complexity of path planning and coordinating multiple AGVs is reduced. Nevertheless, planning collision-proof safe paths for a group of AGVs and creating and optimizing schedules to achieve better performance (higher throughput and less likely occurrence of conflicts) remains a challenging task [6]. Path planning is usually solved using graph-based search algorithms such as A*-based search, where optimal approaches [7,8] are feasible for a smaller number of vehicles since the computational complexity is exponential with the number of vehicles. The coordination overhead in multi-AGV systems is further reduced by suboptimal approaches, where the problem is decoupled from finding individual vehicles and conflicts are resolved by assigning traffic rules, priorities, or distributed multi-agent negotiations [6,9,10].

AMRs (unlike AGVs) are more flexible (in terms of their navigation capabilities and the services they can provide) and can move freely in dynamic environments where they locate, navigate, and act autonomously [4]. Free space is mapped based on knowledge of static obstacles, and dynamic obstacles are avoided using sensors. Since movement is not restricted to predefined paths but is possible in the continuum of obstacle-free space, the complexity of path planning must be reduced. A common approach is to discretize the environment into cells of equal size and use grid-based path planning [11,12]. Since

pathfinding usually examines only 4 or 8 neighbourhood directions, the paths obtained are not smooth.

Another approach is to apply a discrete set of motion primitives or actions that a vehicle can apply to advance to new locations. The motion primitives can be Bezier curves [13,14], clothoids [15], or other smooth curvature curves [16–18]. This usually results in smoother paths that a vehicle can easily follow. Different optimization strategies can be used to select suitable motion primitives. In high-dimensional spaces, randomised planners such as the Rapid Exploring Random Tree (RRT*) and the kinodynamic RRT* are popular choices [3,19]. A state lattice graph can be constructed from a discrete set of motion primitives that have smooth curvature transitions in the joints [20,21]. Graph search algorithms such as A* [22,23], D* [11,24], and E* [25] can be used to find the final path. The computational complexity can be addressed by hybrid search approaches such as Hybrid A* or HE * [14,26], where a computationally efficient discrete graph-based search is applied to obtain the heuristics for more efficient construction and search of the lattice graph, where the motion primitives form the edges of the graph.

Potential field-planning methods are also popular, where the potential function for online navigation can be used to guide the search or control algorithm. The goal with minimum potential value can be achieved by simply following the direction of the steepest descent of the potential field. A common problem of the potential field is local minima in which the robot may be trapped. Several approaches have been proposed to avoid local minima. Concave obstacles can be simply modelled as convex [27] in the environment map, or an adaptive potential field can be generated using multiple points of attraction instead of just one in the target [28]. It is also possible to modify the potential field in unstable equilibrium by introducing perturbations into the field [29] or adding virtual obstacles to repel the robot from the local minima [30]. The potential field can also be interpolated from a discrete cost map obtained from an optimal grid-based search [31,32]. Relying only on the reactive behavior of a potential field may result in unwanted oscillations in the presence of obstacles where alternating repulsive and attractive fields may cause approaching and moving-away behavior [33]. Therefore, prediction capabilities are needed to achieve more deliberative actions where planning and control are combined in receding dynamic window approaches or trajectory roll-out algorithms considering convergent navigation function, such as in [31,34–36]. Here, the obtained performance depends on a control law and uses an objective function, which needs to incorporate mapped static obstacles and sensed dynamic obstacles to find feasible optimal trajectories in a prediction horizon.

Moving obstacle avoidance is required for efficient multiple vehicle navigation. Coordinated motion of multiple vehicles can be dealled by assigning traffic rules in decentralised manner as in [10] or by combining a centralized supervisor, which detects collisions and assignes priorities for decentralised planner and scheduling for collision avoidance [37]. The decentralised decoupled approach is proposed in [18], where vehicles first plan optimal paths independently; then, conflict resolution is performed based on a priority scheme. In [38], a model predictive scheme is proposed where local deviations from the existing reference path are optimized considering collision avoidance with static and moving obstacles. In [39], an integration of the focused D* graph search algorithm for path planning and the dynamic window algorithm for generating admissible robot trajectories around the planned global path is proposed.

In this paper, the main contributions are the following. We propose a global navigation function applied in model predictive control to safely navigate the vehicle to the goal destination. The navigation function depends on a potential field for the environment layout and the driving direction. The potential function for a known target is computed in advance by an E* graph search algorithm on a discrete rectangular grid. A smooth surface with arbitrary potential values and slope directions is obtained by bicubic interpolation. It allows navigation from any location to the target and can be precomputed for any known target to which AMR must deliver.

Constrained Model Predictive Control (MPC) is a method of finding optimal trajectories given the proposed navigation function and constraints on robot kinematics, maximum veloc-

ities and accelerations, convergent behavior, and the coordination of multiple robots. MPC combines local motion planning and control in the presence of static and dynamic obstacles.

Adaptive horizon length is introduced in MPC to improve performance in terms of safety and achieved curve optimality.

A novel optimization strategy for MPC is proposed that combines a discrete set of command velocities proposed in [36] with randomly perturbed particle swarm optimization candidates. This approach extends the navigation of a single robot [32] to multiple robots.

Coordinated navigation in the presence of multiple vehicles is obtained locally as a constraint in the MPC objective function. The approach assumes that cooperative vehicles share their planned trajectories within the prediction horizon. For non-cooperative objects, the motion trajectories must be estimated from measurements.

The performance of the proposed approaches is illustrated by several examples.

2. Vehicle Autonomous Navigation and Control

For an example of a simple production layout, see Figure 1, where robots typically need to transport material between known, fixed locations. Suppose a destination is the dropoff point shown in Figure 1, to which robots must deliver material from several other locations, such as pickup point, workstations, and storage aria. A navigation function can be created to guide the robot safely from any starting point in the environment to the desired destination.

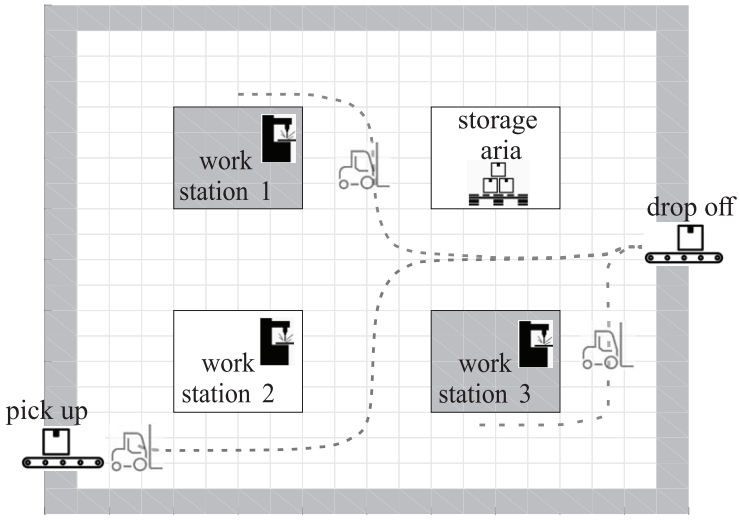

Figure 1. Production layout with multiple known and fixed delivery points and defined free corridors. Shown are robot delivery paths from three starting locations to the same destination. The paths can be effectively determined by a single navigation function, as shown in Figure 6.

Basic idea of applied navigation and control diagram is illustrated in Figure 2.

In Figure 1, three paths are shown that are automatically determined based on the derived navigation function shown in Figure 6, which is interpolated at runtime from a stored discrete potential field of the layout, as shown in Figure 3. Other navigation functions are determined for other desired frequent destinations such as pickup belts, workstations, or battery charging arias. Such navigation functions can be computed in advance if the pickup and drop-off locations are fixed and the robots can move in predefined corridors. This leads to a computationally efficient approach with high-quality trajectories that takes static obstacles into account during design and can also be extended to include detected dynamic obstacles. Further details of the navigation function and control algorithm are presented below.

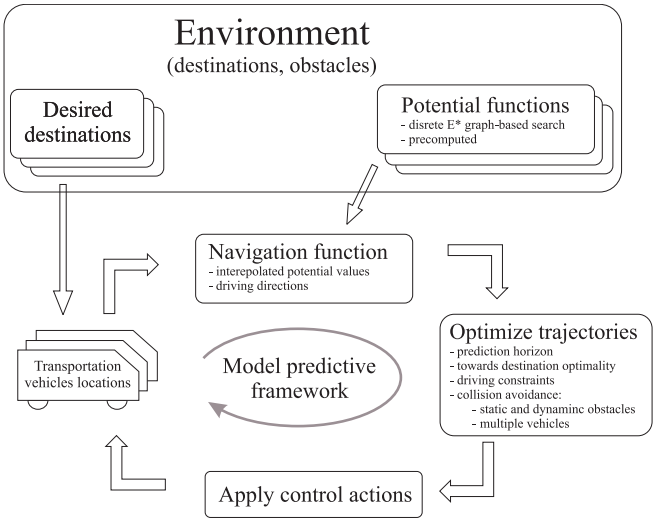

Figure 2. Basic idea of navigation and control of multiple vehicles based on the proposed navigation function and model predictive control.

2.1. Concept of Navigation

The navigation function $N(x, y, \varphi)$ is used to drive a wheeled robot with position x, y and orientation φ) safely between obstacles to the goal. A control algorithm therefore steers the robot to locations where $N(x, y, \varphi)$ decreases. The unimodal potential function is a good choice for $N(x, y, \varphi)$ because it has a single minimum ($N(x, y, \varphi) = 0$) at the goal and no local minima where the control algorithm might get stuck. Additionally, $N(x, y, \varphi)$ must have the highest values at the obstacles. A graph search algorithm such as D* for dynamic environments can be used to obtain such a potential function $U(x, y)$ as shown in Figure 3. The value of $U(x, y)$ represents the distance to the target cell, which is computed as the sum of the distances between cells (d_c) along the path. Such a search is computationally efficient since it is performed on a discrete grid of the environment but is not suitable for a control algorithm since $U(x, y)$ is constant for any robot position within a discrete cell. Therefore, the grid-based navigation function must be modified to obtain a unique value for each position within a cell that retains the property of a single minimum [31]. In the following, we propose a bicubic interpolation approach.

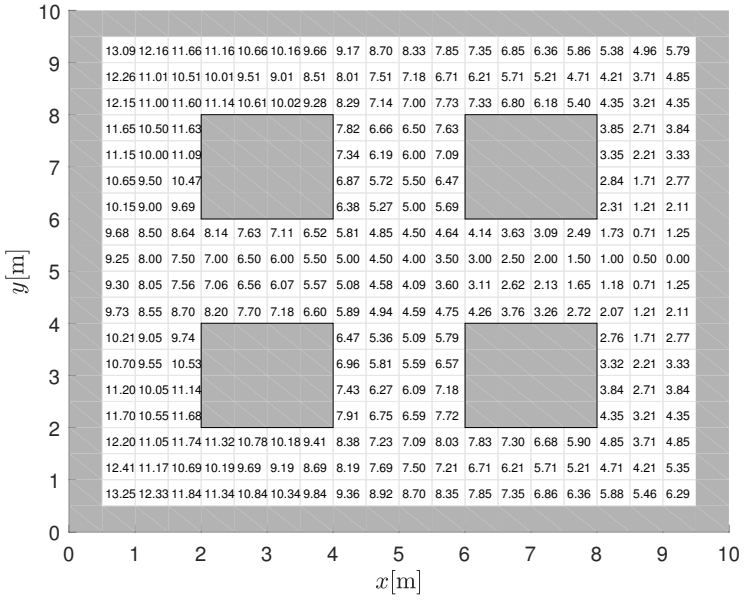

Figure 3. 3D view of the discrete potential function from obtained a grid-based search with 0.5 m resolution, a target position at $x = 9.25$ m, $y = 5.25$ m, and occupied cells with $U(x, y) = \infty$ (grey cells).

2.2. Bicubic Interpolation

To obtain a smooth interpolated potential $P(x, y)$ from a discrete potential $U(x, y)$, bicubic interpolation [40] is used. For a given arbitrary position $[x, y]^T$ within a cell **M**, an interpolated potential is calculated based on a 4×4 cell neighborhood, as shown in Figure 4. Depending on the quadrant of cell **M** in which the point $[x, y]^T$ is located, an appropriate four-cell neighborhood is determined whose centers form a square, as shown in Figure 4 shown by a dashed line.

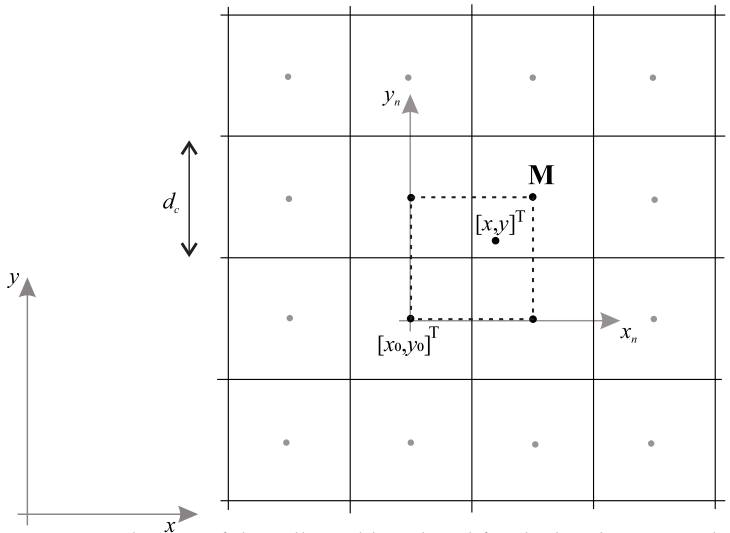

Figure 4. Selection of the cell neighbourhood for the bicubic interpolation of the potential field based on a point $[x, y]^T$ within the cell **M**.

The normalized coordinates $x_n, y_n \in [0, 1]$ are defined as $x_n = \frac{x - x_0}{d_c}$, $y_n = \frac{y - y_0}{d_c}$, where the origin $[x_0, y_0]$ is defined by the lower left corner of the dashed square and d_c is the cell size. The interpolated and discrete potential in normalized coordinates are expressed as $P_n(x_n, y_n) = P(x, y)$ and $U_n(x_n, y_n) = U(x, y)$, respectively. Define the potential and estimated partial derivatives for the four adjacent cell centers (corners of a dashed square in Figure 4)

$$p_{rc} = U_n(x_n, y_n)\Big|_{x_n=r,\, y_n=c}$$

$$f_{xrc} = \frac{\partial P_n}{\partial x_n}\Big|_{x_n=r,\, y_n=c} \approx \frac{U_n(r+1, c) - U_n(r-1, c)}{2}$$

$$f_{yrc} = \frac{\partial P_n}{\partial y_n}\Big|_{x_n=r,\, y_n=c} \approx \frac{U_n(r, c+1) - U_n(r, c-1)}{2}$$

$$f_{xyrc} = \frac{\partial^2 P_n}{\partial x_n \partial y_n}\Big|_{x_n=r,\, y_n=c} \approx \frac{U_n(r+1,c+1) - U_n(r-1,c+1) - U_n(r+1,c-1) - U_n(r-1,c-1)}{4},$$

where $r, c \in \{0, 1\}$ and $U_n(x_n, y_n) = U(x, y)$.

For a given arbitrary position, the interpolated potential is then defined by bicubic interpolation as follows

$$P_n(x_n, y_n) = \begin{bmatrix} 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \mathbf{A} \begin{bmatrix} 1 & y_n & y_n^2 & y_n^3 \end{bmatrix}^T, \tag{1}$$

where the matrix of coefficients is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & f_{y00} & f_{y00} \\ p_{10} & p_{11} & f_{y10} & f_{y11} \\ f_{x00} & f_{x01} & f_{xy00} & f_{xy01} \\ f_{x10} & f_{x11} & f_{xy10} & f_{xy11} \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

And negative gradient of $P(x,y)$ in $[x,y]^T$ is computed in a closed-form as

$$\begin{aligned} \overrightarrow{\mathbf{g}}(x,y) &= -\nabla P(x,y) = -\left[\frac{\partial P(x,y)}{\partial x}, \frac{\partial P(x,y)}{\partial y}\right]^T = \\ &= -\frac{1}{d_c}\left[\frac{\partial P(x_n,y_n)}{\partial x_n}, \frac{\partial P(x_n,y_n)}{\partial y_n}\right]^T \end{aligned} . \tag{2}$$

For non-holonomic robots (with the kinematics given in 6), the final navigation function depends on the interpolated potential $P(x,y)$ and on the robot orientation φ

$$\begin{aligned} N(x,y,\varphi) &= P(x,y) + \xi e(\varphi) \\ e(\varphi) &= \min_{k=\{0,1,-1\}} |\angle \overrightarrow{\mathbf{g}}(x,y) - \varphi + 2k\pi|, \end{aligned} \tag{3}$$

where $e(\varphi)$ is the absolute orientation error, $\xi > 0$, and $\angle \overrightarrow{\mathbf{g}}(x,y)$ is the orientation of the negative gradient .

In Figure 5, the interpolated potential function $P(x,y)$ of the free space and the centrally located target is shown for discrete potentials obtained by A* and E* grid-based searches. Since A* uses 4 and 8 neighbourhood connections, respectively, the gradients of $P(x,y)$ remain multiples of 90° and 45°, respectively (see contours of the same potential in Figure 5). The E* [25] is a dynamic path planning algorithm that can approximate continuous gradients (and contours). It uses 4 neighbour connectivity (such as A* or D*), but instead of one, two parent nodes in orthogonal directions are used to get a better cost-to-goal estimate for each cell. Using the discrete potential field obtained from the E* algorithm, the interpolated potential function (1) is smoother with arbitrary direction of the negative gradient (2).

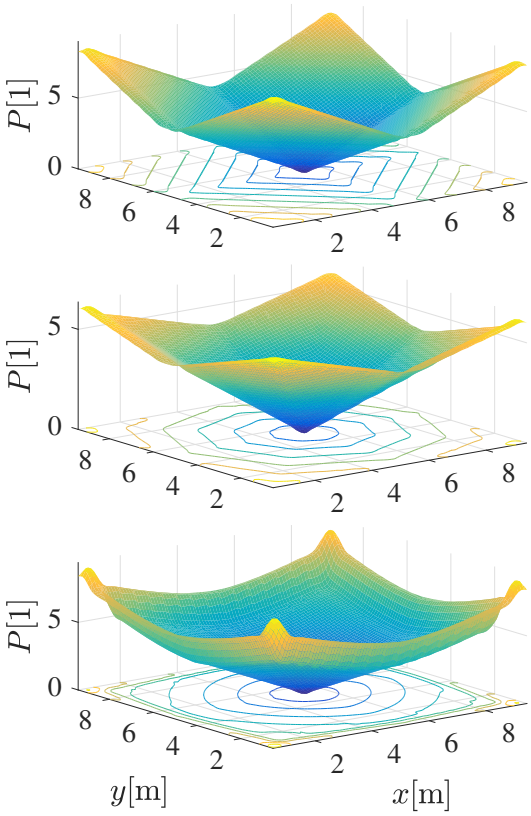

Figure 5. Interpolated potential function $P(x,y)$ based on the discrete potential obtained by A* with 4 and 8 neighbour cells connections and by E*. The obtained gradient is in the directions of multiples of 90° or 45° when 4 or 8 neighbourhood connections are used in A*. While E* can have arbitrary directions, which can be seen from the contours of equal potentials orthogonal to the gradient direction.

An example of an occupied space and its computed interpolated smooth navigation function is given in Figure 6. Additionally, three paths are drawn from different starting points following the negative gradient towards the target with the lowest potential. The obtained paths are orthogonal to the contours of the same potential (see the lower part of Figure 6).

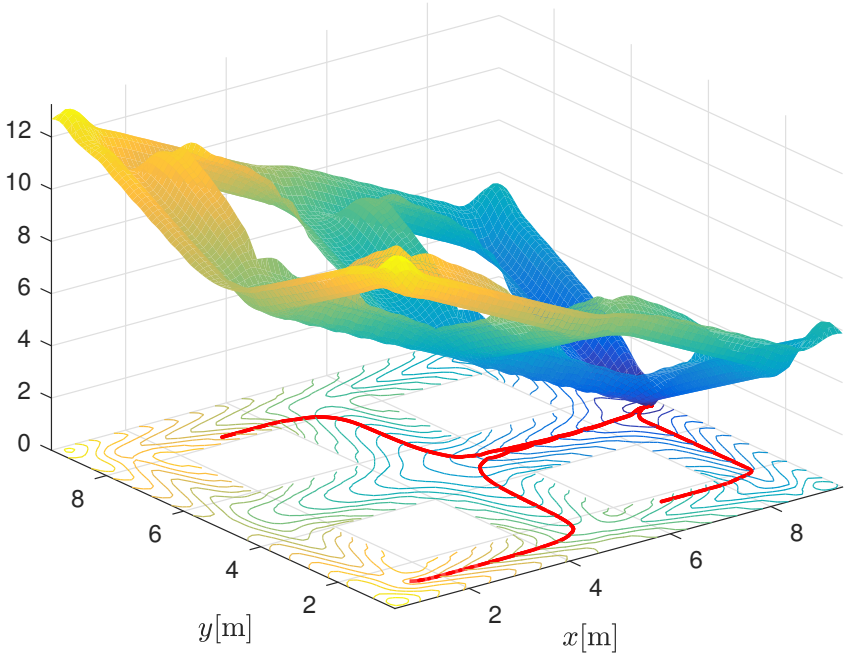

Figure 6. 3D view of the interpolated navigation function $N(x, y, \varphi)$. For clarity, $e(\varphi)$ is set to zero in Equation (3). Three paths are drawn from different starting points, following the negative gradient towards the goal location with the lowest potential value.

3. Coordinated Model Predictive Control

The proposed interpolated potential function Equation (1) allows the simple application of control of a single robot to safely navigate from anywhere to the target while automatically avoiding obstacles. The vehicle only needs to follow the given negative gradient direction Equation (2), and since the gradient has soft transitions (see e.g., Figure 3), feasible trajectories result. Such an approach lacks predictive capabilities and assumes a static environment without any other vehicles.

Therefore, the control behavior is defined as follows. The simple gradient-following reactive behavior is improved by incorporating prediction, so that the current control action also depends on the future states of the vehicles. The navigation function already includes knowledge of a static map in which the space occupied by obstacles has infinite potential. However, observed dynamic obstacles (cooperative obstacles such as other transport vehicles or non-cooperative obstacles such as humans or forklifts controlled by humans, etc.) are not included in the navigation function as this would require constant replanning. Dynamic obstacles are observed by sensors (laser range finder, camera, etc.), and their movement is estimated in the prediction horizon of the controller. A feasible trajectory is determined in the prediction horizon that is consistent with the navigation function, does not conflict with other vehicles or other detected obstacles, and is within the kinematic and dynamic constraints of the vehicle.

3.1. Control Definition

Model predictive control (MPC) is defined as an optimization problem with constraints. Optimal controls $\mathbf{u}(t) = [v(t), \omega(t)]^T$ are found for a differential robot over a prediction horizon h that minimize the objective function J at the current robot state $\mathbf{s}(t) = [x, y, \varphi]^T$

$$\begin{aligned} J(\mathbf{s}(t)) \quad &= \min_{\mathbf{u}(i-1)} \sum_{i=1}^{h} \left(N(\mathbf{s}(i)) + \mathbf{u}^T(i-1)\mathbf{R}\mathbf{u}(i-1) \right) \\ &\text{subject to:} \\ &\text{- free configuration space (Equation (5))} \\ &\text{- driving constraints (Equation (7))} \\ &\text{- convergent behavior (Equation (11))} \\ &\text{- collision-free with dynamic obstacles (Equation (12))} \end{aligned} \tag{4}$$

where v and ω are translational and angular velocity, i is a shorthand notation for time $t + iT_s$, T_s is the sampling time, and $\mathbf{R}$ is the weighting matrix. Note that MPC considers the robot's motion model and the environment model, where the target and static obstacles are already considered in the navigation function. However, due to kinematic constraints, collisions with static obstacles may still occur. Therefore, a valid trajectory in the horizon must lie in the free configuration space of the static map Q_{free}

$$\mathbf{s}(i) \in Q_{free} \quad , \quad i = 1, \ldots, h. \tag{5}$$

Similarly, collisions with dynamic obstacles are considered. For more details, see the Section 3.1.4. The future state of the robot $\mathbf{s}(i) = [x(i), y(i), \varphi(i)]^T$ is predicted using differential drive kinematics

$$\begin{aligned} x(i+1) &= x(i) + v(i)T_s \cos\left(\varphi(i) + \tfrac{\omega(i)T_s}{2}\right) \\ y(i+1) &= y(i) + v(i)T_s \sin\left(\varphi(i) + \tfrac{\omega(i)T_s}{2}\right) \\ \varphi(i+1) &= \varphi(i) + \omega(i)T_s. \end{aligned} \tag{6}$$

3.1.1. Driving Constraints

Control actions are constrained by maximum velocities and accelerations by

$$\begin{aligned} 0 \le v(i) \le v_{\max} \quad &, \quad |\omega(i)| \le \omega_{\max} \\ \tfrac{|v(i)-v(i-1)|}{T_s} \le a_{\max} \quad &, \quad \tfrac{|\omega(i)-\omega(i-1)|}{T_s} \le \alpha_{\max}, \end{aligned} \tag{7}$$

where $v_{\max}$, $\omega_{\max}$, $a_{\max}$, and $\alpha_{\max}$ are maximum allowable translational and rotational velocities and accelerations.

3.1.2. Length of the Horizon

During the horizon, let the robot travel on an arc, where $v(i)/\omega(i)$ is its radius. A constant arc in the horizon is convenient because it reduces the computational cost of the MPC problem since it only requires the optimization of two parameters. The choice of horizon length affects the driving performance, safety, and computational cost of MPC.

The minimum horizon length ($h = h_{min}$) is chosen so that the robot travelling at maximum speed can safely decelerate to a stop at the end of the prediction horizon

$$h_{min} = \left\lceil \max\left(\frac{v_{\max}}{a_{\max} T_s}, \frac{w_{\max}}{\alpha_{\max} T_s} \right) \right\rceil + 1. \tag{8}$$

This prevents the worst case collision with the maximum robot speed and the newly observed (static) object at the end of the prediction horizon. The control $\mathbf{u}(i), i \in 0, 1, \ldots, h-1$ in the horizon therefore decreases linearly to zero at the end of the horizon, as follows

$$\mathbf{u}(i) = \begin{cases} \mathbf{u}(0) & , \quad 0 \leq i \leq h-1-N_{dec} \\ \mathbf{u}(0)\frac{h-1-i}{N_{dec}} & , \quad h-1-N_{dec} < i \leq h-1. \end{cases} \tag{9}$$

where the required number of deceleration samples at the end of the horizon is $N_{dec} = \left\lceil \max\left(\frac{v(0)}{a_{\max} T_s}, \frac{w(0)}{\alpha_{\max} T_s} \right) \right\rceil$ at the current robot velocity $\mathbf{u}(0) = [v(0), \omega(0)]^T$.

Choosing a larger horizon ($h > h_{min}$) also increases safety for moving objects because the motion of the obstacle is predicted early enough to find alternative trajectories that efficiently avoid the collision. However, too large a horizon increases the computational cost and may lead to worse trajectories in free space due to the averaging effect since a longer constrained trajectory does not fit as optimally on the surface of the navigation function.

As a compromise, we choose a larger horizon ($h > h_{min}$) and allow the robot to stop even before the end of the horizon (e.g. at $h_{stop} \leq h$) and keep the remaining number of samples $h - h_{stop}$ still. This effectively makes the horizon variable (with variable stopping time), and since all optimised trajectories have the same number of samples (h), their objective functions in Equation (11) are still comparable. The velocity profile in the horizon is then

$$\mathbf{u}(i) = \begin{cases} \mathbf{u}(0) & , \quad 0 \leq i \leq h_{stop}-1-N_{dec} \\ \mathbf{u}(0)\frac{h_{stop}-1-i}{N_{dec}} & , \quad h_{stop}-1-N_{dec} < i < h_{stop} \\ [0,0]^T & , \quad h_{stop} \leq i \leq h-1 \end{cases} \tag{10}$$

where the candidates for h_{stop} are selected according to the previous optimal curve by evaluating four possibilities $h_{stop} \rightarrow h_{stop} + \{0, -1, -2, +1\}$ that need to be in range $(N_{dec} + 1) \leq h_{stop} \leq h$. Initial value is set to $h_{stop} = h_{min}$.

Optimal control sequence $\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(h-1)$, which minimizes Equation (4), defines the best feasible future trajectory, and its first control action is applied to the robot in the current time. In the next time sample, the procedure repeats.

3.1.3. Convergent Behavior

To ensure convergent behavior of the MPC control, the summands $V(\mathbf{s}(i)) = N(\mathbf{s}(i)) + \mathbf{u}^T(i-1)\mathbf{R}\mathbf{u}(i-1)$ in the criteria Equation (4) will have to decrease in the horizon. This follows from the convergence constraint in Equation (4)

$$N(\mathbf{s}(i)) \geq N(\mathbf{s}(h)), \; i = 1, \ldots, h. \tag{11}$$

In the worst case, if all the candidate control actions in Equation (4) result in trajectories that violate the convergence constraint Equation (11), the robot can still choose the optimal trajectory from the previous control step, shifted by one sample. Since the trajectory slows down at the end of the horizon (see Equations (9) and (10)), the robot will start slowing down earlier. This can happen if some dynamic (non-cooperative) objects block its path.

3.1.4. Preventing Conflicts with Dynamic Obstacles

The environment may contain dynamic obstacles that can be treated as cooperative objects (e.g., other robots) and non-cooperative objects (e.g., forklifts operated by humans). Cooperative objects are assumed to have intentions and trajectories known to the robot for at least a prediction horizon h. The intentions of non-cooperative objects can be estimated from sensor observations (e.g., laser range scans) of their past movements by estimating their velocities and predicting the most likely trajectories in the horizon.

For a given control $\mathbf{u}(i)$ in Equation (4), the robot trajectory $\mathbf{s}(i)$ is collision-safe (CS) if it does not collide with any moving object trajectory (static obstacles are already considered in Equation (5)). Let $o \in \mathcal{O}$ denote all other moving objects from a set $\mathcal{O}$,

and let $\mathbf{s}_o(i) = [x_o(i), y_o(i), \varphi_o(i)]^T$ be the location of an object at horizon prediction time $i \in 1, \ldots, h$

$$CS \implies \nexists o \in \mathcal{O} : (||\mathbf{s}(i) - \mathbf{s}_o(i)|| < d_{safe}) \; and \; |\varphi(i) - \arctan(\tfrac{y_o(i)-y(i)}{x_o(i)-x(i)})| < \varphi_{safe}, \quad (12)$$

d_{safe} is the required safety distance between the robot and an object o, and φ_{safe} is the range of angular deviation from the robot's forward motion that can lead to a collision, $\arctan(\cdot)$ is the four-quadrant inverse tangent version. Any robot trajectory that is not collision safe to moving objects is rejected in Equation (4).

The same procedure is followed for all cooperative robots. After one robot determines optimal controls (and trajectory in the horizon), the other robots can adapt by finding collision-proof trajectories to the previous robot. To prevent chattering behavior (where two robots can switch between different optimal controls while avoiding collisions), the selection of optimal controls (trajectories) is done sequentially, which is natural if all robots are controlled from a central computer. If the first robot determines an optimal trajectory that avoids all other robots (taking into account the predicted trajectories of the others), the next robot will adapt by finding collision-safe paths (e.g., swerving to the other side or slowing down). This will automatically lead to consensus since the current predicted trajectories are inherited and known to the others in the same sample. When robots plan and control autonomously, if there is a possibility of collision, they need to only negotiate the order in which they compute their trajectories. Alternatively, they can consider priorities when they are assigned (e.g., for priorities for transportation tasks), as in [10] and [18]. In this way, the first robot computes the optimal trajectory in the prediction horizon, which takes into account the previous trajectories of the others. Additional traffic rules (e.g., swerving to the right for head-on collisions) can also be used [9].

The proposed navigation approach is computationally efficient since the navigation function is precomputed for a static environment, and collisions with dynamic obstacles in the control (Equation (4)) are avoided at runtime.

Note that local minima can still occur (but this is unlikely in practise) if another robot (the second robot or the moving object) approaches another robot (the first robot) exactly from the direction opposite to the negative gradient of the first robot's navigation function. This also means that the second robot uses a different navigation function, to a different destination whose gradient is in the opposite direction to that of the first robot. When this happens, both robots may slow down as this can be cheaper (according to the MPC control cost (Equation (4)) than driving with the increased values of the navigation functions while avoiding a collision.

In this particular case, it may be a good choice to perform a dynamic replanning of the navigation functions with the detected possible collision position of the other robot. In this way, no slowdown or local minima can occur since the navigation function also considers moving obstacles. This replanning requires additional computation time and should therefore be performed incrementally using the dynamic E* algorithm [25]. Moreover, a relatively fine grid should be chosen (e.g., at most half the robot size) to reduce the discretization error when the predicted collision location is snapped to a grid. Therefore, replanning can only occur if an object is in the collision with the robot in the predicted horizon. Note that the presented MPC approach remains the same if replaned navigation function is used in (Equation (4)), where the objects contained in the navigation function will no longer appear as dynamic collision constraints (Equation (12).

4. Optimisation Strategy in MPC Control

In the following, we propose a novel optimization strategy for solving (Equation (4)) that combines optimization with a fixed set of control action candidates and particle swarm optimization.

4.1. Fixed Candidate Optimization

To reduce the computational cost, Fixed Candidate Optimization (FCO) is introduced in [36] to solve the MPC problem. Given the current velocities $\mathbf{u}_c = [v_c, \omega_c]^T$ (applied to the robot at time $t - T_s$), a set of possible discrete accelerations $a_c \in \{-a_{max}, 0, a_{max}\}$, $\alpha_c \in \{-\alpha_{max}, 0, \alpha_{max}\}$ is defined to produce a set of 9 candidate velocities for optimization

$$\mathbf{u}(t) = \mathbf{u}_c + T_s[a_c, \alpha_c]^T \tag{13}$$

constrained by Equation (7).

The main strengths of the proposed MPC with FCO are the low computational complexity and the generation of near-optimal trajectories with a guaranteed convergence, as shown in [36]. However, the obtained velocity profile contains higher noise due to the coarse set of possible accelerations.

4.2. Particle Swarm Optimization

Particle swarm optimization (PSO) uses a stochastic strategy with a swarm of randomly perturbed particles to find a solution. Applying PSO to the MPC problem yields arbitrary velocities $\mathbf{u}(t)$ sampled from a continuum and constrained by Equation (7). Each particle k is parameterized by a parameter vector $\mathbf{p}_k = [v_k, \omega_k]^T$ defining its velocities and an increment vector $\Delta\mathbf{p}_k$ defining the change in velocities. During MPC optimization, the population of all particles is iteratively updated and validated according to the objective function (4). Each particle keeps track of its parameters and remembers its best previously achieved parameter $\mathbf{pB}_k$, along with its associated objective function $J_k = f(\mathbf{pB}_k)$, where f is the function that is minimized in Equation (4). During optimization, the global best parameter vector of the entire swarm $\mathbf{gB}$ is also remembered.

In each sample time, the particles are iteratively updated according to the following rules

$$\begin{aligned} \Delta\mathbf{p}_k &\leftarrow \gamma\Delta\mathbf{p}_k + c_1\mathbf{rand}(0,1)\cdot(\mathbf{pB}_k - \mathbf{p}_k) + c_2\mathbf{rand}(0,1)\cdot(\mathbf{gB} - \mathbf{p}_k) \\ \mathbf{p}_k &\leftarrow \mathbf{p}_k + \Delta\mathbf{p}_k, \end{aligned} \tag{14}$$

where $\gamma > 0$ is the inertia factor, $c_1 > 0$ is the self-cognitive constant, $c_2 > 0$ is the social constant, and $\mathbf{rand}(0,1)$ is a vector of uniformly distributed values in the range $[0,1]$. At the end of the optimization, the best parameter is applied to the robot $\mathbf{u}(t) = \mathbf{gB}(t)$.

MPC with PSO produce smoother velocity profiles and can find better solutions since no velocity discretization is used. However, the computational complexity becomes much higher (compared to MPC with FCO) due to multiple required iterations with more particles. Due to the random nature of PSO, both the solution and the convergence of the search are not guaranteed.

4.3. Combined Deterministic-Stochastic Optimization

The main idea is to combine FCO and PSO in the so-called combined deterministic-stochastic optimization (CDS) and to exploit the advantages of both algorithms to generate trajectories with a smooth speed profile, with guaranteed convergence and low computational complexity.

CDS is a modified PSO algorithm (shown in Algorithm 1) that executes $K_F = 9$ fixed particles and K_C changing particles in parallel. Fixed particles are initialized by Equation (13) and are not updated during optimization. These fixed particles provide good starting parameters that can be used by other changing particles through $\mathbf{gB}$ when iteratively updated through Equation (14). In this way, CDS provides a better (more optimal and smoother) or at least as good a solution as FCO itself. MPC with CDS is guaranteed to converge to the goal in a finite time from any unoccupied location in the environment where the goal is reachable ($N(x, y, \varphi) < \infty$). The algorithm is computationally efficient since the number of changing particles K_C in CDS can be much smaller than in the corresponding PSO.

Algorithm 1 Combined deterministic-stochastic optimization.

Require: List of particles $k = 1, \cdots, K$ where first $k = 1, \ldots, K_F$ are fixed particles.
  for each particle $k = 1, \cdots, K_F$ **do**
    Initialize $\mathbf{p}_k$ by Equation (13).
  end for
  for each particle $k = K_F + 1, \cdots, K$ **do**
    Randomly initialize $\mathbf{p}_k$, $\Delta\mathbf{p}_k = [0, 0]^T$, $\mathbf{pB}_k = \mathbf{p}_k$.
  end for
  $J_{\text{best}} = \infty$, $iter = 1$
  repeat
    for each particle $k = 1, \cdots, k$ **do**
      if $k > K_F$ | $iter == 1$ **then**
        Compute objective J_k by (4) considering ramp down (Equation (9) and add penalty for Equation (7), Equation (12) violation.
        Set *convergent* condition by Equation (11).
        if $J_k < f(\mathbf{pB}_k)$ **then**
          $\mathbf{pB}_k = \mathbf{p}_k$
        end if
        if $f(\mathbf{pB}_k) < J_{\text{best}}$ & *convergent* **then**
          $\mathbf{gB} = \mathbf{pB}_k$, $J_{\text{best}} = f(\mathbf{gB})$
        end if
      end if
    end for
    for each particle $k = K_F + 1, \cdots, K$ **do**
      Update $\mathbf{p}_k$ and $\Delta\mathbf{p}_k$ by Equation (14) constrained by Equation (7).
    end for
  until $iter \leq MAXiter$

5. Simulation Results

5.1. Single Robot Navigation

The performance of a single robot in the environment from Figure 1 is first illustrated when the robot needs to transport products or materials from different starting locations to different destinations. A possible scenario could be that the robot has to deliver a semi-finished product to workstation 1 and then transport it to the dropoff location (see top left image in Figure 7).

The navigation functions are computed in advance for known locations, which minimizes the online computational overhead (no online planning is required) and makes the system robust to disturbances during control (e.g., deviations from the original desired path due to errors in robot location, control performance, or dynamic obstacles). Since only a discrete cost map needs to be stored to interpolate the navigation values from it, this is also not memory-intensive (20×20 cost-to-goal for the environment in Figure 1) with the cell size $d_c = 0.5$ m.

In Figure 7, the desired target is at $x = 3.1$ m, $y = 8.3$ m (e.g., workstation 1 in Figure 1) for red paths. The destination can be safely reached from any location considering the navigation function (the top right image in Figure 7) in the proposed MPC control. For a different desired destination (e.g., drop-off location at $x = 9.3$ m, $y = 5.1$ m in Figure 1), a different navigation function is used for all green paths (the bottom right image in Figure 7). The simulation results are obtained using the following parameters. The interpolation of the navigation function is performed on the grid-based search with a cell size resolution of $d_c = 0.5$ m. As the interpolation is applied, good navigation and control performance can be obtained even at coarse resolutions. Optimization in MPC is performed using a fixed horizon length $h = 14$ (with deceleration at the end, as shown in Equation (9)) and sample time $T_s = 0.1$ s, and by considering the constraints on velocities and accelerations $v_{max} = 1$ ms^{-1}, $\omega_{max} = 6$ s^{-1}, $a_{max} = 1$ ms^{-2}, and $\alpha_{max} = 6$ s^{-2}.

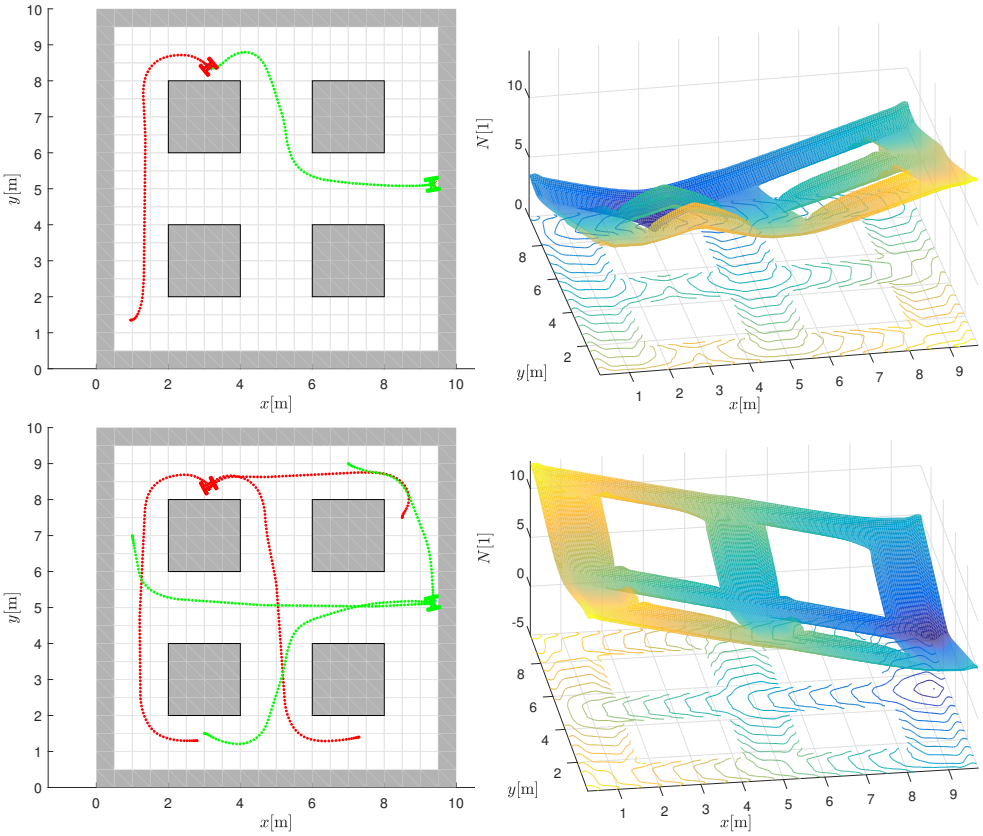

Figure 7. Examples of navigation and control of a single robot in two different destinations defined by the minimum values of the navigation functions in the right column. All red paths are obtained by the navigation function in the top right image, while the green paths are obtained by the navigation function in the bottom right image.

The performance of model predictive control with the proposed combined deterministic-stochastic optimization (MPC-CDS) is compared with the results of fixed candidate optimization (MPC-FCO) and particle swarm optimization (MPC-PSO). In addition, MPC-based algorithms are compared with the kinodynamic stable sparse RRT planning approach (SST) approach [19]. The results are shown in Figures 8 and 9 for the U-obstacle map, Maze map, and the Random-obstacle map. All MPC trajectories are computed for the bicubic interpolation function and also collected in Table 1. The results are compared in terms of obtained trajectories, velocity profiles, length of trajectory L, travel time t_{goal}, cumulative navigation A_N, and normalized computational efficiency E_{comp} (according to MPC-FCO). The computational complexity of MPC-FCO depends on its implementation. In our case, it allows for real-time operation with a refresh rate of at least 50 Hz on a 2.80 GHz Intel dual-core processor with C++ implementation.

The best performance of MPC is obtained by the PSO optimization approach (Figures 8 and 9 and Table 1), where the obtained trajectories are short and fast and the velocities have a smooth profile. However, the computational complexity of MPC-PSO is higher (than MPC-FCO or MPC-CDS) because it uses 25 particles and 20 iterations to optimize each control sample. Similar performance in terms of trajectory length and travel time is obtained with MPC-CDS, which uses nine fixed particles and only two changing particles. The results of CDS are a compromise between the quality of trajectories generated by PSO and the computational complexity of FCO. CDS produces smoother velocity profiles than FCO and requires much less computational effort than PSO. SST produces similarly long trajectories, sometimes shorter since it does not take into account safety costs around the obstacles, but with much slower velocity profiles due to the randomness of velocity selection during the search process. Unlike the MPC-based algorithms, the SST algorithm computes the entire trajectory to the goal before the robot begins execution.

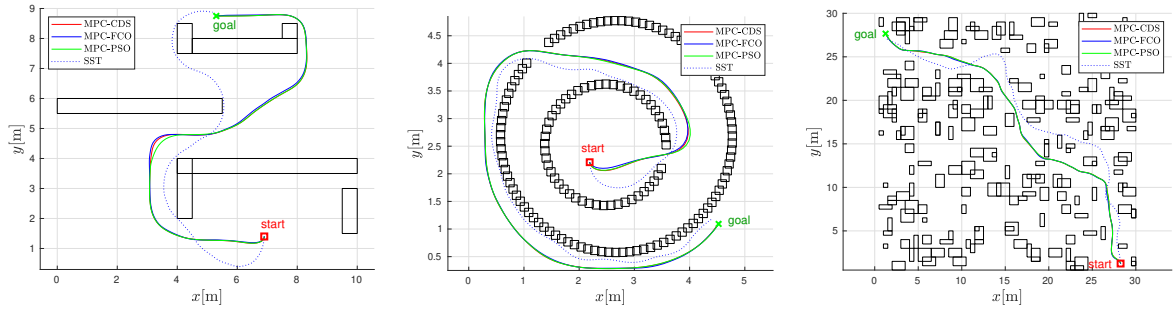

Figure 8. Trajectory comparison for the U-obstacle map (left), the Maze map (middle), the Random-obstacle map (right), the bicubic interpolated navigation, and the SST algorithm.

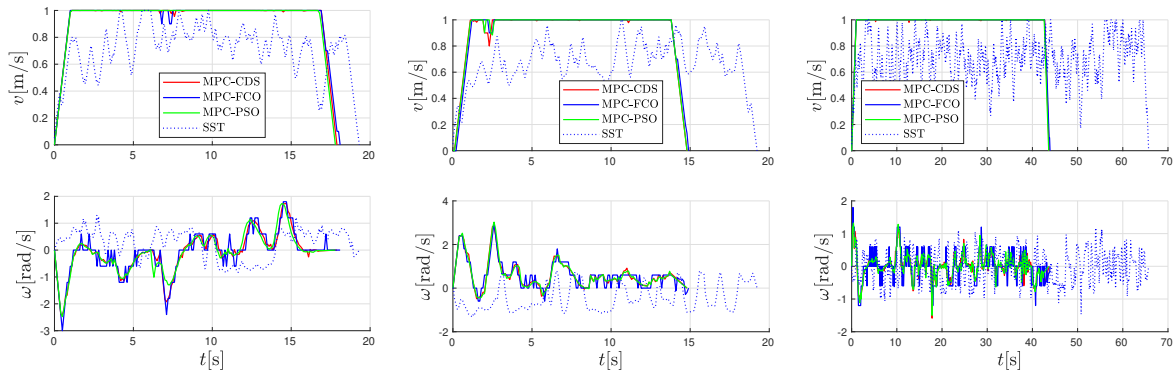

Figure 9. Comparison of velocity profiles for the U-obstacle map (left), the Maze map (middle), and the Random-obstacle map (right).

Table 1. Algorithms validation on three different maps.

	Alg.	L [m]	t_{goal} [s]	A_N [1]	E_{comp} [1]
U map	MPC-CDS	16.88	17.90	230.39	1.61
	MPC-FCO	16.95	18.10	230.84	1.00
	MPC-PSO	16.79	17.80	228.68	13.11
	SST	13.82	19.18	n/a	n/a
Maze map	MPC-CDS	13.63	14.80	111.30	1.73
	MPC-FCO	13.64	14.90	111.66	1.00
	MPC-PSO	13.65	14.80	111.73	13.63
	SST	12.87	19.09	n/a	n/a
Rnd map	MPC-CDS	42.69	43.70	1090.24	2.90
	MPC-FCO	42.78	43.90	1092.28	1.00
	MPC-PSO	42.59	43.60	1091.18	28.01
	SST	46.07	65.59	n/a	n/a

5.2. Multiple Robot Coordinated Navigation

Analysis of the selection of the horizon length in MPC performance is first explained. The minimum horizon length h_{min} (Equation (8)) is sufficient for navigation in static environment (obstacles mapped or unknown in the navigation function), but it may not be good enough for moving obstacles. For moving obstacles and $h > h_{min}$, safety and navigation performance increases because collision threats can be predicted early enough so that better avoidance routes can be found. The analysis of the varying horizon length (where the moment of deceleration can also occur before the end of the horizon, as defined in Equation (10)) for the navigation of two robots approaching a head-on collision and a cross collision ([9]) is shown in Figure 10 and Table 2. In a head-on collision (left image in Figure 10), the robots stop to avoid collision when the minimum horizon $h = h_{min} = 11$ is

chosen, while the robots can safely navigate to the target for $h > h_{min}$. This scenario is more difficult than the cross-collision (right part of Figure 10) since the other robots move in the opposite direction of the negative gradient of the navigation function. A larger horizon can provide better collision avoidance but increases the computational cost of navigation (E_{comp} increases in Table 2). E_{comp} is the normalized computational load corresponding to the computational time at $h = h_{min}$ (where in the first line the value $E_{comp} = 1$ is normalized by the computational time of the ignored collision, since the robots do not reach the goals). A larger horizon can slightly reduce both the distance traveled (joint distance $\sum L$ in Table 2) and the travel time (joint travel time $\sum T$ in Table 2), which means that the robots do not need to wait or slow down to avoid a collision. Note that the improvement in travel time and distance is relatively small compared to the increased computational cost. Therefore, the main reason for increasing the horizon is safety and collision avoidance performance.

Table 2. Performance at variable horizon.

	h/h_{min}	$\sum L$ [m]	$\sum T$ [s]	E_{comp} [1]
head-on	11/11	/	/	1.00
	16/11	14.27	16.40	1.46
	22/11	14.26	16.40	2.05
cross	11/11	13.48	16.20	1.00
	16/11	13.38	15.90	1.63
	22/11	13.33	15.80	2.27

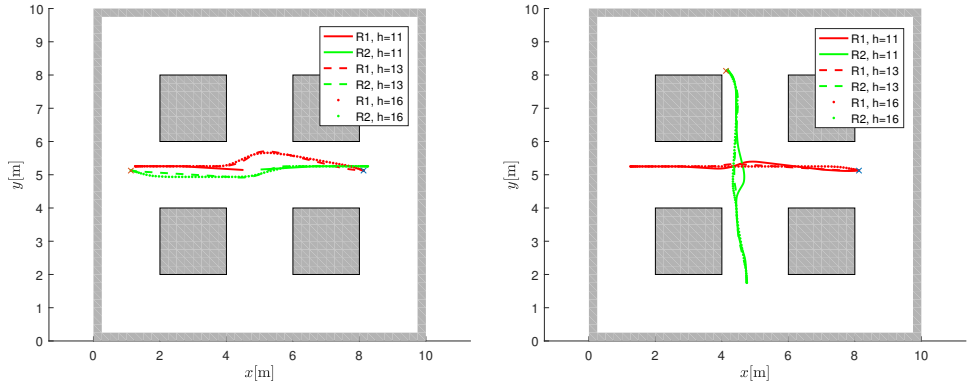

Figure 10. Analysis of horizon length in frontal collision avoidance (left) and cross collision avoidance (right). The target locations are indicated by a cross. A larger prediction horizon can find better trajectories than the minimum horizon $h_{min} = 11$.

Some other examples of coordinated navigation and control of multiple robots can be found in Figure 11, where the navigation results are shown without collision avoidance (left images, where the robots drive over each other) and with coordinated collision avoidance navigation (right images). The starting position of the i-th robot is marked with Ri, and its target position coincides with the final robot position. The occurrence of collisions is marked (left image in Figure 11) by ellipses Ci, where $C1$ is the first collision between robots 2 and 4; $C2$ between robots 2, 3 and 4; $C3$ between robots 1 and 3; and $C4$ between robots 4 and 5. The controller with coordinated predictive collision avoidance (right image in Figure 11) successfully avoids all collisions.

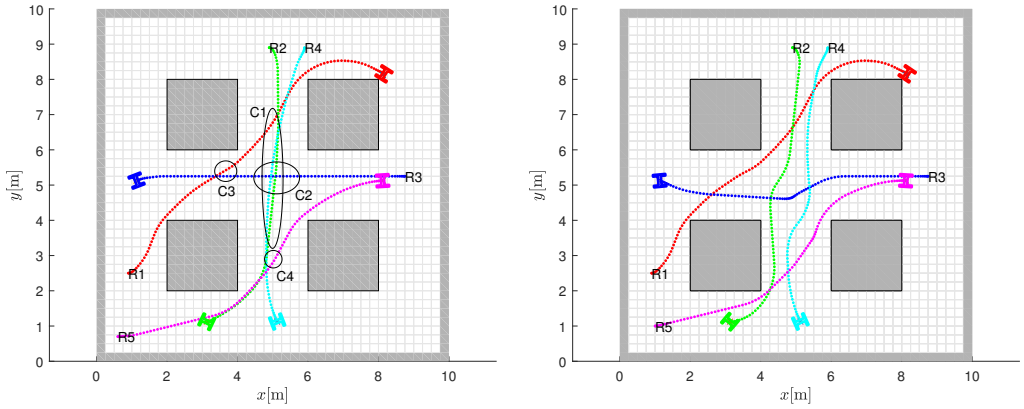

Figure 11. Coordinated collision avoidance. Obtained robot path without using collision avoidance in MPC (where robots drive over each other), with collision cases marked by ellipses (left image). Collision avoidance with prediction horizon $h = 20$ finds safe routes with similar travel times (right image).

Coordinated collision avoidance for symmetric initial locations and congested traffic in the centre of the map is shown in Figure 12. The obtained control with avoidance and prediction horizon $h = 15$ fails to navigate the robots to the destinations as the robots stop safely to avoid collisions (Figure 12, left image). Increasing the horizon to $h = 25$ results in safe trajectories to the targets (Figure 12, right image).

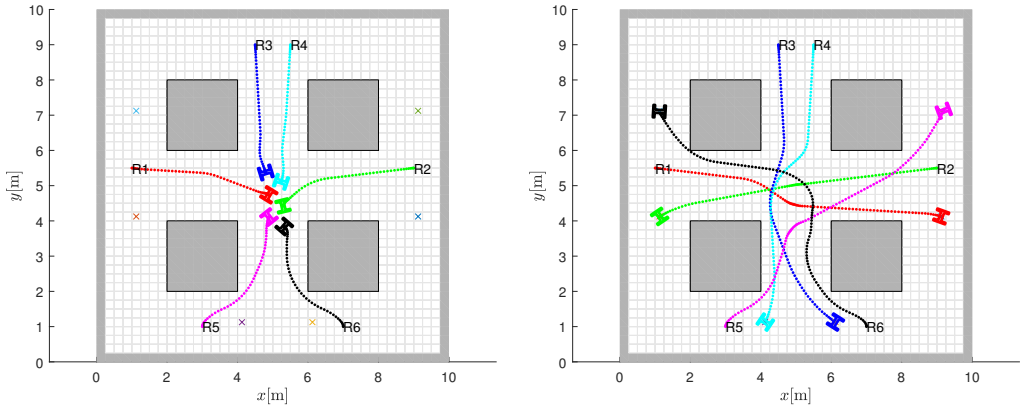

Figure 12. The obtained control with avoidance and prediction horizon $h = 15$ cannot steer the robots towards the destinations since the robots stop safely to prevent collision (left image). Increasing the horizon to $h = 25$ leads to safe trajectories towards the target locations (right image).

5.3. Experiments

Navigation is performed also in the real map shown in Figure 15 and 16 (floor plan of our laboratory). The map is an occupancy grid with 10 cm resolution created with the Sick LMS200 laser range finder. Four target locations $\mathbf{G}_{N1} = [2.4, 7]^T$, $\mathbf{G}_{N2} = [6, 6]^T$, $\mathbf{G}_{N3} = [10.5, 6]^T$, $\mathbf{G}_{N4} = [15, 4]^T$ (e.g., locations of workstations) are defined on the map. A robot can reach a desired goal through MPC control (Equation (4)) by following the navigation function (Equation (3)), which consists of an appropriate interpolated potential field. Figure 13 shows potential fields for the defined targets.

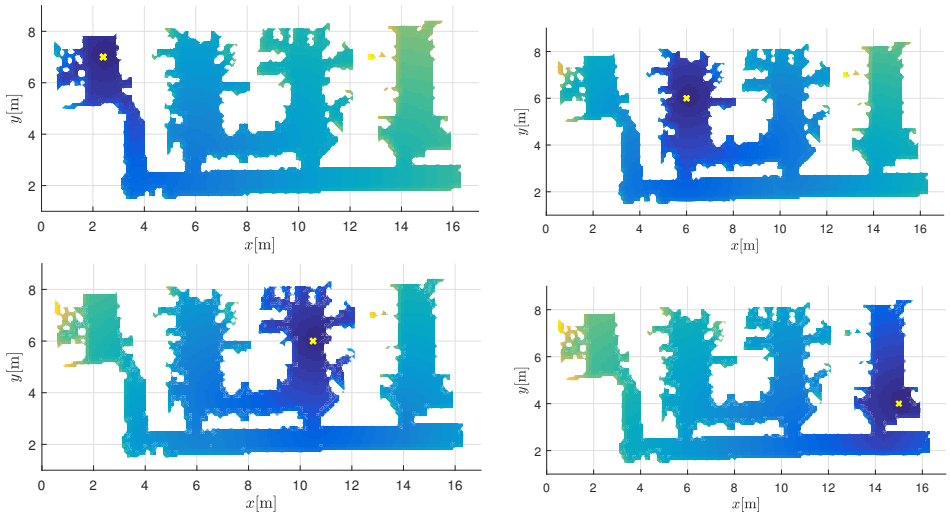

Figure 13. Four interpolated potential fields are used in the navigation functions $N_i(y, x, \varphi)$ ($i \in 1, \ldots, 4$) to find one of the desired target positions $\mathbf{G}_{N1} = [2.4, 7]^T$, $\mathbf{G}_{N2} = [6, 6]^T$, $\mathbf{G}_{N3} = [10.5, 6]^T$, and $\mathbf{G}_{N4} = [15, 4]^T$ from any initial position. The targets are located at the lowest value of the potential field (darkest region) and are marked by a cross.

In experiments, Roomba cleaning robots (Figure 14) are used to simulate transportation tasks between desired workstations. For localization, a camera is used to detect Aruco markers on the ceiling placed at known locations. The robots are controlled by a built-in Raspberry Pi, which sends velocity commands with an update frequency of 10 Hz ($Ts = 0.1$ s).

Figure 14. Roomba robots used to simulate transportation tasks in the laboratory layout from Figures 15 and 16. View of the robots (left) and closer robot view with integrated Raspberry Pi and camera (right).

During navigation, velocities and accelerations are constrained by $v_{max} = 0.45$ ms^{-1}, $\omega_{max} = 3$ s^{-1}, $a_{max} = 0.5$ ms^{-2}, and $\alpha_{max} = 3$ s^{-2}. To predict collision hazards with other robots, the horizon $h = 20 > h_{min}$ ($h_{min} = 11$ according to Equation (8)) is chosen and safety distance and angle are set to $d_{safe} = 0.35$ m and $\phi_{safe} = \pi/2$.

In Figure 15, the first robot uses the first navigation function (the upper left image in Figure 13) to get to the destination $\mathbf{G}_1 = \mathbf{G}_{N1}$. Similarly, the destination for the second robot is reached by the third navigation function ($\mathbf{G}_2 = \mathbf{G}_{N3}$) and the destination for the third robot is reached by the second navigation function ($\mathbf{G}_3 = \mathbf{G}_{N2}$).

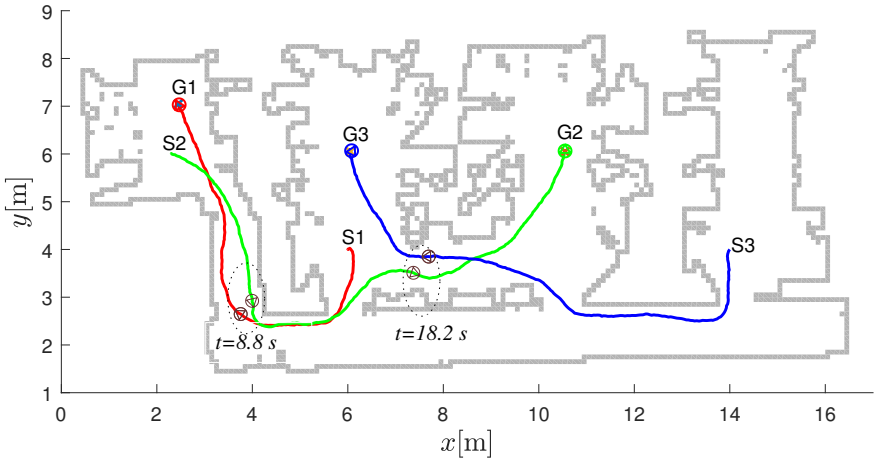

Figure 15. Robot paths in the laboratory layout. The start location of the i-th robot is denoted by $\mathbf{S}_i$ and the destination locations by $\mathbf{G}_i$. Safe navigation during collision avoidance is represented by dark gray circles belonging to passing robots at the same time (the left pair of circles belongs to time $t = 8.8$ s and the right pair to time $t = 18.2$ s).

In Figure 16, the first robot uses the second navigation function (the upper left image in Figure 13) to reach the destination $\mathbf{G}_1 = \mathbf{G}_{N2}$. Similarly, the destination for the second robot is reached by the fourth navigation function ($\mathbf{G}_2 = \mathbf{G}_{N4}$) and the destination for the third robot is reached by the third navigation function ($\mathbf{G}_3 = \mathbf{G}_{N3}$).

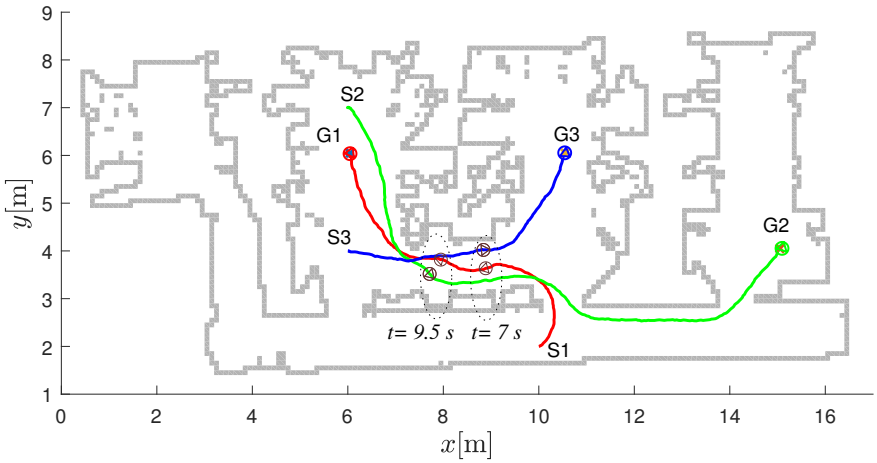

Figure 16. Robot paths in the laboratory layout. The start location of the i-th robot is denoted by $\mathbf{S}_i$ and the destination locations by $\mathbf{G}_i$. Safe navigation during collision avoidance is represented by dark gray circles belonging to the robots passing simultaneously (the right pair of circles belongs to time $t = 7$ s and the left pair to time $t = 9.5$ s).

6. Discussion

From simulations and experiments, the proposed interpolated navigation function combined with MPC computes collision-safe paths for a single vehicle that are near-optimal considering static obstacles. Moreover, the completeness of the system is guaranteed since the potential field does not contain local minima. The approach assumes that the global information about the system layout is known and static. This allows a discrete potential function (e.g., a distance-to-goal cost map) to be computed in advance and bicubic interpolation to be performed only at runtime to obtain a computationally efficient continuous estimate of the potential field values and their negative gradients. In manufacturing or similar applications, vehicles need to deliver cargo between several defined destinations. For each destination, a suitable navigation function can be precomputed, which increases the computational efficiency.

Good trajectories are also obtained when avoiding collisions with multiple robots. Other robots or moving objects are only considered locally within the prediction horizon. Choosing a minimum horizon (h_{min}) ensures that the robot navigates safely while moving past other robots and, in the worst case, stops to avoid a collision. Extending the prediction horizon (e.g., to $2h_{min}$ or more) allows the robots to navigate safely without unnecessary emergency stops to prevent collision. Since local information is considered, optimality is not guaranteed, although good results are obtained in practice. The robot could navigate into a narrow corridor (following the negative gradient) based on the static navigation function, although there is not enough space to avoid collision with another vehicle. MPC will try to follow the direction that reduces the potential while searching among the possible trajectories to avoid collision with another approaching vehicle. In the worst case, if there is not enough free space, the robots would safely stop before a collision occurs.

It would also be possible to globally take into account information about other moving objects and re-create the navigation function. This would require dynamic replanning of the discrete potential field, which is computationally more challenging. However, since the future positions of other objects are not known in advance (they can only be predicted within the sensor's field of view), the navigation function would need to be modified early enough for the robot to safely follow the modified negative gradient of the re-planned potential, which would prevent a collision. The same requirements apply to the length of the prediction horizon as for the static potential field (the horizon must be long enough according to h_{min}). Additionally, the resolution of the discrete grid would need to be fine (much smaller than the size of the robot) to allow for more accurate updating for detected moving obstacles, which is important for narrow passages. A finer resolution of the grid would further increase the computational complexity.

7. Conclusions

In this work, we proposed a novel navigation function obtained from a discrete graph search and smoothed by bicubic interpolation. The navigation function has no local minima and decreases monotonically in the direction of a target, allowing a mobile robot to safely navigate from an arbitrary initial configuration to a desired target. For environments where a set of desired targets is known and fixed, such as on the shop floor or in a warehouse, the appropriate navigation functions can be precomputed. This allows for computationally efficient navigation with rather modest memory requirements. The navigation function is coupled with model predictive control (MPC), which extends navigation to multiple robots and introduces variable horizon and combined stochastic and deterministic search in the optimization to improve performance. Coordination of multiple vehicles is solved locally in MPC as a constrained optimization problem where the cooperating vehicles must share their trajectories in the horizon, while for other objects the trajectories must be estimated from observations. The applicability of the proposed solutions is illustrated by several simulations and experiments. In the future, we will explore alternative approaches for interpolating navigation function. In addition, coordination overhead could be reduced by introducing traffic guidelines and a one-way option in the navigation function, which would improve performance and reduce coordination overhead in narrow corridor areas.

Author Contributions: Conceptualization, methodology, and software, G.K. and M.S.; writing—original draft preparation, G.K.; writing—review and editing, G.K. and M.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Slovenian Research Agency and Epilog d.o.o. (research core funding no. P2-0219 and L2-3168) and partially by the Croatian Ministry of Science and Education through the European Regional Development Fund (DATACROSS) (grant KK.01.1.1.01.0009).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was partially supported by the Slovenian Research Agency and Epilog d.o.o. (research core funding no. P2-0219 and L2-3168) and partially by the Croatian Ministry of Science and Education through the European Regional Development Fund (DATACROSS) (grant KK.01.1.1.01.0009).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jahn, U.; Heß, D.; Stampa, M.; Sutorma, A.; Röhrig, C.; Schulz, P.; Wolff, C. A taxonomy for mobile robots: Types, applications, capabilities, implementations, requirements, and challenges. *Robotics* **2020**, *9*,109. [CrossRef]
2. Gonzalez-Aguirre, J.A.; Osorio-Oliveros, R.; Rodríguez-Hernández, K.L.; Lizárraga-Iturralde, J.; Morales Menendez, R.; Ramírez-Mendoza, R.A.; Ramírez-Moreno, M.A.; Lozoya-Santos, J.d.J. Service Robots: Trends and Technology. *Appl. Sci.* **2021**, *11*,702. [CrossRef]
3. Oyekanlu, E.A.; Smith, A.C.; Thomas, W.P.; Mulroy, G.; Hitesh, D.; Ramsey, M.; Kuhn, D.J.; Mcghinnis, J.D.; Buonavita, S.C.; Looper, N.A.; et.al. A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications. *IEEE Access* **2020**, *8*, 202312–202353. [CrossRef]
4. Fragapane, G.; de Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
5. Wood, L. Global AGV (Automated Guided Vehicles) and AMR (Autonomous Mobile Robots) Market Forecast to 2026: Contains Analysis of More Than 500 Players. 2021. Available online: https://www.globenewswire.com/news-release/2021/01/13/2157658/0/en/Global-AGV-Automated-Guided-Vehicles-and-AMR-Autonomous-Mobile-Robots-Market-Forecast-to-2026-Contains-Analysis-of-More-Than-500-Players.html (accessed on 15 January 2022)
6. Digani, V.; Sabattini, L.; Secchi, C.; Fantuzzi, C. Ensemble coordination approach in multi-agv systems applied to industrial warehouses. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 922–934. [CrossRef]
7. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66. [CrossRef]
8. Standley, T. Finding optimal solutions to cooperative pathfinding problems. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; Volume 24, pp. 173–178.
9. Zhang, Z.; Guo, Q.; Chen, J.; Yuan, P. Collision-free route planning for multiple agvs in an automated warehouse based on collision classification. *IEEE Access* **2018**, *6*, 26022–26035. [CrossRef]
10. Pallottino, L.; Scordio, V.G.; Bicchi, A.; Frazzoli, E. Decentralized cooperative policy for conflict resolution in multivehicle systems. *Trans. Rob.* **2007**, *23*, 1170–1183. [CrossRef]
11. Stentz, A. Optimal and efficient path planning for partially-known environments. In Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 3310–3317.
12. Čikeš, M.; Đakulović, M.; Petrović, I. The path planning algorithms for a mobile robot based on the occupancy grid map of the environment – A comparative study. In Proceedings of the Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on. IEEE, Sarajevo, Bosnia and Herzegovina, 27–29 October 2011; pp. 1–8.
13. Zdešar, A.; Škrjanc, I. Optimum velocity profile of multiple bernstein-bézier curves subject to constraints for mobile robots. *ACM Trans. Intell. Syst. Technol.* **2018**, *9*, 1–23. [CrossRef]
14. Klančar, G.; Seder, M.; Blažič, S.; Škrjanc, I.; Petrović, I. Drivable path planning using hybrid search algorithm based on e* and bernstein-bézier motion primitives. *IEEE Trans. Syst. Man, Cybern. Syst.* **2021**, *51*, 4868–4882. [CrossRef]
15. Gim, S.; Adouane, L.; Lee, S.; Dérutin, J.P. Clothoids composition method for smooth path generation of car-like vehicle navigation. *J. Intell. Robot. Syst. Theory Appl.* **2017**, *88*, 129–146. [CrossRef]
16. Klančar, G.; Blažič, S. Optimal Constant Acceleration Motion Primitives. *IEEE Trans. Veh. Technol.* **2019**, *68*, 8502–8511. [CrossRef]
17. Ghilardelli, F.; Gabriele, L.; Piazzi, A. Path generation using $\eta 4$-splines for a truck and trailer vehicle. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 187–203. [CrossRef]
18. Draganjac, I.; Miklić, D.; Kovačić, Z.; Vasiljević, G.; Bogdan, S. Decentralized control of multi-agv systems in autonomous warehousing applications. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1433–1447. [CrossRef]
19. Li, Y.; Littlefield, Z.; Bekris, K.E. Asymptotically optimal sampling-based kinodynamic planning. *Int. J. Robot. Res.* **2016**, *35*, 528–564. [CrossRef]
20. Pivtoraiko, M.; Knepper, R.; Kelly, A. Differentially constrained mobile robot motion planning in state lattices. *Robot. Auton. Syst.* **2009**, *26*, 308–333. [CrossRef]
21. Likhachev, M.; Ferguson, D. Planning long dynamically-feasible maneuvers for autonomous vehicles. *Int. J. Robot. Res.* **2009**, *28*, 933–945. [CrossRef]
22. Montemerlo, M.; Becker, J.; Bhat, S. ; Dahlkamp, H. ; Dolgov, D. ; Ettinger, S. ; Hähnel, D. ; Hilden, T. ; Hoffmann, G. ; Huhnke, B.; et al. Junior: The Stanford entry in the Urban Challenge. *J. Field Robot.* **2008**, *25*, 569–597. [CrossRef]
23. Wooden, D.; Malchano, M.; Blankespoor, K.; Howardy, A.; Rizzi, A.A.; Raibert, M. Autonomous navigation for BigDog. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–8 May 2010; pp. 4736–4741. [CrossRef]

24. Bagnell, J.A.; Bradley, D.; Silver, D.; Sofman, B.; Stentz, A. Learning for autonomous navigation. *IEEE Robot. Autom. Mag.* **2010**, *17*, 74–84. [CrossRef]
25. Philippsen, R.; Siegwart, R. An interpolated dynamic navigation function. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, 18–22 April **2005**, Volume 4, pp. 3782–3789.
26. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **2010**, *29*, 485–501. [CrossRef]
27. Lu, H.; Yin, Y. Fast path planning for autonomous ships in restricted waters. *Appl. Sci.* **2018**, *8*, 1–24. [CrossRef]
28. Cosio, F.A.; Castaneda, M.P. Autonomous robot navigation using adaptive potential fields. *Math. Comput. Model.* **2004**, *40*, 1141–1156. [CrossRef]
29. Guerra, M.; Efimov, D.; Zheng, G.; Perruquetti, W. Avoiding local minima in the potential field method using input-to-state stability. *Control. Eng. Pract.* **2016**, *55*, 174–184. [CrossRef]
30. Park, M.G.; Lee, M.C. A new technique to escape local minimum in artificial potential field based path planning. *Ksme Int. J.* **2003**, *17*, 1876–1885. [CrossRef]
31. Ogren, P.; Leonard, N.E. A convergent dynamic window approach to obstacle avoidance. *IEEE Trans. Robot.* **2005**, *21*, 188–195. [CrossRef]
32. Klančar, G.; Seder, M. Combined stochastic-deterministic predictive control using local-minima free navigation. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 25–29 October, **2021**, pp. 5788–5793. [CrossRef]
33. Klančar, G.; Mušič, G.; Chen, H.; Seder, M. Wheeled robot navigation based on a unimodal potential function. In Proceedings of the 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 28–30 August 2019, pp. 1–5. [CrossRef]
34. Borkowski, P. Computational mathematics in marine navigation. *Sci. J. Marit. Univ. Szczecin, Zesz. Nauk. Akad. Morskiej Szczecinie* **2010**, *21*, 20–27.
35. Kiss, D.; Tevesz, G. A receding horizon control approach to obstacle avoidance. In Proceedings of the 2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 19–21 May 2011, pp. 397–402.
36. Seder, M.; Baotić, M.; Petrović, I. Receding horizon control for convergent navigation of a differential drive mobile robot. *IEEE Trans. Control. Syst. Technol.* **2017**, *25*, 653–660. [CrossRef]
37. Demesure, G.; Defoort, M.; Bekrar, A.; Trentesaux, D.; Djemai, M. Decentralized motion planning and scheduling of AGVs in FMS. *IEEE Trans. Ind. Inf.* **2017**, *14*, 1744–1752. [CrossRef]
38. Brito, B.; Floor, B.; Ferranti, L.; Alonso-Mora, J. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 4459–4466. [CrossRef]
39. Seder, M.; Petrovic, I. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 1986–1991. [CrossRef]
40. Keys, R. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoust. Speech Signal Process.* **1981**, *29*, 1153–1160. [CrossRef]