actuators

Article

An Optimization-Based High-Precision Flexible Online Trajectory Planner for Forklifts

Yizhen Sun [1,2,*], Junyou Yang [1], Zihan Zhang [1] and Yu Shu [1]

1 School of Electrical Engineering, Shenyang University of Technology, Shenyang 110020, China
2 Intelligent Robot Laboratory, Shenyang Open University, Shenyang 110020, China
* Correspondence: sunyizhen@smail.sut.edu.cn

Abstract: There are numerous prospects for automated unmanned forklifts in the fields of intelligent logistics and intelligent factories. However, existing unmanned forklifts often operate according to offline path planning first followed by path tracking to move materials. This process does not meet the needs of flexible production in intelligent logistics. To solve this problem, we proposed an optimized online motion planner based on the output of the state grid as the original path. Constraints such as vehicle kinematics; dynamics; turning restriction at the end of the path; spatial safety envelope; and the position and orientation at the starting point and the ending point were considered during path optimization, generating a precise and smooth trajectory for industrial forklifts that satisfied non-holonomic vehicle constraints. In addition, a new rapid algorithm for calculating the spatial safety envelope was proposed in this article, which can be used for collision avoidance and as a turning-angle constraint term for path smoothing. Finally, a simulation experiment and real-world tray-insertion task experiment were carried out. The experiments showed that the proposal was effective and accurate via online motion planning and the tracking of automated unmanned forklifts in a complicated environment and that the proposal fully satisfied the needs of industrial navigation accuracy.

Keywords: unmanned forklift; online planning; path optimization; safety envelope

check for updates

Citation: Sun, Y.; Yang, J.; Zhang, Z.; Shu, Y. An Optimization-Based High-Precision Flexible Online Trajectory Planner for Forklifts. *Actuators* **2023**, *12*, 162. https://doi.org/10.3390/act12040162

Academic Editors: Luis Payá, Oscar Reinoso García and Helder Jesus Araújo

Received: 13 March 2023
Revised: 26 March 2023
Accepted: 31 March 2023
Published: 4 April 2023

1. Introduction

In recent years, with the rapid development of modern industry, traditional factories have gradually merged with intelligent factories. As a result, unmanned forklifts have been widely implemented for in-plant logistics [1]. An unmanned forklift in a factory is shown in Figure 1.

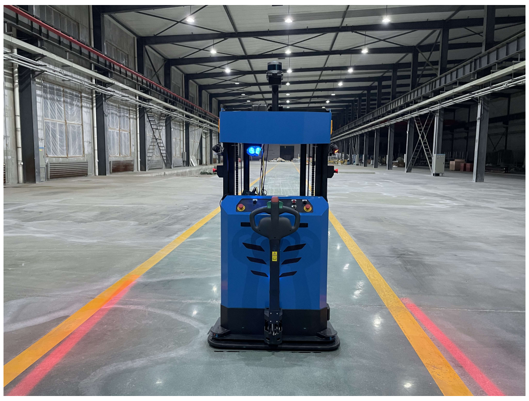

Figure 1. Automated unmanned forklift.

At present, the navigational solutions of industrial unmanned forklifts can only accomplish the automatic navigation of fixed routes, and all the operational tracks are generated offline [2]. If the navigation area changes, the navigation route has to be adjusted, and redeploying the whole navigation system presents a time-consuming, inflexible, and costly procedure. This inflexible deployment mode significantly limits the application of the system.

Therefore, current unmanned forklifts all operate on pre-designed fixed routes. When they arrive at the handling point, they usually employ the "blind fork" method to complete their task. If these forklifts deviate from the fixed routing, these forklifts are unable to fulfill their mission. Furthermore, they have also occasionally caused serious accidents [3]. In this off-line path generation method, even if the deviation of the final posture can be detected by the sensor, it is challenging for the vehicle controller to correct the deviation within a relatively short distance. Therefore, it is difficult for current unmanned forklifts to meet the handling requirements of flexible production required at an industrial scale, which significantly limits the application of unmanned forklifts in intelligent digital factories.

This study focuses on the precise flexible online trajectory planning of industrial unmanned forklifts. In order to solve issues regarding the inflexible deployment of the forklift navigation system and overcome the limitations of flexible online trajectory planning in modern factories, we proposed and evaluated a precise flexible online trajectory planner for industrial forklifts. The proposal can be used for online motion planning of car-like vehicles regardless of the end-pose. In this method, the original path output of the motion planner based on sampling is used to smooth the path based on optimization, which can then achieve a smooth and collision-free trajectory online and improve the pose accuracy at the end of the trajectory.

This study contributed the following innovations: (1) We proposed a new path smoothing approach based on a two-step method that can be used in a complete online navigation system to plan high-precision driving trajectories for car-like vehicles. Firstly, a feasible primitive path was found, and then the optimal control function was constructed. The security envelope of each point in the primitive path, the kinematics constraints, the dynamics constraints, and the turning constraint at the end of the path were used as the constraint conditions of the objective function, and the path smoothing was carried out using an optimization method. This method not only effectively improved the accuracy of path planning, but also eliminated the additional curvature and collision safety reviews typically required after path smoothing, improving the efficiency and safety of path planning. (2) A new, rapid calculation method of a spatial security envelope was proposed. These safety envelopes added the attitude and angle constraints of the vehicle while considering the positional collision avoidance constraints, which limited the search space of the robot to a collision-free state.Firstly, an off-line calculation method was used to pre-calculate the spatial safety envelope set of the vehicle according to the forklift model. The envelope set considered both the area of the polygon envelope and the range of the rotation angle. It then generated a lookup table (PC: polygon constraint) of the envelope set via the weight relationships. Secondly, based on the collision-free path $\hat{\zeta}$ (which satisfied the kinematic constraints) output by the lattice motion planner, each waypoint in the path was generated in a spatial security envelope, and finally the spatial security envelope around the whole path was obtained. See Appendix A for details of the summary of the variable comparison table.

2. Related Work

At present, unmanned forklifts used in factories operate according to predefined trajectories, which are typically generated by manual definitions or manually by operators [4]. Although these methods are simple in principle, the trajectories generated by these methods bear issues that warrant being addressed. These include their high deployment costs and lack of flexibility in adapting to environmental alterations in factory settings, such as adding goods to assembly points or modifying distribution routes in warehouses. To resolve such problems, a new trajectory planner is required for the flexible online trajectory generation of industrial forklifts.

At present, researchers have suggested numerous motion-planning methods. Motion planning is usually divided into two categories: front-end path search and back-end path optimization. The front-end path search is responsible for identifying a safe and collision-free path on the map, and then the back-end path optimization considers the kinematics, dynamics, and environmental factors of vehicles as constraints; optimizes the waypoints on the initial path in terms of smoothness and safety; and then generates a smooth path. Therefore, the motion-planning method combining a front-end path search with a back-end path optimization has become the standard in motion planning.

The following provides a detailed introduction to the literature on related topics. The existing front-end path-search methods for industrial unmanned forklifts include graph-based search methods and sampling-based methods. These methods usually discretize the continuous state space into a node graph and then search for effective links from the starting node to the target node in the graph. Path search methods based on graph search mainly include A* [5,6], Dijkstra [7], and so on. The path-search method based on sampling is divided into two steps, namely, sampling the state space and the control space. Typical methods for sampling a state space include the state lattice method [8,9] and rapidly exploring random-tree sequence [10,11]. Therefore, typical control space sampling methods have included the hybrid A algorithm [12] and dynamic window method [13]. Specifically, Dang et al. [12] suggested the hybrid A-star algorithm, which combined front-end searching with a vehicle kinematics model, and considered vehicle kinematics constraints while searching. Subsequently, this not only improved search efficiency but also satisfied the vehicle model with an under-actuated structure, such as a forklift truck. It was used to solve the vehicle planning problem in parking lots, U-turns, and other scenarios. However, the path searched by this kind of algorithm was not an optimal path in terms of vehicle kinematics constraints. Therefore, during the search process, the algorithm only focused on a localized optimal solution, rather than a global optimal solution. Pivtoraiko and Kelly [14] suggested the concept of the state lattice search, which was used to build an efficient dynamic motion planner. However, the algorithm state lattice search was very expensive and difficult to build online. Therefore, the path generated by this method had the disadvantage of a discontinuous state space, which failed to meet the high-precision parking demands when used alone.

Back-end path optimization is divided into special curve-smoothing methods and optimization methods [15]. (1) The special curve-smoothing method employs the waypoint obtained by the path search [16,17] as input, and then uses a special curve to generate a feasible smooth path. Commonly used curves have included the Dubins curve [18], the Bezier curve [19], the B-spline curve, and so on. It was necessary to change the original path by using the above special curve method directly, which the collision-free path could not guarantee. Therefore, an extra processing step was required to determine whether the path in the new representation was still conflict-free. This proved to be time-consuming and negatively affected the speed of online solutions. (2) The trajectory planning task was described as an optimal control problem (OCP) based on an optimization method, and was discretized into a nonlinear-programming (NLP) problem [20]. Functional optimization method obtained smooth paths by minimizing constraints such as velocity, acceleration, and jerk. Zhu et al. [21] proposed a trajectory-smoothing method based on convex optimization, which was used for trajectory smoothing and speed optimization of mobile robots with similar automobile dynamics. In addition, Choi et al. [22] proposed a path-planning algorithm for non-holonomic, constrained vehicles operating in a semi-structured environment. Sprunk et al. [23] considered the smoothness of the path while ignoring security constraints during optimization; therefore, it was necessary to evaluate the security after optimization.

Others have used security envelope constraints during path optimization, which limited the movement range of each waypoint. However, the security envelope was of the same size and lacked flexibility [24]. Furthermore, the authors of [25] used a variable-sized safety envelope, that is, the size of the safety envelope was dynamically generated according to the distance between the vehicle model and the obstacle. However, this method did not consider the rotation angle of the vehicle on the safety envelope, especially for a vehicle body with an under-driven structure, such as a forklift, for which the angle constraint was equally important. In summary, the extraction of safety envelopes has proven to be a time-consuming process. To achieve online path generation, rapid path-safety-envelope extraction is critical.

This study provided various insights.First, we used the anytime repairing A* (ARA*) algorithm to search a collision-free feasible path under finite time constraints. This method ensured that the reference trajectory could be quickly obtained within a finite time period. The state grid map was reasonably constructed according to the real-time positioning and orientation of the forklifts. Second, a path-smoothing method was proposed based on optimizations including the safety envelope, rotation angle constraints, kinematics, and dynamics constraints. This method directly added state-space constraints during the smoothing process, which eliminated a second verification step after smoothing. Third, since the generation of the secure envelope has been problematic due to being time-consuming and inflexible, we designed a rapid and secure envelope extraction method that considered both positioning and angle constraints. Lastly, using the B-spline curve method, the optimized path was guided by time parameters to ensure a safe and smooth trajectory that met the kinematics and dynamics constraints. In the experiment, the industrial unmanned forklifts using the flexible online trajectory planner proposed in this paper generated smooth, drivable trajectories for any target pose within a few seconds.

3. Methodology

This paper focused on the development of high-precision flexible online trajectory planning for industrial unmanned forklifts. A motion planning algorithm was proposed based on optimization. We used a five-step method, as shown in Figure 2.

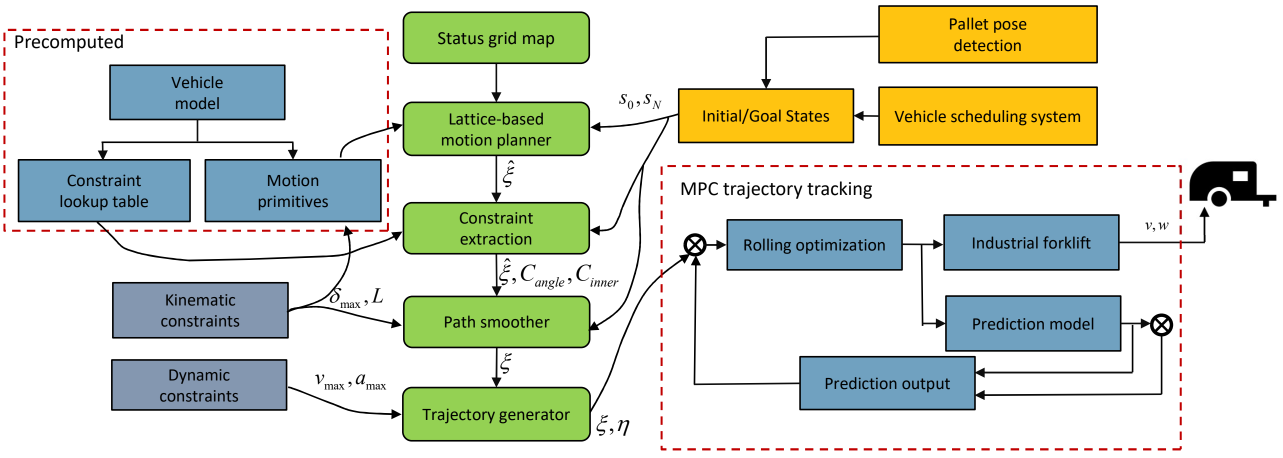

Figure 2. Navigation system framework. The variables s_0 and s_N indicate the starting pose and target pose of the vehicle, respectively; $\delta_{\max}$ indicates the maximum steering angle; L indicates the distance between the front and rear axles of the forklift truck, $v_{\max}$ and $a_{\max}$, respectively, as constraints on velocity and acceleration in the forward direction. The ξ and η denote two collection of points in the state space and the control space, respectively, C_{angle} is the angle constraint, and C_{inner} is the inner constraint. A pre-computed constraint lookup table was used to extract a set of position and angle inequality constraints for the vehicle model.

First, we used the lattice-based motion planner to calculate the trajectory path that would satisfy the forklift kinematics constraint, which was the basis of the path constraint step. Second, we pre-calculated the safety envelope set of the vehicle offline, according to the forklift model, and used the offline method to calculate the envelope set to improve the efficiency of the trajectory optimization steps. Next, we extracted the free-space envelope around the path according to the original path and the calculated space envelope set, and expressed it in the form of a convex polyhedron constraint of the vehicle state. The path space envelope generated in this step was used as a security constraint item for the subsequent path-smoothing step. The path-smoothing based on the optimization method proposed in this paper was then used to address the path and the corresponding constraints, and a collision-free continuous path was obtained. Finally, the trajectory generator used a B-spline interpolation method to generate a safe and smooth trajectory that satisfied the dynamic and time constraints. A model-based predictive control algorithm was used to track the generated trajectory with high precision. Since this study focused on safety envelope extraction and path optimization, the additional details in Figure 2 are briefly described for the completeness of the algorithm description.

3.1. Establishment of a Kinematic Model of the Forklift Truck

Because of the low speed of forklifts in logistics transportation, the lateral forces of forklifts are low. Assuming that the forklift would not skid during operation, we only needed to establish the forklift kinematic model to complete the trajectory planner design. Assuming that the forklift body was rigid and the deformation of wheels could be ignored, the forklift motion would not pitch or roll on the plane, and the contact point between the wheels and the ground would only produce rolling contact without sliding. The kinematic model of the forklift is shown in Figure 3.

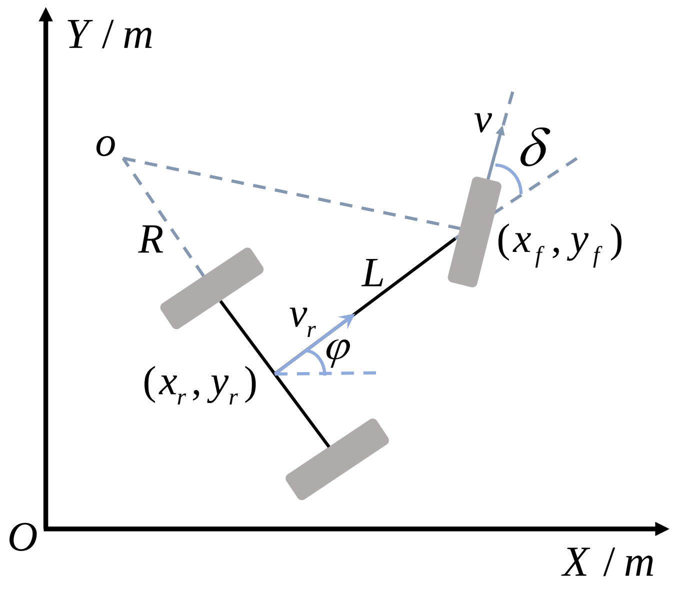

Figure 3. Kinematics model diagram of the forklift truck.

In Figure 3, XOY is the world coordinate system; (x_f, y_f) represents the forklift front-wheel coordinates; (x_r, y_r) represents the center rear-wheel coordinates; o represents the AGV forklift instantaneous rotation center; L represents the AGV front-and-rear wheelbases; v represents the steering wheel linear speed; w represents the steering wheel angular speed; v_r represents the vehicle's instantaneous linear speed; w_r represents the vehicle's instantaneous angular speed; δ and φ represent front steering wheel angle and vehicle heading angle, respectively; and R represents the turning radius of vehicle body. The turning radius of fork AGV was determined by the following:

$$R = \frac{L}{\tan \delta} \tag{1}$$

The kinematic relation is expressed as follows:

$$\begin{cases} \dot{x} = v_r \cos\varphi \\ \dot{y} = v_r \sin\varphi \\ v_r = v\cos\delta \\ \dot{\varphi} = \frac{v_r}{R} = v_r \frac{\tan\delta}{L} \\ \dot{\delta} = w \end{cases} \tag{2}$$

Assuming that the control space of the forklift $c = [v, w]$, the state space $s = [x, y, \varphi, \delta]$ and $\dot{s} = f(s, c)$ was the state transition of the forklift equation. The reference position point (x, y) of the forklift was in the two-dimensional space, as well as the heading angle φ, and the steering angle δ, and then ξ and η were forklift state space and control space A set of N discrete points, respectively, as follows:

$$\xi = \{s_i\} = \{(x_i, y_i, \varphi_i, \delta_i)\} \tag{3}$$

$$\eta = \{c_i\} = \quad \{(v_i, \omega_i)\} \tag{4}$$

Given the initial state of the forklift $s_0 = (x_0, y_0, \varphi_0, \delta_0)$ and the target state $s_N = (x_N, y_N, \varphi_N, \delta_N)$, a set of N state points ξ and corresponding control points η needed to be calculated that could control the forklift's movements from s_0 to s_N, without colliding with any known obstacles.

3.2. State Grid Map

After establishing the kinematic model of the forklift and expanding the obstacles, the forklift control system had to generate all feasible paths that satisfied the kinematic constraints of the forklift. At present, the path obtained by motion planning based on the grid map cannot consistently meet the kinematics requirements of the forklift. Therefore, our forklift path-planning used a state grid map [26]. As shown in Figure 4, the state grid map was based on the ordinary grid map, and added the constraints of the kinematic model of the forklift to ensure that the forklift could travel along the path generated between two adjacent points. As compared to the ordinary grid map, in the state lattice of the state grid map, the connection lines of each vertex were generated according to the kinematic model, and they were all feasible paths [27].

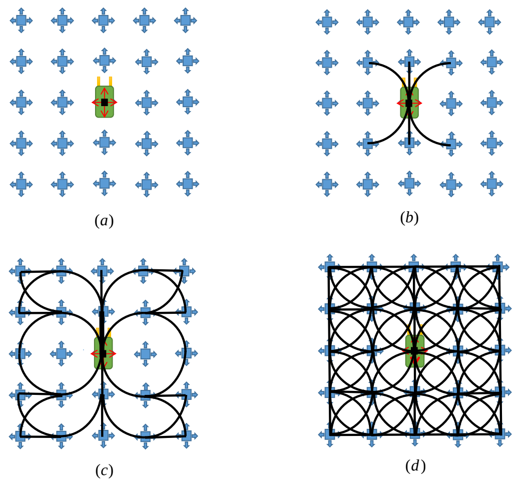

Figure 4. Constructing the lattice for the forklift. (**a**) A discretization in C-space was defined; (**b**) feasible path for 8 neighbor nodes around the origin 8 neighbor nodes around the origin; (**c**) feasible paths extended outward to 24 neighbors, only a few direct paths shown; (**d**) complete lattice.

In our thesis, we construct a 4D state lattice based on the position (x,y), heading (φ), curvature (k) for the kinematics model of the forklift. In order to simplify the construction process, we use a special discretization of state and control space [28]. The orientation of the forklift is assumed to be uniformly distributed in space [0,2π], and the discrete curvature K is also uniformly distributed between $[-k_{\max}, k_{\max}]$, where $k_{\max} = \frac{1}{R_{\min}}$. Figure 4 shows the accessible graph for the forklifts with three actions. Due to the random position of the forklifts in autonomous navigation space, the state lattice is required to have spatial repeatability, motion primitives generated at a certain node can be applied to other nodes too. The motion primitives in Figure 4 are generated by copying motion primitives from the starting point to other nodes, so the global paths generated by the control system are also feasible for the forklift. Through the generation of motion primitives in the state lattice, the global path planning problem is transformed into the path search and decision problem in different motion primitives, and then generate the feasible trajectory. Combined with the commonly used ARA* search algorithm, we can then obtain a path from the start point to the target that satisfies the kinematic constraints for the forklifts.

3.3. ARA Path Search*

Among the traditional path search algorithms, the A* algorithm is widely used in industrial fields because it can effectively solve the shortest path within a short time period. However, the heuristic functional construction of the A* algorithm was relatively simple, and there were more redundant nodes in the execution process of the algorithm, which were time-intensive and, thus, expensive to resolve. In order to achieve a rapid online solution, we employed the ARA* algorithm to search the path on an expanded raster map. The algorithm first found an optimal non-global feasible path under a given relaxation condition and then gradually strengthened the constraints in the remaining time, continuously improving the sub-optimal solution by using the searched node information. If it had sufficient time, the algorithm would find an optimal global feasible path. The algorithm reused the previous search results in the process of path optimization. The cost estimation function of the ARA* algorithm was improved, as compared to the A* algorithm. The cost estimation function was expressed as follows:

$$f(s) = g(s) + \varepsilon * h(s) \tag{5}$$

where ε is the expansion factor, s represents the current node, f(s) is the cost from the starting point to the end point, g(s) is the actual cost from the starting point of the path to the current node, h(s) is the cost from the current node to the end point cost estimate. During the execution of the algorithm, the expansion factor ε gradually decreases from the set initial value. If the algorithm can run for enough time, then finally $\varepsilon_{\min} = 1$. If the runnable time is insufficient, then ε will be as close to 1 as possible, and subsequently the obtained path is a suboptimal solution.

In this paper, the ARA* algorithm was used to search for a collision-free path $\hat{\xi}$ in the state grid that satisfied the kinematics and time constraints for forklift travel, and the algorithm obtained the optimal solution within the time constraints.

3.4. Offline Pre-Computed Security Envelope Set

The safety envelope was an artificially determined convex polyhedron area that wrapped around the initial path, and the various poses of the robot in this area had to satisfy the kinematic model of the vehicle. During the path optimization step, the safety envelope was used for collision-avoidance constraints, so that the points on the path could be adjusted within the safety envelope. While calculating the optimal solution for local path optimization, this ensured that the optimized path avoided collisions. The correct setting of the safety envelope eliminated the safety hazards caused by the robots colliding at a smooth point due to trajectory optimization. However, safety envelope extraction has typically been a difficult and time-consuming process and has not been able to fulfill the needs of online trajectory generation for forklifts. In this study, a new algorithm for the rapid computation of path-space security envelopes was proposed. The algorithm divided the extraction of safety envelope constraints into two steps: an offline pre-computed safety envelope set and an online path-envelope extraction, both of which greatly improved the efficiency of safety envelope extraction.

The process of the entire security envelope extraction algorithm is shown in Figure 5. After the path-planning module searched for an initial collision-free path, the pre-computed envelope set was used to extract constraints for each path point in collision-free path. Next, the safety envelopes of each position point in the path were superimposed to obtain the overall safety envelope [29] of the path, which was then used as the optimization safety obstacle avoidance constraint for the next path. This section introduces the offline security envelope set calculation. The extraction process of the online path security envelope is detailed in the following subsections.

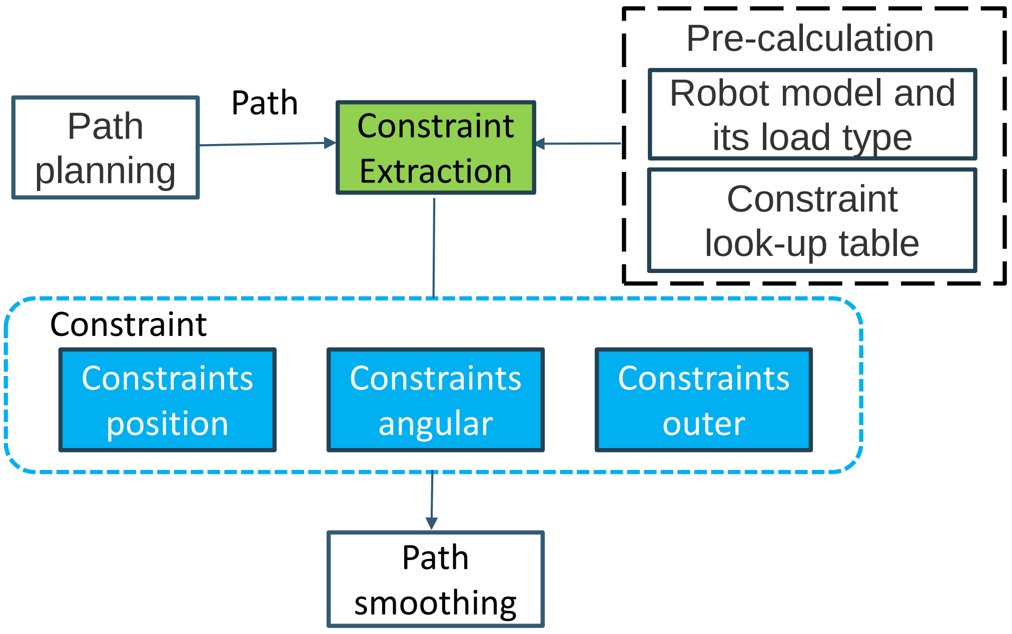

Figure 5. Block diagram of the safety envelope constraint extraction system: After the path-planning module had searched for a collision-free path, it extracted the safety envelope constraint of the path, and the extracted envelope was used as a constraint item for path smoothing. The offline pre-calculation module included the calculation of the robot safety envelope set and the generation of the lookup table.

The pre-calculation module employed in this paper generated different robot envelopes according to different forklift models and corresponding load models, and also calculated a pre-set internal position constraint C_{inner} and angle constraint C_{angle} that were stored in the table. The size of the inner constraint C_{inner} in the safety envelope constraint set PC was the rectangle formed by the linear combination of the displacement of the forklift reference point in four directions; forward, backward, left, right, in addition to angle. The angle constraint C_{angle} was the linear combination of the left and right offsets based on the direction of the forklift reference point. We arranged and combined the internal constraints C_{inner} and the angle constraints C_{angle} freely and, then, calculated the area and angle constraints C_{angle} of each set of internal constraints C_{inner}. We used their product as the weight w of each constraint set and stored them in the security envelope constraint set PC, in order of weight. Finally, according to the differences of the forklift envelopes, the outer constraints C_{outer} generated by each pair of constraints were calculated. As shown in Figure 6, the two graphs display the combination of the two kinds of internal constraints C_{inner} (blue rectangle) and angle constraints C_{angle} (red arrow and green arrow) in the lookup table, which generated a schematic diagram of the outer constraint C_{outer} (yellow polygon). The algorithm flow is shown in Algorithm 1.

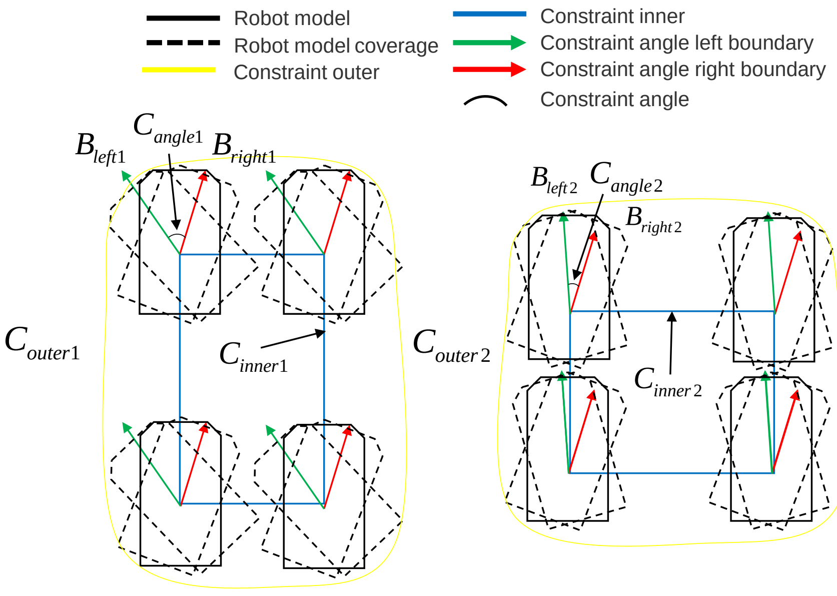

Figure 6. Offline pre-computed spatial security envelope set: Figure **left** and **right** show two examples of security envelopes, respectively. Among them, C_{angle} is the angle constraint, B_{left}, B_{right} is the boundary of the angle constraint, C_{inner} is the inner constraint, C_{outer} is an external constraint. Different combinations of inner and angle constraints resulted in different outer constraints.

Algorithm 1 Lookup table PC calculation.

```
Input:  Lookup table: $PC$
        x Distance deviation: n1,n2,n3,n4,n5,n6,n7
        y Distance deviation: m1,m2,m3,m4,m5,m6,m7
        Angle deviation: $a1, a2, a3, a4, a5, a6, a7$
        Item: $i = j = k = 1$
1:  Set constraint center $(x, y)$
2:  while $trajectory.get()$ do
3:      for $i \leq 7$ do
4:          Set rectangle angle deviation is $a1$
5:          Generate $C_{angle}$ of element in $PC$
6:          for $i \leq 7$ do
7:              Set rectangle length is $(x - ni, x + nj)$
8:              for $k \leq 7$ do
9:                  Set rectangle width is $(y - nk, y + nk)$
10:                 Generate $C_{inner}$ of element in $PC$
11:                 Generate $C_{outer}$ according to $C_{angle}$ and $C_{inner}$
12:                 Calculate $w$ of $C_{outer}$
13:                 $k++$
14:             end for
15:             $j++$
16:         end for
17:         $i++$
18:     end for
19:     Sort the elements in $PC$ according to $w$
20:     Output $PC$
21:     endwhile
22: end while
```

3.5. Online Computed Path Security Envelope

Based on the pre-computed safety envelope constraint set PC obtained in the previous section, when the starting and ending points of the forklift navigation were provided, the motion planner searched for a kinematically feasible collision-free path. The path-envelope-generation module generated the safety envelope of the path according to the grid-map information, the original path, and the safety envelope constraint set PC. The safety envelope of a path was the collision-free free space region around the path, expressed in the form of a convex polyhedron constraint for the vehicle state, divided into an inner constraint C_{inner} and an angle constraint C_{angle}. Among them, the left boundary B_{left} and the right boundary B_{right} of the angle constraint C_{angle} limited the directions that the forklift, and the internal constraint C_{inner} defined the safe movement range of the forklift reference point during optimization. The constraint generation algorithm proposed in this paper defined a constraint combination formed by a space constrained by different positions and directions at each waypoint and then superimposed angles on the four vertices of the inner constraint C_{inner}. Through the forklift model constraint, the area covered by C_{angle} in all directions obtained the outer constraint C_{outer} of this point.

The definition of the internal constraint C_{inner} is shown in the Equation (6). C^i_{inner} represents the vector between two points, where $i = 0, 1, \ldots .n$, n is the total number of path points. P is the coordinates of the i-th path point, P_2, P_2, P_3, P_4 are the four vertices of the C_{inner} rectangle.

$$\begin{cases} (C^i_{inner}(P_2, P_1)) \bullet (C^i_{inner}(P, P_1)) \geq 0 \\ (C^i_{inner}(P_4, P_1)) \bullet (C^i_{inner}(P, P_1)) \geq 0 \\ (C^i_{inner}(P_4, P_3)) \bullet (C^i_{inner}(P, P_3)) \geq 0 \\ (C^i_{inner}(P_2, P_3)) \bullet (C^i_{inner}(P, P_3)) \geq 0 \end{cases} \tag{6}$$

The definition of the angle constraint C_{angle} is shown in Equation (7), and $\varphi(i)$ is the angle of the path point.

$$C^i_{angle}(B_{left}) < \varphi(i) < C^i_{angle}(B_{right}) \tag{7}$$

The definition of the external constraint C_{outer} is as follows (8):

$$C_{outer} = C^i_{angle} \cup C^i_{inner} \tag{8}$$

The above constraints defined all possible sets of safety envelopes for each waypoint in the original path of the vehicle, without considering obstacles. Next, we introduced the solution of the maximum safe envelope for each waypoint, considering obstacles. For each waypoint, our constraint generation algorithm generated a local map $L - MAP$ at the point, which saved the largest external constraint $PC[0]$ and set the grid $Grid$ covered by the local map $L - MAP$ in $PC[0]$ to Occ and saved these $Grid$ in array O_0. If we relocated the local map $L - MAP$ to each waypoint P and if O_0 covered the obstacle grid on the world map, it would continue to traverse the next element in the security envelope constraint set PC according to the weight w. We followed this method until the O_i of the first uncovered obstacle corresponding to $PC[i]$ in the safety envelope constraint set PC was found, that is, the outer constraint C_{outer} of this waypoint, which is shown in the following figure. Conversely, the maximum external constraint $PC[0]$ was the constraint of the path point P. The algorithm block diagram of the waypoint-finding constraints is shown in Figure 7. The triangle area is the obstacle, the black area is the part covered by the external constraints of the robot, the red dotted line is the local map, the green line represents the largest external constraints in the PC, and the yellow line is the found external constraints. The safety envelope of the entire feasible path was obtained by superimposing the external constraints C_{outer} of all path points. The secure envelope of the path is shown in Figure 8. The algorithm flow is shown in Algorithm 2.

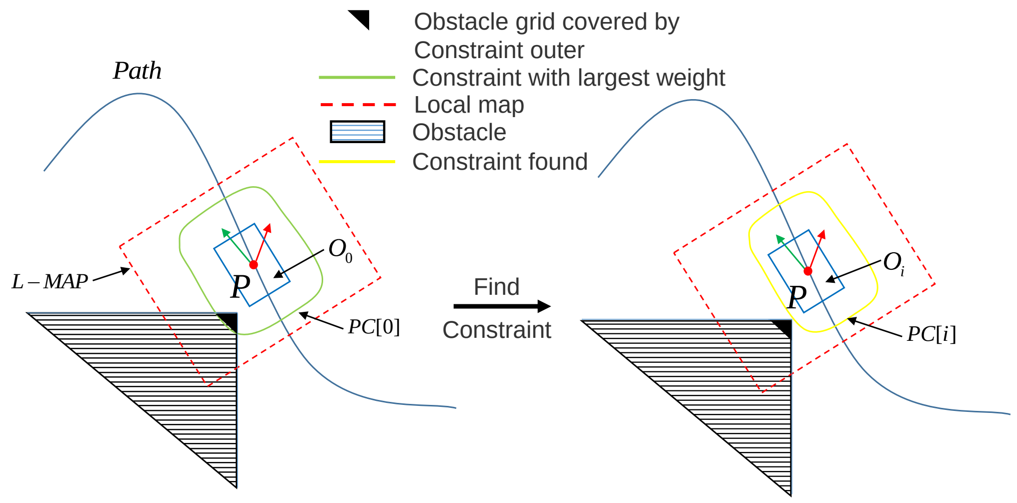

Figure 7. Example of safe online path envelope constraint extraction: **Left**: Generated a local map with maximum outer constraints at this point. **Right**: The constraints of the current point obtained by the constraint search. Among them, $L-MAP$ is the local map, PC is the lookup table, the numbers in square brackets represent the corresponding elements in the array, $PC[0]$ is the largest external constraint in the lookup table, O_0 is the grid covered by $PC[0]$ on $L-MAP$, $PC[i]$ is the found outer envelope, and O_i is $PC[i]$ Grid overlaid on $L-MAP$.

Algorithm 2 Find constraint in path point P.

```
Input:  Lookup table:$PC$
        Item: $i = 0$
 1: Generate local grid map with $PC[i]$
 2: Set $Grid = Occupied$ in $PC[i]$
 3: Transfer $Grid$ in local map to $P$
 4: Get obstacle grid array $O_i$ covered by $PC[i]$
 5: while $PC[i] \neq PC.end()$ do
 6:     for $o$ in $O_i$ do do
 7:         if $o.isobstacle()$ then
 8:             break
 9:         end if
10:         if $o = O_i.end()$ then
11:             Output $PC[i]$
12:         end if
13:         else
14:         Continue
15:         end else
16:         if $PC[i].output()$ then
17:             break
18:         end if
19:         if $PC[i] = PC.end()$ then
20:             Error reporting
21:             break
22:         end if
23:         $i++$
24:         end while
25:     end for
26: end while
```

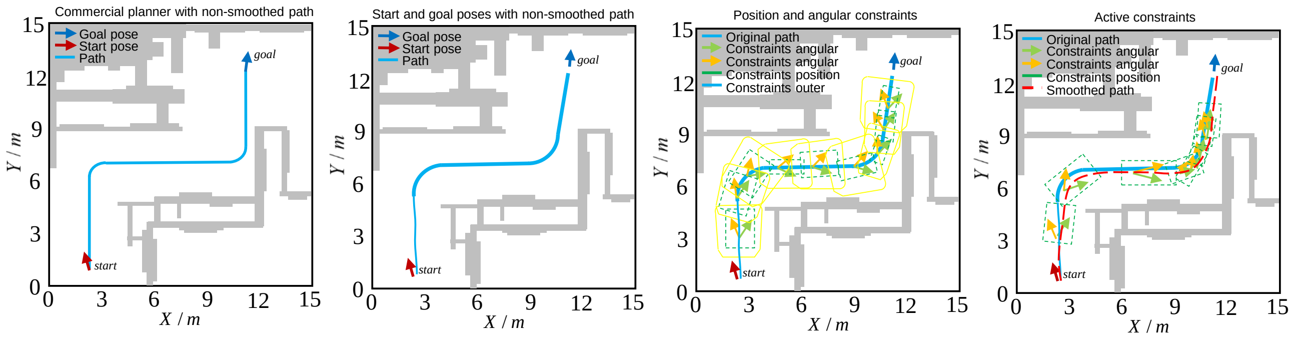

Figure 8. Path-smoothing step. **Left**: Given the start pose and target pose, a commercial motion planner computed the path. **Middle left**: Given a starting pose and a target pose, the motion planner calculated the path $\hat{\xi}$. **Middle right**: Spatial constraint extraction. Position constraints (green box), angular constraints (orange and green arrows) and their spatial combination (yellow line) were computed for each $(\hat{x}_i, \hat{y}_i) \in \hat{\xi}$. For clarity, the figure represents only a subset of state points $(\hat{x}_i, \hat{y}_i) \in \hat{\xi}$. **Right**: Motion constraints drawn on a smooth path.

3.6. Path Smoothing

This section introduces the implementation steps of the path-smoothing method based on optimization, which used the original path searched and considered constraints such as the starting point, target point, kinematics, dynamics, turning limit at the end of the path, and the path safety envelope of the robot, to generate a high-precision smooth path. The detailed process of imposing constraints was as follows. Given the starting point and target point (s_0, s_N), respectively, of the forklift by the task-scheduling system, the motion planner generated a collision-free kinematic path, and the path consisted of a series of state points. However, sampling-based non-holonomic motion-planning methods have generated paths that resulted in discretization errors and discontinuities within the paths themselves or between the stopping point and the goal state. This did not fulfill the requirements of end-pose accuracy required for industrial applications. Because the original path of the ARA* search was from a discretized starting state $\bar{s}_0$ to a discretized target state $\bar{s}_N$, $\bar{s}_0$ and $\bar{s}_N$ represented the state points closest to s_0 and s_N in the state grid map, respectively.

To solve this problem, we modified the starting and target states of the forklift to precisely correspond to the current vehicle state $\bar{s}_0$ and the actual target state $\bar{s}_N$. Additional constraints were imposed during the path optimization function, as shown in Equation (9). Equation (9) effectively addressed the inaccuracy of the initial and final states caused by the discretization in state-grid-based motion planners.

$$\begin{aligned} \bar{s}_0 - s_0 &= 0 \\ \bar{s}_N - s_N &= 0 \end{aligned} \tag{9}$$

In order to ensure that the planned path satisfied the kinematic and dynamic constraints of the forklift, additional constraints were imposed, as shown in Equation (10):

$$\begin{aligned} \delta_{\min} \leq \delta &\leq \delta_{\max} \\ -v_{\max} \leq vs. &\leq v_{\max} \\ -\omega_{\max} \leq \omega &\leq \omega_{\max} \\ -a_{\max} \leq a &\leq a_{\max} \end{aligned} \tag{10}$$

In an industrial forklift with an under-actuated kinematic structure, it is relatively difficult to adjust the lateral deviation while driving. In order to improve parking accuracy, a docking point with a constant attitude is typically added before the parking point to ensure that the vehicle continues in a straight line at the end of the path. The method of adding docking points at the end of the path improved the parking accuracy; however, it also increased the difficulty of operation and reduced efficiency. In order to solve the problems of planning efficiency and parking accuracy at the same time, we imposed steering-angle constraints on the state points at the end of the trajectory, as shown in Equation (11):

$$\delta_i = 0, i = N-1, N \tag{11}$$

It was assumed that the distances between the state points of the path were approximately equal, and the time ΔT between the consecutive state points was set to any fixed positive value. Therefore, the time T through the entire path was set to a constant value $N\Delta T$.The optimal control function of the forklift is shown in Equation (12), and the constraint term of the optimization function is shown in Equation (13), where α is a weighting factor of adjusting the minimum travel distance and minimum rotation of the forklift; $C^i_{angle}(B_{left})$ is the left boundary of the angle constraint of the i path point; $C^i_{angle}(B_{right})$ is the right boundary of the angle constraint of the i path point; b, A_0, A_1 are intermediate variables, such as in Equation (6), C^i_{inner} is the inner constraint of the i path point, P is the coordinates of the waypoint, P_2, P_2, P_3, P_4 are the vertices of the rectangle:

$$\min_{\zeta,\eta} \sum_{i=0}^{N} v_i^2 + \alpha \sum_{i=0}^{N} \omega_i^2 \tag{12}$$

$$\begin{aligned}
&\text{s.t.}\\
&s_{i+1} = \hat{f}(s_i, u_i), i = 0, \ldots, N-1\\
&\bar{s}_0 - s_0 = 0\\
&\bar{s}_N - s_N = 0\\
&\delta_{\min} \le \delta \le \delta_{\max}, i = 0, \ldots, N\\
&C^i_{angle}(B_{left}) < \varphi(i) < C^i_{angle}(B_{right})\\
&(C^i_{inner}(P_2, P_1)) \bullet (C^i_{inner}(P, P_1)) \ge 0\\
&(C^i_{inner}(P_4, P_1)) \bullet (C^i_{inner}(P, P_1)) \ge 0\\
&(C^i_{inner}(P_4, P_3)) \bullet (C^i_{inner}(P, P_3)) \ge 0\\
&(C^i_{inner}(P_2, P_3)) \bullet (C^i_{inner}(P, P_3)) \ge 0
\end{aligned} \tag{13}$$

Our optimization variables were the forklift's front wheel's travel speed v and angular velocity w, and the goal was to minimize the total travel distance and the amount of rotation imposed on the front wheel angle of the forklift. The objective function in Equation (12) minimized the combination of the total distance traveled by the forklift and the total steering wheel rotation.

The optimal control Equation (12) and the constraint item (13) were optimized and calculated as the optimal path-state point s_i. The optimized state point s_i was the obtained smooth path point.

After feeding s_i into a trajectory generator, we used the trajectory generator from Section 3.7 to generate a set of values for each state in the path, given the constraints of the initial and final velocities, steering velocity, steering acceleration, velocity, and acceleration, to allocate the maximum speed.

3.7. Trajectory Generation and Track

After the planner generates a series of discrete state points at fixed intervals, the trajectory generator output a trajectory with a fixed ΔT of 50 ms as the input to the controller [30,31]. As previously mentioned, there are two types of trajectory generation methods typically used in the literature: trajectory generation based on a polynomial function and trajectory generation based on a B-spline curve.

The trajectory generation based on a polynomial function generated the operating trajectory of the forklift by calculating the polynomial coefficients that satisfied the constraints and drew on the polynomial function. However, in order to ensure the safety of the forklift, this method required multiple iterations, which was considerably time-consuming.

The curvature of the B-spline curve was continuous at the nodes of adjacent curve segments, and if the local constraints of the trajectory were not satisfied, the trajectory could be corrected by adjusting the corresponding control points without affecting other trajectory segments [32]. Among them, the cubic B-spline curve exhibits the characteristics of second-order continuity at the nodes, which could optimize the curve and meet the requirements of the acceleration and speed continuity of a moving unmanned forklift. Therefore, the trajectory generator employed in this paper was the cubic B-spline curve.

In this paper, the model predictive control (MPC) method was used to realize trajectory tracking [33–36]. Given a set of state inputs, the controller then used an objective function based on the tracked trajectory to optimize the control output. Since the controller is not the focus of this article, we have not provided the relevant theoretical exposition.

4. Experimental Evaluation

In order to fully evaluate the comprehensive performance of the algorithm proposed in this study, an experimental verification of the algorithm was carried out in simulated and real environments. The navigation algorithm was operated on a personal computer with an Ubuntu 16.04 operating system, an i5-6200u CPU, 8 GB of RAM, and 512 GB storage space. The entire navigation system was implemented within the Robot Operating System (ROS). In the simulation experiment, a forklift model that satisfied the car-like kinematics was created, and a gazebo simulation environment was built. The experiment was equipped with a high-precision locator, the translational positioning accuracy was within 1cm, the rotation angle positioning accuracy was 0.001 radians, and the on-board navigation control system could access the encoder and positioning data. In the real-world environment, the modified industrial unmanned forklift was equipped with three on-board lasers, two of which were safety lasers located on the ground for safe parking in emergency situations. The second navigation laser was an RS-LIDAR-16, a 3D laser rangefinder with a viewing angle of 360 degrees and 16 lines. The communication between the navigation computer and the motor driver through CAN bus was used to collect odometer data and control vehicle motion, and an Ethernet interface was used to transmit LiDAR distance data. Therefore, the navigation computer received encoder and positioning data and sent steering and driving commands, as shown in Figure 9.

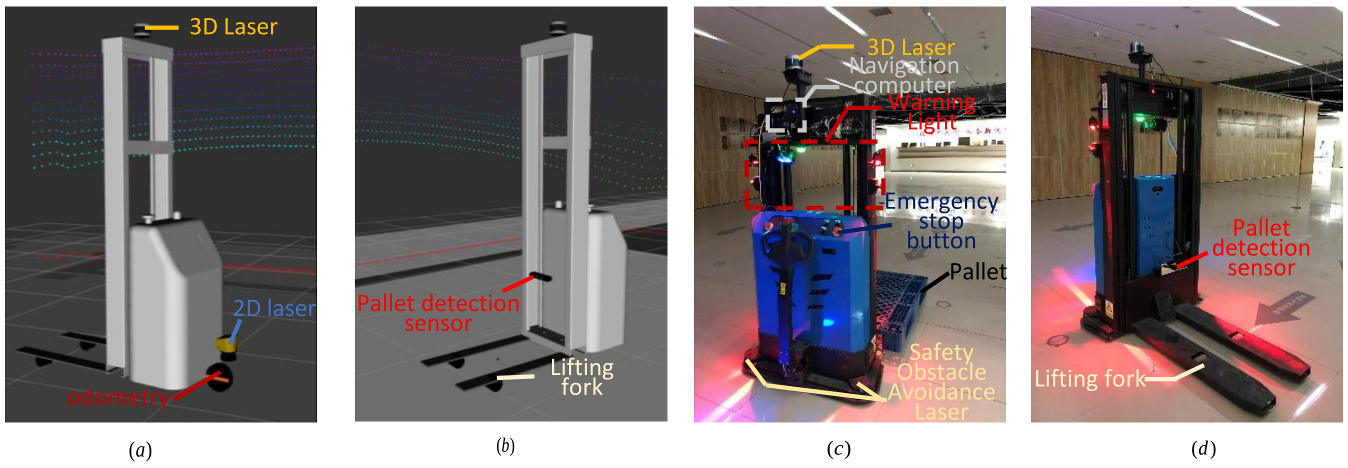

Figure 9. Experimental model. (**a**) Simulation forklift forward view. (**b**) Simulation forklift rear view. (**c**) Forklift forward view. (**d**) Forklift rear view.

The overall goal of our experimental evaluation was to demonstrate the real-time performance, reliability, and accuracy of online-generated trajectories. We divided the experimental evaluation into four parts. (1) Path-smoothing evaluation, which considered the importance and realization of path smoothing; (2) random-target experimental evaluation to further test the robustness of our system on arbitrary target poses and longer distances;

(3) obstacle extension evaluation to analyze the operation of the system in more challenging scenarios; and (4) the evaluation of real pallet fork-and-pick tasks.

For a non-holonomic vehicle, it was difficult to control the direction and the position perpendicular to the motion direction, which was convenient for analyzing the data. We decomposed the errors into forward errors, lateral errors, and heading errors, as shown in Figure 10. For each type of error, we showed the results in terms of mean, standard deviation, and maximum value obtained during the test operation.

4.1. Path-Smoothness Evaluation

In this experiment, the importance and implementation of path smoothing were evaluated. Our focus centered around the accuracy of the online trajectory planning and tracking reaching the accuracy of offline trajectory planning and tracking while fulfilling the requirements of industrial applications. We extracted 10 sets of target poses from hand-defined paths (shown on the left of Figure 8) and used them as targets for a lattice-based motion planner. The output of the motion planner was passed to a path smoother, which was then tracked by the MPC controller. Note that when the path was not processed by the smoother, in order to ensure the parking accuracy, the target state was set as the last point of the trajectory, thus allowing the MPC controller to correct the motion direction caused by the rough path.

The experimental results are shown in Table 1 as a comparison of the results obtained by the motion planner, with and without path smoothing. The error description is shown in Figure 10, which depicts the forward, lateral, and heading errors between the target position and the final parking point. These results confirmed that our system could produce smooth trajectories with excellent end-pose accuracy, with improvements in forward, side, and orientation errors, as compared to no smoothing. Our method could completely replace navigation systems that generate paths manually, without loss of accuracy. (In typical factory settings, the positional accuracy required by the forklift to pick up the pallet was 0.03 m, and the direction accuracy was 0.017 rad [37]). The experimental results further showed that it was necessary to use path smoothing in order to obtain high accuracy of the final parking attitude.

Table 1. Forward and side translation and orientation errors.

	Forward Error			**Side Error**			**Heading Error**		
Method	**Mean (m)**	**Std (m)**	**Max (m)**	**Mean (m)**	**Std (m)**	**Max (m)**	**Mean (rad)**	**Std (rad)**	**Max (rad)**
Planned path without smoothing	0.0288	0.0311	0.0934	0.061	0.021	0.1214	0.0712	0.0613	0.1885
Planned path with smoothing	0.0014	0.0017	0.0055	0.0045	0.0042	0.0167	0.0023	0.0015	0.0098
Ratio increase	95.14%	94.53%	94.11%	92.62%	80.10%	86.24%	96.77%	97.55%	94.80%

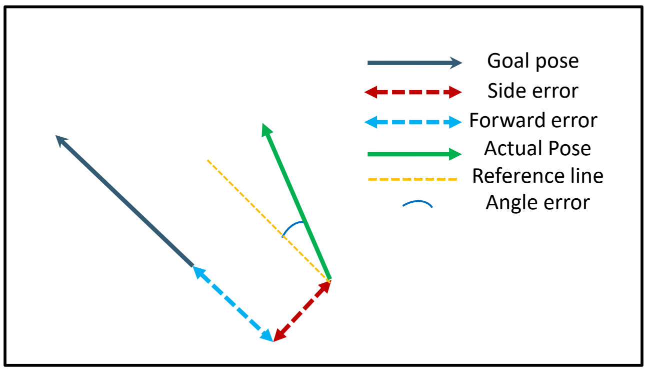

Figure 10. Error decomposition diagram, including side, forward, and heading errors, used to evaluate the planner.

4.2. Randomized Target Experimental Evaluation

To further test the robustness of our system for arbitrary target poses and longer distances, we used two test scenes, as shown in Figure 11, each with two blocks marked by dashed lines area. A total of 20 target groups were randomly generated between two areas. The vehicle was tested in two scenarios, from one area to another, reciprocating between multiple poses, and a total of 40 paths were tested.

The experimental results obtained are given as follows. The forward, lateral, and heading errors are shown in Table 2 while the motion-planning time, path-constraint-extraction time, and path-optimization time are shown in Table 3. Table 2 compares the improvement ratio of various indicators before and after path smoothing, with the average improvement value being 92.42%. Figure 12 is a visual description of Table 3. The experimental results indicated that the proposed path-smoothing method had high robustness and great improvements in parking errors, even when the maximum position accuracy error was less than 3cm. In addition, the total time demand of path planning was shortened, which fulfilled the real-time and precision requirements of online path generation for forklifts.

Table 2. Random goals errors.

	Forward Error			Side Error			Heading Error		
Method	**Mean (m)**	**Std (m)**	**Max (m)**	**Mean (m)**	**Std (m)**	**Max (m)**	**Mean (rad)**	**Std (rad)**	**Max (rad)**
20 random goals short	0.0021	0.0013	0.0052	0.0093	0.0057	0.0265	0.0019	0.0018	0.0072
20 random goals long	0.0019	0.0023	0.0064	0.009	0.0064	0.0231	0.0022	0.0017	0.0091

Table 3. Computational time for motion planning and path smoothing (short/long).

Method	Mean (s)	Std (s)	Max (s)
Motion planning	0.395/2.002	0.413/1.734	1.502/5.433
Constraint extraction	0.134/0.356	0.032/0.054	0.231/0.487
Path smoothing	0.492/5.752	0.273/3.312	1.523/12.651

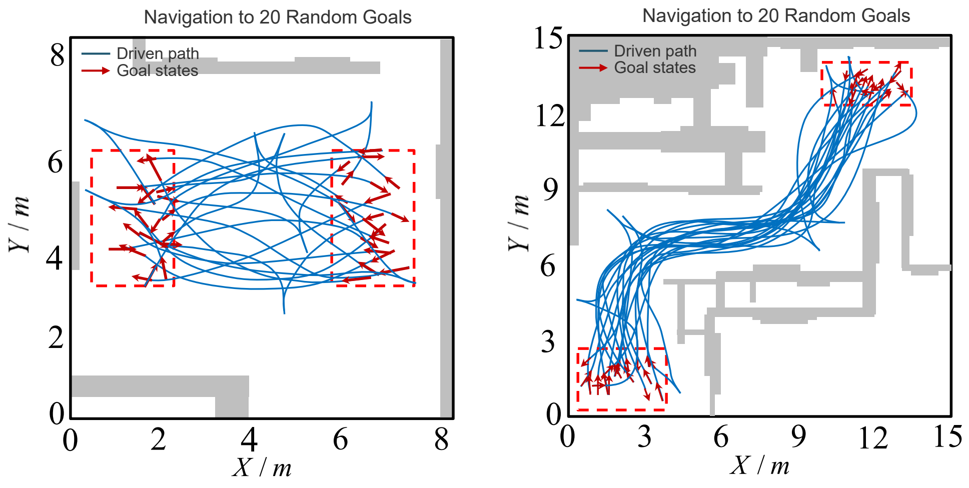

Figure 11. Random target selection. **Left**/**Right**: Path driven when navigating to 40 randomly chosen targets (short/long).

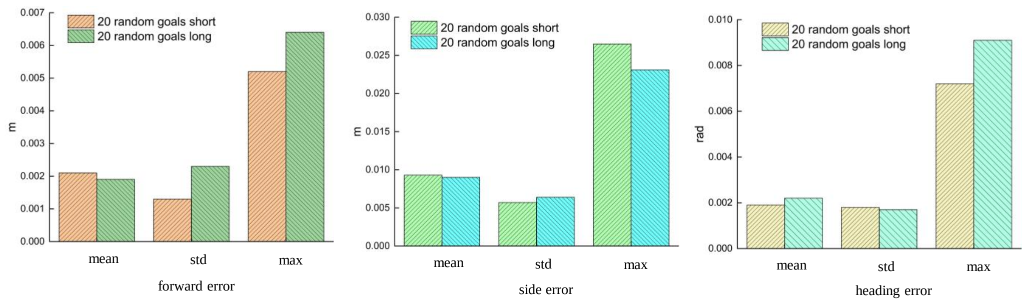

Figure 12. Error visualization description.

4.3. Barrier Extension Assessment

In order to test the operation of our method in more challenging scenarios, we conducted experimental evaluations in narrower and longer scenarios with additional obstacles, as shown in Figure 13. In the figure, the shaded part is the additional obstacle, as expected, due to the more complex space of these two test scenes; the deviation of the smoothed path from the original path is smaller, and the number of space constraints is higher. The experimental results showed that the planning method proposed in this paper was also effective in more challenging scenarios.

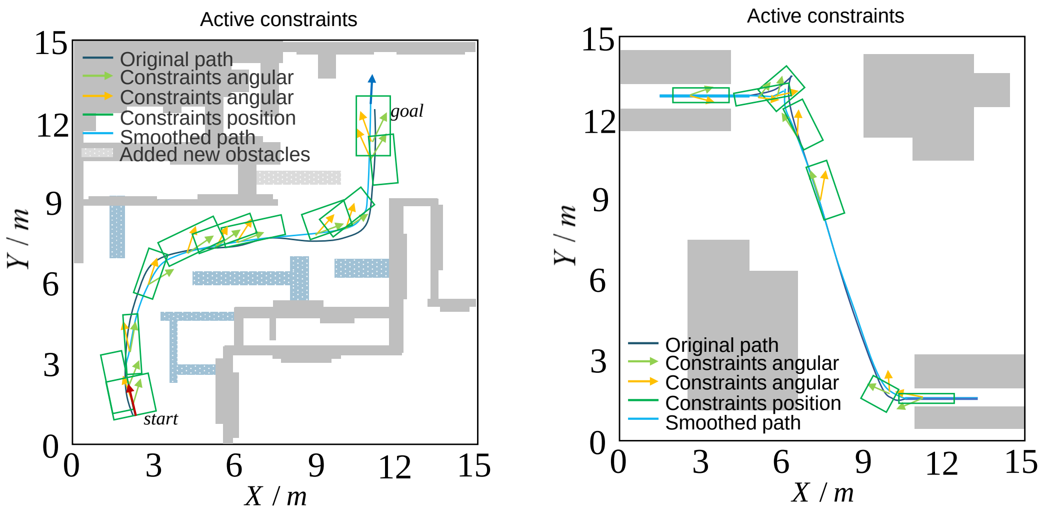

Figure 13. Obstacle expansion. **Left**: The same start and target poses were added to the map Figure 8 with additional obstacles. **Right**: Narrow and long test environment.

4.4. Real Pallet Fork Experiment

In order to verify the reliability and robustness of our proposed algorithm for performing real tasks, we carried out a flexible fork-and-pick experiment on a pallet in a 600-square-meter experimental test site. The selected tray specifications were as follows: grid plastic tray, 1.2 m long, 1.2 m wide, and 0.155 m high. We used a high-precision pallet recognition-and-positioning system: The recognition frequency was 1 HZ, the recognition position accuracy was 0.008 m, and the attitude accuracy was ±1.5 degrees.

The starting point Ps was defined in the experimental site, the photographing point Pa was fixed, the real pallet position point Pr was recognized, and the fork adjustment point Pd was automatically generated, as shown in Figure 14. In each fork-and-pick experiment, the pallet placement position was set within a certain position and attitude deviation relative to the photo point (position deviation $\leq$ 0.2 m, attitude deviation $\leq$ 0.5 rad), and the forklift navigated from any position Ps on the site to the photo point Pa. Actions, such as image captures, the flexible adjustment of poses, and the tray insertion, were completed successively. After 10 random pallet fork experiments, including the empty space near the pallet placement position (as shown in Figure 15) and the more difficult challenge of a tray placed near the wall (as shown in Figure 16), the task success rate was 100%.

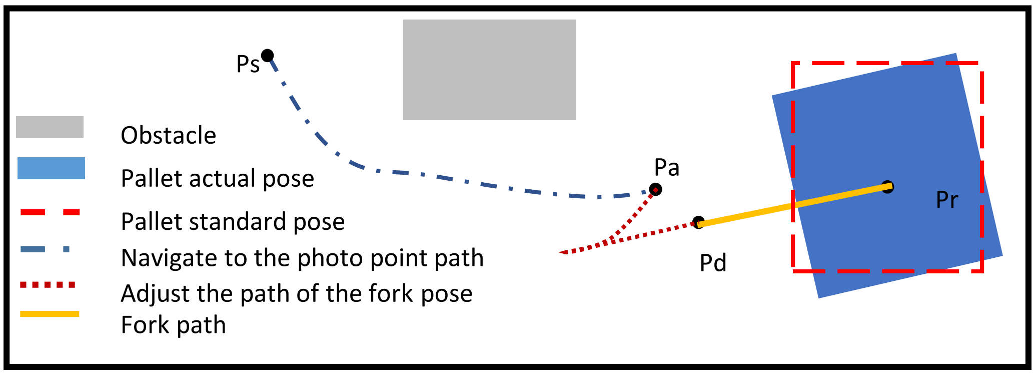
Ps
Pa
Pd
Pr
Obstacle
Pallet actual pose
Pallet standard pose
Navigate to the photo point path
Adjust the path of the fork pose
Fork path

Figure 14. Schematic diagram of the fork experiment process.

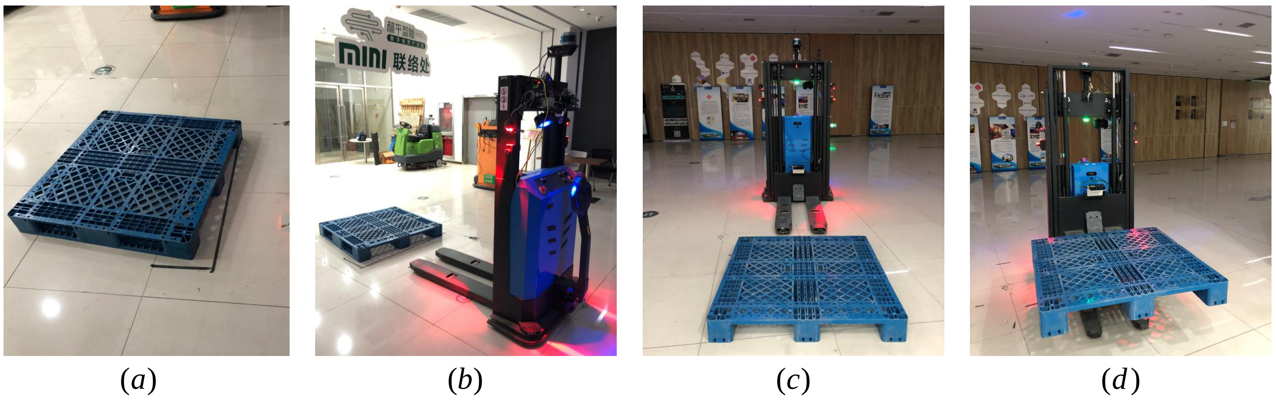

Figure 15. Empty space near the pallet placement position. (**a**) Pallet. (**b**) Photo spot. (**c**) Correction point. (**d**) Pallet fork.

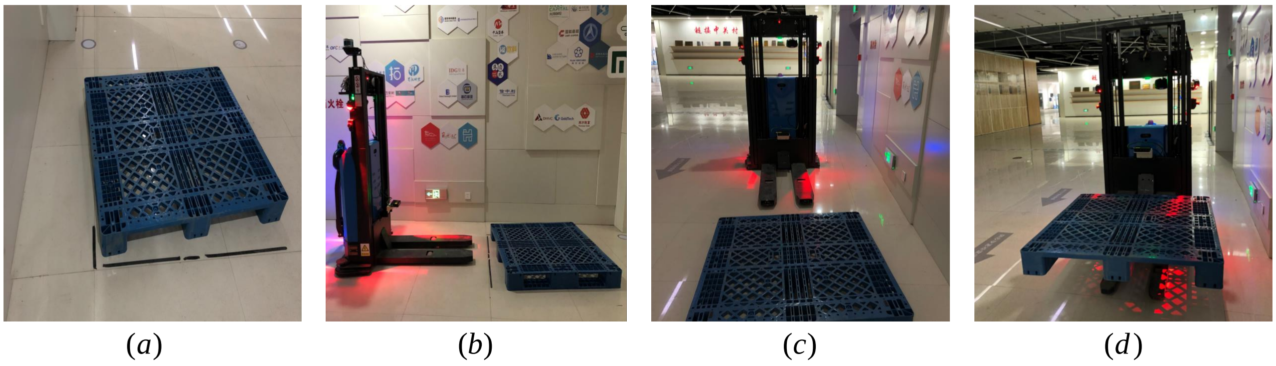

Figure 16. Difficult scene present near the pallet placement position. (**a**) Pallet. (**b**) Photo point. (**c**) Correction point. (**d**) Pallet fork.

5. Conclusions and Future Works

Current forklift trajectory planning systems that use offline generation have been unable to provide flexible, smooth route planning. We therefore proposed and designed a new flexible online trajectory planner for industrial forklifts in this study. There are two main advantages offered by our proposed method:

(1) It operates online. From the front-end original path search to the back-end path optimization, time constraints were considered. The online search employed the ARA*

method to ensure search-time constraints. The time-consuming space safety envelope was divided into two steps involving the offline calculation of the envelope set and the online generation of the path safety envelope. The safety envelope constraint with position and attitude angles was directly added to the back-end path optimization, which effectively eliminated the second verification step of trajectory after the path had been smoothed. Additional curvature constraints were incorporated into the path optimization process, thus no secondary curvature evaluations were required on the final path. These steps ensured the overall time efficiency of the online trajectory generation.

(2) It demonstrates high precision. The trajectory planner designed in this paper was conducted online, which improved the accuracy, smoothness, and security of planning. The front-end path search used a sampled-state lattice-based path-search method. The original path search satisfied the kinematic constraints of the vehicle; however, problems such as discretization errors and curvature discontinuities occurred on the path itself or between the parking point and the target state. This did not fulfill the end-pose accuracy requirements for industrial applications. The back-end optimization method smoothed the path. The state constraints of the starting and the target points were added to the path optimization, which ensure the pose accuracy of the optimized path at the starting and the target points. Before the fork point, by imposing corner constraints to ensure smoothness of the end path, the parking attitude accuracy of the target point was effectively improved. These steps ensured high precision of the trajectory generated by the navigation system.

In our future research, we plan to improve the execution of the navigation task as well as reduce execution time. At present, the pursuit of refined controls in digital factories for the transportation of goods is directly connected with production and assembly. In order to improve overall production and handling efficiency, the factory's task scheduling system not only sends handling instructions to the forklift, but also issues the time required to complete the navigation task. Therefore, the execution of navigation tasks with time constraints is critical to achieving improved production efficiency, assembly, and transportation in future intelligent factories.

Author Contributions: Conceptualization, Y.S. (Yizhen Sun); Methodology, Y.S. (Yizhen Sun) and J.Y.; Software, Y.S. (Yizhen Sun); project administration, J.Y.; data curation, Z.Z.; resources, Z.Z.; visualization, Y.S. (Yu Shu); writing—original draft, Y.S. (Yu Shu) and Z.Z.; writing—review and editing, Y.S. (Yizhen Sun); funding acquisition, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Central Government Guides Local Science and Technology Development Funds (Grant No. 2021JH6/10500216), the Natural Science Foundation of Shenyang City (Grant No. 22-315-6-02), and the Liaoning Province Modern Distance Education Institution (2022XH-12).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: All data generated or analyzed during this study are included in this published article.

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

Appendix A

Table A1. Summary of variable comparison table.

Variable Interpretation	Variable
forklift front wheel coordinates	(x_f, y_f)
rear wheel center coordinates	(x_r, y_r)
AGV forklift instantaneous rotation center	o
AGV front and rear wheel wheelbase	L
steering wheel linear speed	v
steering wheel angular speed	w

Table A1. *Cont.*

vehicle instantaneous linear speed	v_r
vehicle instantaneous angular speed	w_r
front steering wheel angle	δ
vehicle heading angle	φ
control space	c
state space	s
state transition	$\dot{s}$
forklift reference position point	(x, y)
state space set	ξ
control space set	η
internal position constraint	C_{inner}
angle constraint	C_{angle}
safety envelope constraint set	PC
weight of each constraint set	ω
outer constraints	C_{outer}
left boundary of the angle constraint	B_{left}
right boundary of the angle constraint	B_{right}
total number of path points	n
coordinates of the i-th waypoint	P
four vertices of the rectangle	P_2, P_2, P_3, P_4
local map	$L - MAP$
grid	$Grid$
grid covered by the local map	Occ
array save grids covered by the local map	O
starting point of the forklift	s_0
target point of the forklift	s_N
discretized starting state	$\bar{s}_0$
discretized target state	$\bar{s}_N$
time between consecutive state points	ΔT
time through the entire path	$N\Delta T$
weighting factor of adjusting the minimum	α

References

1. Milanowicz, M.; Budziszewski, P.; Kdzior, K. Numerical analysis of passive safety systems in forklift trucks. *Saf. Sci.* **2018**, *101*, 98–107. [CrossRef]
2. Fazlollahtabar, H.; Saidi-Mehrabad, M. Methodologies to Optimize Automated Guided Vehicle Scheduling and Routing Problems: A Review Study. *J. Intell. Robot. Syst.* **2015**, *77*, 525–545. [CrossRef]
3. Cservenák, K. *Further Development of an AGV Control System*; Springer: Cham, Switzerland, 2018.
4. Balatti, P.; Fusaro, F.; Villa, N.; Lamon, E.; Ajoudani, A. A collaborative robotic approach to autonomous pallet jack transportation and positioning. *IEEE Access* **2020**, *8*, 142191–142204. [CrossRef]
5. Liu, C.; Mao, Q.; Chu, X.; Xie, S. An improved A-star algorithm considering water current, traffic separation and berthing for vessel path planning. *Appl. Sci.* **2019**, *9*, 1057. [CrossRef]
6. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A survey of path planning algorithms for mobile robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
7. Zheng, W.; Shi, J.; Wang, A.; Fu, P.; Jiang, H. A Routing-Based Repair Method for Digital Microfluidic Biochips Based on an Improved Dijkstra and Improved Particle Swarm Optimization Algorithm. *Micromachines* **2020**, *11*, 1052. [CrossRef]
8. Bergman, K.; Ljungqvist, O.; Axehill, D. Improved optimization of motion primitives for motion planning in state lattices. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 2307–2314.
9. Sánchez-Ibáñez, J.R.; Pérez-del Pulgar, C.J.; García-Cerezo, A. Path planning for autonomous mobile robots: A review. *Sensors* **2021**, *21*, 7898. [CrossRef]
10. Kang, J.G.; Lim, D.W.; Choi, Y.S.; Jang, W.J.; Jung, J.W. Improved RRT-connect algorithm based on triangular inequality for robot path planning. *Sensors* **2021**, *21*, 333. [CrossRef]
11. Wang, H.; Li, G.; Hou, J.; Chen, L.; Hu, N. A path planning method for underground intelligent vehicles based on an improved RRT* algorithm. *Electronics* **2022**, *11*, 294. [CrossRef]
12. Dang, C.V.; Ahn, H.; Lee, D.S.; Lee, S.C. Improved Analytic Expansions in Hybrid A-Star Path Planning for Non-Holonomic Robots. *Appl. Sci.* **2022**, *12*, 5999. [CrossRef]
13. Wu, B.; Chi, X.; Zhao, C.; Zhang, W.; Lu, Y.; Jiang, D. Dynamic Path Planning for Forklift AGV Based on Smoothing A* and Improved DWA Hybrid Algorithm. *Sensors* **2022**, *22*, 7079. [CrossRef] [PubMed]

14. Pivtoraiko, M.; Kelly, A. Kinodynamic motion planning with state lattice motion primitives. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 2172–2179.
15. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors* **2018**, *18*, 3170. [CrossRef]
16. Noreen, I.; Khan, A.; Habib, Z. Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 97–107. [CrossRef]
17. Arslan, O.; Berntorp, K.; Tsiotras, P. Sampling-based Algorithms for Optimal Motion Planning Using Closed-loop Prediction. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
18. Živojević, D.; Velagić, J. Path planning for mobile robot using Dubins-curve based RRT algorithm with differential constraints. In Proceedings of the 2019 International Symposium ELMAR, Zadar, Croatia, 23–25 September 2019; pp. 139–142.
19. Gao, F.; Wu, W.; Gao, W.; Shen, S. Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *J. Field Robot.* **2019**, *36*, 710–733. [CrossRef]
20. Li, X.; Sun, Z.; Cao, D.; He, Z.; Zhu, Q. Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications. *IEEE/ASME Trans. Mechatronics* **2015**, *21*, 740–753. [CrossRef]
21. Zhu, Z.; Schmerling, E.; Pavone, M. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; pp. 835–842.
22. Choi, J.W.; Huhtala, K. Constrained path optimization with Bézier curve primitives. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 246–251.
23. Sprunk, C.; Lau, B.; Pfaff, P.; Burgard, W. Online generation of kinodynamic trajectories for non-circular omnidirectional robots. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9–13 May 2011.
24. Sprunk, C.; Lau, B.; Pfaff, P.; Burgard, W. An accurate and efficient navigation system for omnidirectional robots in industrial environments. *Auton. Robot.* **2017**, *41*, 473–493. [CrossRef]
25. Gao, F.; Wu, W.; Lin, Y.; Shen, S. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 344–351.
26. Sands, T. Flattening the curve of flexible space robotics. *Appl. Sci.* **2022**, *12*, 2992. [CrossRef]
27. Pivtoraiko, M.; Kelly, A. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 3231–3237.
28. LaValle, S.M. *Planning Algorithms*; Cambridge University: Cambridge, UK, 2006.
29. Jing, C.; Liu, T.; Shen, S. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. In Proceedings of the IEEE International Conference on Robotics & Automation, Stockholm, Sweden, 16–21 May 2016.
30. Sandberg, A.; Sands, T. Autonomous trajectory generation algorithms for spacecraft slew maneuvers. *Aerospace* **2022**, *9*, 135. [CrossRef]
31. Raigoza, K.; Sands, T. Autonomous trajectory generation comparison for de-orbiting with multiple collision avoidance. *Sensors* **2022**, *22*, 7066. [CrossRef] [PubMed]
32. Stoican, F.; Prodan, I.; Popescu, D.; Ichim, L. Constrained trajectory generation for uav systems using a b-spline parametrization. In Proceedings of the 2017 25th Mediterranean Conference on Control and Automation (MED), Valletta, Malta, 3–6 July 2017; pp. 613–618.
33. Xu, Y.; Tang, W.; Chen, B.; Qiu, L.; Yang, R. A model predictive control with preview-follower theory algorithm for trajectory tracking control in autonomous vehicles. *Symmetry* **2021**, *13*, 381. [CrossRef]
34. Xie, F. *Model Predictive Control of Nonholonomic Mobile Robots*; Oklahoma State University: Stillwater, OK, USA, 2007; pp. 3494–3499.
35. Liu, K.; Zhang, Q.; Han, D.; Ma, A.; Xia, Y. Distributed model predictive control for nonholonomic multivehicle formation tracking. *Int. J. Robust Nonlinear Control.* **2021**, *31*, 8961–8973. [CrossRef]
36. Nascimento, T.P.; Dórea, C.; Gonalves, L.M.G. Nonlinear model predictive control for trajectory tracking of nonholonomic mobile robots: A modified approach. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 172988141876046. [CrossRef]
37. Krug, R.; Stoyanov, T.; Tincani, V.; Andreasson, H.; Mosberger, R.; Fantoni, G.; Lilienthal, A.J. The next step in robot commissioning: Autonomous picking and palletizing. *IEEE Robot. Autom. Lett.* **2016**, *1*, 546–553. [CrossRef]