Fast Safe Rectangular Corridor-based Online AGV Trajectory Optimization with Obstacle Avoidance

Shaoqiang Liang[1], Songyuan Fa[1] and Yiqun Li[1]

Abstract— Automated Guided Vehicles (AGVs) are essential in various industries for their efficiency and adaptability. However, planning trajectories for AGVs in obstacle-dense, unstructured environments presents significant challenges due to the nonholonomic kinematics, abundant obstacles, and the scenario's nonconvex and constrained nature. To address this, we propose an efficient trajectory planning framework for AGVs by formulating the problem as an optimal control problem. Our framework utilizes the fast safe rectangular corridor (FSRC) algorithm to construct rectangular convex corridors, representing avoidance constraints as box constraints. This eliminates redundant obstacle influences and accelerates the solution speed. Additionally, we employ the Modified Visibility Graph algorithm to speed up path planning and a boundary discretization strategy to expedite FSRC construction. Experimental results demonstrate the effectiveness and superiority of our framework, particularly in computational efficiency. Compared to advanced frameworks, our framework achieves computational efficiency gains of 1 to 2 orders of magnitude. Notably, FSRC significantly outperforms other safe convex corridor-based methods regarding computational efficiency.

I. INTRODUCTION

Recently, the deployment of Automated Guided Vehicles (AGVs) has witnessed a significant upsurge across diverse industrial and logistical contexts [1]. Their growing popularity can be attributed to their efficiency and adaptability [2], finding applications in sectors such as mining [3], surveillance [4], forklift operations [5], traffic management [6], electronic manufacturing [7], delivery [8] and IoT scenarios [9]. AGVs promise to automate warehouse operations, curb operational costs, and enhance overall production processes [10], [11]. However, effectively employing AGVs in obstacle-rich environments [12], [13] remains a considerable challenge.

Trajectory planning plays a crucial role in the functioning of AGV systems. As AGVs traverse intricate and ever-evolving environments, their primary task is to navigate while avoiding collisions with stationary and moving obstacles. The challenges of real-time AGV operations underscore the critical role played by efficient trajectory planning frameworks. In obstacle-rich, unstructured environments, trajectory planning for AGVs remains daunting, primarily due to their nonholonomic kinematics, the abundance of obstacles, and the nonconvex, constrained nature of the scenario. This paper

[1]Shaoqiang Liang, Songyuan Fa and Yiqun Li are affiliated with the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China. `sqliang@hust.edu.cn`, `liyiqun@hust.edu.cn`

aims to develop efficient and optimal trajectory planning solutions for AGVs operating in such environments.

A. Related Works

Optimization-based methods describe trajectory planning as an optimal control problem (OCP) [14]–[17], solving an optimization problem to obtain the optimal trajectory. This approach excels at finding trajectories that meet multiple constraints simultaneously. Thus, this paper adopts this method to address AGV trajectory planning, aiming to obtain collision-free and kinematically feasible trajectories. However, solving optimization problems in this manner is demanding and time-consuming [16], and it may become intractable in cases of excessively complex constraints.
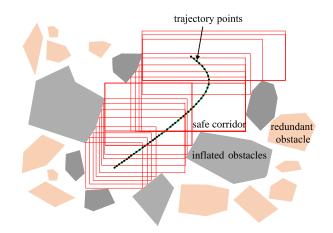

Fig. 1: Illustration of safe convex corridor-based methods: the red rectangles represent the constructed convex corridors, with each trajectory point corresponding to a rectangle. The light orange areas indicate redundant obstacles.

Bergman et al. [18] proposed a systematic approach using numerical optimal control techniques to compute local solutions for motion planning. Their method combines Sequential Quadratic Programming with homotopy methods to transform the problem into its original form, enabling the computation of locally optimal solutions for motion planning. Chai et al. [19] introduced a two-stage trajectory optimization framework for generating optimal parking trajectories for vehicles. Their approach employs a multi-level optimization strategy to enhance convergence and computation efficiency. However, their methods struggle to maintain real-time performance in obstacle-dense scenarios.

Guo et al. [20] proposed a reduced initialization strategy for optimization-based trajectory planning.

Unstructured environments are characterized by irregularly shaped, non-convex obstacles distributed irregularly and in large numbers. As a result, trajectory optimization problems in these environments typically involve numerous collision avoidance constraints, which can decrease solution efficiency. In contrast, safe convex corridor-based methods are better suited for addressing such complex scenarios [21]–[25]. These methods transform general nonlinear, non-convex collision avoidance constraints into convex constraints by confining the robot within a convex polygon or polyhedron, thereby reducing redundant collision avoidance constraints and easing the optimization problem's difficulty (see Fig. 1).

Deits et al. [26] introduced the IRIS (Iterative Regional Inflation by Semidefinite Programming) algorithm, which efficiently computes large polytopes and ellipsoidal free spaces using a series of convex optimizations. Chen et al. [27] proposed the safe passage algorithm to generate collision-free trajectories in cluttered environments. This approach uses rectangles to construct convex corridors instead of convex polygons as in IRIS. Liu et al. [28] presented the Convex Feasible Set Algorithm (CFS), transforming the original problem into a series of convex subproblems. This is achieved by obtaining convex feasible sets within non-convex domains and iteratively solving these subproblems with convex constraints until convergence. Additionally, Li et al. [21] introduced Safe Travel Corridors (STC) for the trajectory planning of autonomous vehicles. STC simplifies collision avoidance constraints in automated parking motion planning, improving efficiency in complex environments. The construction time of safe convex corridors significantly impacts the overall efficiency of trajectory planning frameworks, thereby affecting real-time performance. However, some algorithms exhibit prolonged execution times, consequently diminishing the planning efficiency.

Additionally, the efficiency of optimization-based methods is influenced by the quality of the initial solution [29], highlighting the importance of choosing the appropriate path planning algorithm to generate the initial solution. However, algorithms like A* [30] and hybrid A* [31] are constrained by map size and precision, while RRT [32] and RRT* [33] exhibit instability and do not guarantee the shortest path.

B. Motivations, Contributions, and Organization

This study aims to find optimal collision-free trajectories quickly for AGVs in cluttered and unstructured environments. To achieve this, the trajectory planning problem is formulated as an optimal control problem and then solved numerically to obtain trajectories that satisfy various constraints. However, in unstructured environments, solving this problem efficiently is challenging due to the abundance of obstacles. In many studies, including [34], [35], excluding redundant collision avoidance constraints remains difficult. These constraints, being nonlinear and nonconvex, often result in inefficient solutions. This paper proposes the FSRC algorithm to address these challenges, which constructs convex corridors using rectangles to ensure AGV motion safety within specific rectangular areas. Additionally, we optimize the entire trajectory planning framework to enhance efficiency and enable real-time motion planning. The specific contributions of this paper are as follows:

1) The FSRC algorithm is introduced, significantly accelerating the process of constructing convex regions compared to STC [21] and SFC [21]. The corridors established by FSRC serve as representations for obstacle avoidance constraints within OCP, effectively eliminating redundant obstacles.
2) The Modified Visibility Graph (MVG) is proposed, demonstrating remarkable path-searching speeds.
3) A boundary discretization strategy is introduced, enhancing the construction speed of FSRC.

The structure of this paper is as follows: Section 2 defines the discrete trajectory planning problem. Section 3 elaborates on the specific implementation algorithms of the trajectory planning framework, covering obstacle inflation and discretization, the MVG, and the FSRC. Section 4 presents experimental results, and Section 5 concludes the paper.

II. PROBLEM FORMULATION

Based on the objective of performing trajectory planning and obstacle avoidance for AGVs, this section introduces an optimal control problem to describe the trajectory planning problem.

A. Formulation of Trajectory Planning Problem

The workspace for AGV is defined as $W \subset \mathbb{R}^2$, with the area occupied by obstacles denoted by $W_{\text{obs}} \subset W$. Therefore, the AGV must operate in the free space $W_{\text{free}} \subset W \backslash W_{\text{obs}}$. The trajectory planning problem for the AGV aims to find the optimal trajectory connecting the start and end poses of the AGV, satisfying all constraints. Here, the AGV's state profile is represented by $\xi(t) = [x(t), y(t), v(t), \theta(t)]^T$ and the control profile by $u(t) = [a(t), \omega(t)]^T$. The trajectory planning of the AGV is abstracted as

$$\text{Minimize} \quad J = T_f - T_0$$
$$\text{Subject to}$$
$$\dot{\xi}(t) = f(\xi(t), u(t))$$
$$\xi_{\min} \leqslant \xi(t) \leqslant \xi_{\max}$$
$$U_{\min} \leqslant u(t) \leqslant U_{\max} \quad (1)$$
$$\xi(T_0) = \xi_0, u(T_0) = u_0$$
$$\xi(T_f) = \xi_f, u(T_f) = u_f$$
$$g(\xi(t)) \subset W_{\text{free}}, \forall t \in [T_0, T_f],$$

where T_0 and T_f denote the starting and ending times, $T_f - T_0$ is the cost function, $\dot{\xi}(t) = f(\xi(t), u(t))$ describes the AGV kinematics, $[\xi_{\min}, \xi_{\max}]$ and $[U_{\min}, U_{\max}]$ denote the allowable intervals for ξ and u respectively, and $g(\xi(t)) \subset W_{\text{free}}$ denotes collision-avoidance constraints.

B. Discrete Formulation of Trajectory Planning Problem

Discretizing the interval $[T_0, T_f]$ into $N - 1$ segments using the forward Euler method [36] yields the discrete

formulation of the trajectory planning problem, as elaborated in the following equations:
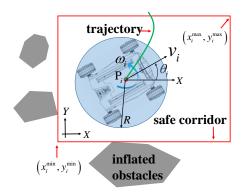

Fig. 2: The kinematic model of the AGV.

Minimize $\quad J = T_f - T_0 \quad$ (2.a)

Subject to

$$x_{i+1} - x_i = v_i \cdot \cos\theta_i \cdot \Delta_t, \ i = 1, 2, ..., N-1 \quad (2.b)$$
$$y_{i+1} - y_i = v_i \cdot \sin\theta_i \cdot \Delta_t, \ i = 1, 2, ..., N-1 \quad (2.c)$$
$$v_{i+1} - v_i = a_i \cdot \Delta_t, \ i = 1, 2, ..., N-1 \quad (2.d)$$
$$\theta_{i+1} - \theta_i = \omega_i \cdot \Delta_t, \ i = 1, 2, ..., N-1 \quad (2.e)$$
$$x_1 = x_{T_0}, \ y_1 = y_{T_0}, \ \theta_1 = \theta_{T_0} \quad (2.f)$$
$$v_1 = v_{T_0}, \ a_1 = a_{T_0}, \ \omega_1 = \omega_{T_0} \quad (2.g)$$
$$x_N = x_{T_f}, \ y_N = y_{T_f} \quad (2.h)$$
$$v_N = v_{T_f}, \ a_N = a_{T_f}, \ \omega_N = \omega_{T_f} \quad (2.i)$$
$$x_i^{\min} \le x_i \le x_i^{\max}, \ y_i^{\min} \le y_i \le y_i^{\max}, \ i = 1, ..., N \quad (2.j)$$
$$v^{\min} \le v_i \le v^{\max}, \ a^{\min} \le a_i \le a^{\max}, \ i = 1, ..., N \quad (2.k)$$
$$\omega^{\min} \le \omega_i \le \omega^{\max}, \ i = 1, 2, ..., N, \quad (2.l)$$
$$(2)$$

where $\Delta_t = \frac{T_f - T_0}{N-1}$ represents the time interval between two time points, equations (2.b∼2.e) capture kinematic constraints for omnidirectional movement. $P_i(x_i, y_i)$ signifies geometric center (see Fig. 2), while v_i, a_i, w_i, and θ_i respectively represent velocity, acceleration, angular velocity, and yaw angle at P_i. (2.f-2.i) define the boundary constraints for the AGV's starting and goal poses. (2.j) represent the obstacle avoidance constraint implemented by FSRC. (2.k) and (2.l) limit the AGV's velocity, acceleration, and angular velocity. (2.b) and (2.c) being nonlinear constraints, they are converted into soft constraints, resulting in a modified objective function:

$$J = (T_f - T_0) + \varpi_1 \sum_{i=1}^{N-1} \left(x_{i+1} - x_i - v_i \cdot \cos\theta_i \cdot \Delta_t \right)^2$$
$$+ \varpi_2 \sum_{i=1}^{N-1} \left(y_{i+1} - y_i - v_i \cdot \sin\theta_i \cdot \Delta_t \right)^2, \quad (3)$$

where ϖ_1 and ϖ_2 represent penalty function weights.

III. ALGORITHM

This section introduces obstacle handling, MVG for path planning, and the specific implementation of FSRC.
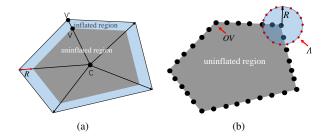

Fig. 3: Inflation of Obstacles and discretization

A. Discretization Strategy for Obstacle Boundariesn

This section introduces an algorithm for discretizing obstacle boundaries, crucial for subsequent path planning and FSRC construction. The process involves gathering three sets: the expanded polygonal obstacle edges Ξ and inflated obstacle vertex Ψ, utilized for obstacle avoidance of MVG, and the inflation obstacle node Λ, used to establish FSRC. These procedures are depicted in Algorithm 1. Here, R represents the coverage circle radius of the AGV (see Fig. 2), N_{obs} denotes the number of obstacles, C_i represents the geometric center of obstacle i, and (ox_{ij}, oy_{ij}) denote the vertices of the obstacle. Fig. 3 illustrates the process of obstacle inflation and discretization, Fig. 3a demonstrates the acquisition of Ξ, and Fig. 3b illustrates the acquisition of Λ. AGVs only need to avoid the boundaries of obstacles to ensure safety. Therefore, this paper focuses on discretizing only the boundaries of obstacles. Additionally, fewer inflation obstacle nodes lead to faster construction of FSRC.

Algorithm 1 Obstacle handling

Require: obstacles O, radius R, discretization precision ℓ
1: $\Xi \leftarrow \emptyset, \Lambda \leftarrow \emptyset, OV \leftarrow \emptyset, \Psi \leftarrow \emptyset,$
2: **for** $i \leftarrow 1$ to N_{obs} **do**
3: $\quad C_i \leftarrow \left(\sum_{j=1}^{N_{Vi}} ox_{ij}, \sum_{j=1}^{N_{Vi}} oy_{ij} \right) \Big/ N_{Vi}$
4: $\quad$ **for** $j \leftarrow 1$ to N_{Vi} **do**
5: $\quad\quad V_{ij}' \leftarrow$ expand (R, V_{ij}) in the direction $\overrightarrow{C_i V_{ij}}$
6: $\quad\quad$ add $V_{ij}' \leftarrow$ to Ψ
7: $\quad$ add $[V_{i,0}', V_{i,N_{Vi}}']$ to Ξ
8: $\quad$ **for** $j \leftarrow 1$ to $N_{Vi} - 1$ **do**
9: $\quad\quad$ add $[V_{i,j}', V_{i,j+1}']$ to Ξ
10: $\quad OV \leftarrow$ discretizeBoundary (O_i, ℓ)
11: $\quad \Lambda_i \leftarrow$ inflateBoundary (OV, R, ℓ)
12: $\quad$ add Λ_i to Λ
13: **return** Λ, Ξ and Ψ

B. Modified Visibility Graph

Building upon [37], the Modified Visibility Graph (MVG) algorithm for path planning is introduced, as depicted in Algorithm 2. This approach encompasses three primary steps. Firstly, a bidirectional weighted graph G is created using the start point s, the goal point g, the expanded polygonal

obstacle edges Ξ, and the inflated obstacle vertex set Ψ. A compact adjacency list $\mathfrak{L}$ is employed to efficiently represent this graph G. The weights between nodes are distance-based, and infinite values indicate collisions. Collision-free paths are delineated by line segments that do not intersect any line segment within Ξ. Secondly, the Dijkstra search [38] focuses on finding the shortest path between s and g rather than examining all possible node pairs. The search concludes once the shortest path between s and g is identified. Thirdly, the outcomes of the second phase are uniformly discretized into N path points. The example can be referenced in Fig. 4.
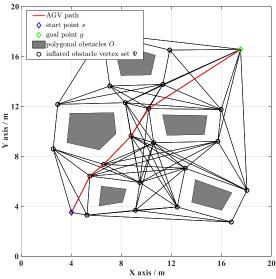

Fig. 4: Results of the Modified Visibility Graph. (Enhance visibility for finer details within the figure by zooming in.)

Algorithm 2 Modified Visibility Graph (MVG)

Require: start point s, goal point g, Ξ, Ψ, number of path points N
1: $\mathfrak{L} \leftarrow$ constructAdjacencyList (s, g, Ξ, Ψ)
2: $(pX, pY) \leftarrow$ Dijkstra $(\mathfrak{L}, s, g)$
3: $path \leftarrow$ discretePath (pX, pY, N)
4: **return** $path$

C. Fast Safe Rectangular Corridor

In this section, we introduce the establishment process of FSRC, which consists of N rectangular boxes $\mathcal{B}_\mathcal{R}(i, :)$, where $i = 1, 2, ..., N$. Each rectangular box $\mathcal{B}_\mathcal{R}(i, :)$ is generated by extending from the path point P_i, and it must not contain any obstacle nodes from Λ. $\mathcal{B}_\mathcal{R}(i, :)$ represents the lengths of the extension from P_i in four directions. The $(x_i^{\max}, x_i^{\max})$ and $(y_i^{\min}, y_i^{\max})$ for Problem (2) can be calculated from the rectangular box $\mathcal{B}_\mathcal{R}(i, :)$ and P_i

$$B_R(i, :) = (L_1, L_2, L_3, L_4)$$
$$x_i^{\min} = x_i - L_2, x_i^{\max} = x_i + L_4 \qquad (4)$$
$$y_i^{\min} = y_i - L_1, y_i^{\max} = y_i + L_3$$

Algorithm 3 Fast Safe Rectangular Corridor (FSRC)

Require: path points $path(X, Y)$, inflation obstacle node set Λ, number of discrete path points N
1: Initialize parameter $\tau, \gamma, L_m, \chi, \mathcal{B}_\mathcal{R}$ and T_m
2: **for** $i \leftarrow 1$ to N **do**
3:     $k \leftarrow 1, x_i \leftarrow X(i), y_i \leftarrow Y(i)$
4:     **if** $i > 1$ and $k \leq T_m$ and (x_i, y_i) in $\mathcal{B}_\mathcal{R}(i - 1, :)$ **then**
5:         $\mathcal{B}_\mathcal{R}(i, :) \leftarrow \mathcal{B}_\mathcal{R}(i - 1, :), k \leftarrow k + 1$
6:         **continue**
7:     $k \leftarrow 1$
8:     $\mathcal{B}_\mathcal{R}(i, :) \leftarrow [L_m, L_m, L_m, L_m]$
9:     $\Lambda_{rm} \leftarrow$ obsCheck$(\mathcal{B}_\mathcal{R}(i, :), \Lambda, x_i, y_i)$
10:     **if** $\Lambda_{rm} = \emptyset$ **then**
11:         **continue**
12:     $\mho \leftarrow [0, 0, 0, 0], box \leftarrow [\tau, \tau, \tau, \tau]$
13:     $\mathcal{B}_\mathcal{R}(i, :) \leftarrow [0, 0, 0, 0]$
14:     **while** sum$(\mho) < 4$ **do**
15:         **for** $j \leftarrow 1$ to 4 **do**
16:             **if** $\mho(j) = 1$ **then**
17:                 **continue**
18:             $tmp \leftarrow \mathcal{B}_\mathcal{R}(i, :), tmp(j) \leftarrow tmp(j) + box_j$
19:             **if** $tmp(j) > L_m$ **then**
20:                 $\mho(j) \leftarrow 1$, **continue**
21:             $\tilde{\Lambda_{rm}} \leftarrow$ obsCheck$(\mathcal{B}_\mathcal{R}(i, :), \Lambda_{rm}, x_i, y_i)$
22:             **if** $\tilde{\Lambda_{rm}} = \emptyset$ **then**
23:                 $\mathcal{B}_\mathcal{R}(i, j) \leftarrow tmp(j), box_j \leftarrow box_j \times \gamma$
24:             **else**
25:                 $box_j \leftarrow \min(box_j/4, \chi)$
26:                 $tmp(j) \leftarrow \mathcal{B}_\mathcal{R}(i, j) + box_j$
27:                 **while** $tmp(j) < L_m$ **do**
28:                     $\tilde{\Lambda_{rm}} \leftarrow$ obsCheck$(\mathcal{B}_\mathcal{R}(i, :), \Lambda_{rm})$
29:                     **if** $\tilde{\Lambda_{rm}} = \emptyset$ **then**
30:                         $\mathcal{B}_\mathcal{R}(i, j) \leftarrow tmp(j)$
31:                         $tmp(j) \leftarrow tmp(j) + box_j$
32:                     **else**
33:                       **break**
34:                 $\mho(j) \leftarrow 1$
35: **return** $\mathcal{B}_\mathcal{R}$

Figure 5 depicts the construction process of the rectangular corridor, utilizing the discrete path point sequence $path$ generated by Algorithm 2 and Λ obtained from Algorithm 1. Figures 5 (b)-(d) illustrate the process of generating the corresponding rectangular box $\mathcal{B}_\mathcal{R}(i, :)$ for $P_i(x_i, y_i)$. Starting from $P_i(x_i, y_i)$, the box is iteratively expanded in the up, left, down, and right directions (denoted as directions 1-4), with each expansion adding an extension rectangle to the box in that direction. Algorithm 3 outlines the process of generating FSRC, which is expedited by the following three strategies.

Strategy 1: The bounding boxes of two adjacent path points P_i and P_{i+1} often exhibit overlapping tendencies owing to their proximity. A check is conducted to avoid redundant box creation: if P_i is within the boundaries of
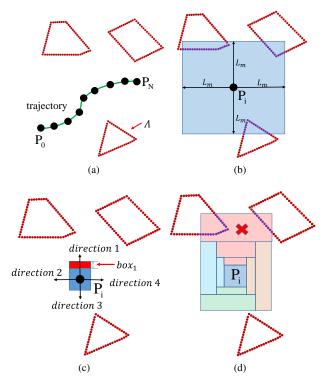
(a)   (b)

(c)   (d)

Fig. 5: The creation process of FSRC: (a) path points and obstacle set Λ, (b) initial bounding box $\mathcal{B}_{\mathcal{R}}(i,:)$, (c) expansion in four directions, (d) expansion encountering obstacles.

the previous bounding box, $\mathcal{B}_{\mathcal{R}}(i-1,:)$, the current process is skipped. This omission step can be repeated up to a maximum of T_m times.

Strategy 2: Initialize a rectangular box denoted as $[L_m, L_m, L_m, L_m]$, where L_m represents the maximum allowable extension length. If no obstacle nodes are within the initial box, subsequent operations are skipped to reduce the construction time. However, if obstacles are detected inside the initial box, expansion starts from point $P_i(x_i, y_i)$, ensuring that the subsequent expansion does not exceed the range of the initial box, i.e., $L_i \leqslant L_m$ for $i = 1, 2, 3, 4$. Moreover, use the obstacle nodes inside the initial box, denoted as $\Lambda_{rm} \subset \Lambda$, for obstacle detection using obsCheck(), instead of using all nodes in Λ. Limiting L_i to expand at most up to the boundary L_m accelerates the execution speed of the obsCheck() function.

Strategy 3: We don't use uniform expansion to quickly reach the obstacle boundary. Instead, we divide expansion into two stages for each direction: rapid growth and linear growth, using a step size of box with an initial value of τ.

- **Rapid Growth**: The expansion length box_j in each direction is doubled by a factor of η in each iteration until it reaches the obstacle boundary or the length of L_m. If the expansion encounters an obstacle or exceeds L_m, the step size box_j is adjusted to the minimum value between $box_j/4$ and χ.
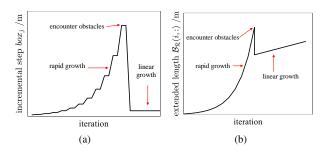- **Linear Growth**: Expansion in the current direction

continues with a constant step size box_j until a collision with an obstacle or reaching L_m occurs.

The relationship between the expansion length box_j and the accumulated expansion length $\mathcal{B}_{\mathcal{R}}(i-1,:)$ with iteration number is illustrated in Fig. 6.

By employing these three strategies, we reduce the FSRC generation time, enhancing the overall efficiency of the trajectory planning framework.

IV. EXPERIMENT RESULTS AND DISCUSSIONS

(a)   (b)

Fig. 6: Rapid growth and linear growth.

TABLE I: Parameters of the experiment.

Parameter	Value
AGV Length	0.612 m
AGV Width	0.582 m
maximum speed $v^{\max}$	3.0 m/s
maximum acceleration $a^{\max}$	1.8 m/s^2
maximum angular velocity $\omega^{\max}$	2.5 rad/s
map size	20×20 m
number of discrete points N for Problem (2)	80
discretization precision of obstacle points ℓ	0.1 m
half-length of initial box L_m	10 m
maximum repetition count T_m	8
growth factor γ	2
minimum growth length χ	0.2 m
grid map precision (x, y, θ) for hybrid A*	0.1 m, 0.1 m, 0.1 rad

This section presents simulation and physical experiments. We demonstrate the computational efficiency of our FSRC algorithm compared to other safe convex corridor-based methods. Additionally, we compare our framework with advanced trajectory planning frameworks in simulation experiments to validate its exceptional performance. Furthermore, we validate the practical viability of our framework through physical platform experiments. Detailed parameter configurations for experiments are provided in Table I.

A. Simulation

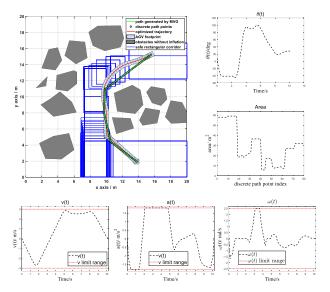In our simulation experiments, we utilized IPOPT [39] and MA57 [40] as the solvers for the Problem (2). These

Fig. 7: Simulation results for trajectory planning in Case1.

experiments were carried out using MATLAB 2020b on a desktop computer with an AMD Ryzen 5 3600X 6 Core CPU running at 3.8 GHz and 16 GB of RAM.

The simulation results of our proposed framework have been presented in five distinct scenarios, as illustrated in Fig.7 and Fig.8, which can be zoomed in to observe details. These scenarios encompass varying obstacle densities: Case1 (low), Case2 (medium), and Case3 (high), along with Case4 and Case5 depicting narrow passages. In all five cases, the blue rectangular boxes generated by FSRC ensure that the AGV avoids contact with obstacles. Specifically, Fig. 7 illustrates that the AGV's speed, acceleration, and angular velocity remained within predefined ranges. Additionally, the area figure demonstrates the area of each rectangular box $\mathcal{B}_{\mathcal{R}i}$ generated by FSRC.

TABLE II: Efficiency comparison of Safe Convex Corridor-based methods (Time: seconds).

Method	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Mean
FSRC	**0.0971**	**0.0959**	**0.0825**	**0.0855**	**0.0903**
STC	0.5766	1.3990	0.5034	0.5014	0.7451
SFC	0.2193	0.2288	0.2910	0.2109	0.2375

To validate the computational efficiency of FSRC, we compared it with two other advanced methods based on safe convex corridors, STC [21] and SFC [21]. FSRC outperformed both, with average efficiency improvements of 725% and 163%, respectively, as shown in Table II. This superiority is attributed to the three strategies implemented to minimize corridor construction time. SFC constructs convex polygons or polytopes, and its process is complex, leading to complexity and lower efficiency. STC uses uniform expansion, with equal lengths for each expansion, which is less efficient than our two-stage approach: rapid and linear growth. Additionally, the FSRC algorithm proposed by us utilizes obstacle boundary points for collision detection in

corridor construction, instead of the entire obstacle area, further enhancing the construction speed.

TABLE III: Efficiency comparison of trajectory planning frameworks (Time: seconds).

case	Method	Mapping	Path	Corridor	Optimization	Total
Case1	Proposed	**0.143**	**0.004**	**0.103**	**0.416**	**0.667**
	Area-based	0.387	11.531	-	21.560	33.477
	STC	0.387	11.531	0.473	1.632	14.021
Case2	Proposed	**0.285**	**0.004**	**0.075**	**0.375**	**0.739**
	Area-based	0.303	0.276	-	48.055	48.634
	STC	0.303	0.276	0.440	0.811	1.830
Case3	Proposed	**0.551**	**0.004**	**0.112**	**0.380**	**1.049**
	Area-based	0.354	28.093	-	5.571	34.018
	STC	0.354	28.093	0.651	0.861	29.605
Case4	Proposed	**0.08**	**0.004**	**0.112**	**0.364**	**0.596**
	Area-based	0.639	0.508	-	3.88	5.027
	STC	0.639	0.508	0.279	0.383	1.809
Case5	Proposed	**0.093**	**0.004**	**0.194**	**0.378**	**0.669**
	Area-based	0.344	1.891	-	4.95	7.185
	STC	0.344	1.891	0.301	0.363	5.537

TABLE IV: The comparison of the optimal path lengths.

Method	Case1	Case2	Case3	Case4	Case5	Mean
Proposed	18.952	21.340	21.185	13.404	19.047	18.786
Area-based	18.724	34.919	19.254	12.247	18.964	20.822
STC	19.456	15.906	20.397	14.545	17.971	17.655

Our proposed framework was comprehensively evaluated by comparing it with two advanced frameworks: the area-based method [35] and the STC-based method [21]. Table III summarizes the computational times for these three frameworks. In this table, "Mapping" represents the map construction time. Our framework constructs an adjacency list (Algorithm 1), while the others make grid maps. "Path" represents the path planning time. We use MVG, while the others use hybrid A*, thus requiring grid map construction. "Corridor" represents the time for safe corridor construction. Since the area-based method does not construct corridors, its time is 0. Significantly, our proposed algorithm outperforms the others in multiple aspects, including path planning, optimization problem-solving, and total computation time, highlighting the superior performance of our framework. This highlights the superior performance of our framework. Compared to these two frameworks, our framework achieves computational efficiency gains of 1 to 2 orders of magnitude.

Finally, Table IV shows the optimal path lengths obtained by the three frameworks in the five cases. The differences in the shortest paths obtained by the three frameworks are insignificant, with STC having the shortest path.

B. PHYSICAL EXPERIMENT

Our physical experiment AGV platform is built upon the AgileX SCOUT MINI robot chassis development platform (refer to Fig. 9). This platform offers exceptional terrain
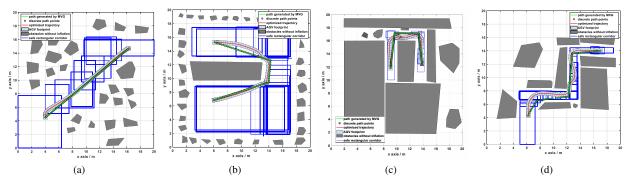
Fig. 8: Simulation results for trajectory planning: (a) Case2, (b) Case3, (c) Case4, (d) Case5.

adaptability and ground clearance, enabling agile movement across various surfaces. The fundamental parameters of the SCOUT MINI can be found in Table I. The AGV platform is equipped with a Velodyne VLP-16 3D LiDAR with a maximum range of 100m, a 9-axis IMU, and an NVIDIA Jetson NX. The NVIDIA Jetson NX features a GPU with 384 cores (Volta architecture @1100MHz + 48 Tensor Cores) and a CPU with NVIDIA Carmel ARMv8.2 (6-core) @1.4GHz (6MB L2 + 4MB L3 cache).

In the beginning, the map of the test environment was constructed using the Velodyne VLP-16 and the mapping method Lio-sam [41]. The test experiment occurred within a 6 m × 8 m map, where obstacles were strategically arranged using cardboard boxes. The decision to create a 3D map was driven by the necessity to account for obstacles of varying heights in the trajectory planning algorithm, ensuring safety. Fig. 10 illustrates both the map of the test environment and the resulting 3D point cloud map. Throughout the physical experiments, we imposed maximum limits: the linear velocity $v^{\max}$ was capped at 0.5 m/s, while the angular velocity $\omega^{\max}$ was restricted to 1.6 rad/s. The experiment's results can be referenced in Fig. 11. For more information, please refer to the video submitted with our presentation.
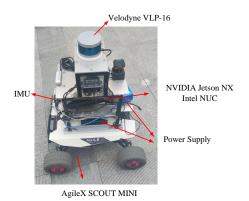

Fig. 9: Physical experiment environment with AgileX SCOUT MINI platform.

Fig. 10: The physical experiment scenario includes five obstacles. The size of each obstacle is $85 \times 60 \times 50$mm. (a) actual scene, (b) constructed point cloud map.

V. CONCLUSIONS

This paper introduces a practical online fast trajectory planning framework for AGVs operating in obstacle-rich environments. The trajectory planning problem is formulated as an optimal control problem. Initially, the Modified Visibility Graph is used for rapid path planning and deriving path points. Subsequently, the Fast Safe Rectangular Corridor algorithm is employed to establish safe corridors quickly. This constrains AGVs' path points within corresponding rectangular boxes for optimization, effectively converting large-scale non-convex redundant obstacle avoidance constraints into linear and easily solvable box constraints. The effectiveness and superiority of our approach are validated through simulation and physical experiments.
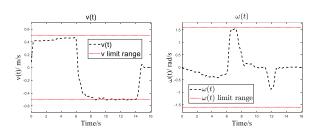

Fig. 11: The corresponding linear velocity, linear acceleration, and angular velocity during SCOUT MINI's execution.

REFERENCES

[1] K. Saleh, M. Hossny, and S. Nahavandi, "Real-time intent prediction of pedestrians for autonomous ground vehicles via spatio-temporal

densenet," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9704–9710.

[2] B. Li, Y. Zhang, Y. Ouyang, Y. Liu, X. Zhong, H. Cen, and Q. Kong, "Real-time trajectory planning for agv in the presence of moving obstacles: A first-search-then-optimization approach," *arXiv preprint arXiv:1902.06201*, 2019.

[3] J. M. Roberts, E. S. Duff, and P. I. Corke, "Reactive navigation and opportunistic localization for autonomous underground mining vehicles," *Information Sciences*, vol. 145, no. 1-2, pp. 127–146, 2002.

[4] X. Zhang, C. Shu, S. Li, C. Wu, and Z. Liu, "Agvs: A new change detection dataset for airport ground video surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 588–20 600, 2022.

[5] L. Li, Y.-H. Liu, M. Fang, Z. Zheng, and H. Tang, "Vision-based intelligent forklift automatic guided vehicle (agv)," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 264–265.

[6] F. Pratissoli, N. Battilani, C. Fantuzzi, and L. Sabattini, "Hierarchical and flexible traffic management of multi-agv systems applied to industrial environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 009–10 015.

[7] W.-Q. Zou, Q.-K. Pan, and M. F. Tasgetiren, "An effective iterated greedy algorithm for solving a multi-compartment agv scheduling problem in a matrix manufacturing workshop," *Applied Soft Computing*, vol. 99, p. 106945, 2021.

[8] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5816–5823, 2021.

[9] M. Elsisi and M.-Q. Tran, "Development of an iot architecture based on a deep neural network against cyber attacks for automated guided vehicles," *Sensors*, vol. 21, no. 24, p. 8467, 2021.

[10] D. Bechtsis, N. Tsolakis, D. Vlachos, and E. Iakovou, "Sustainable supply chain management in the digitalisation era: The impact of automated guided vehicles," *Journal of Cleaner Production*, vol. 142, pp. 3970–3984, 2017.

[11] I. P. Vlachos, R. M. Pascazzi, G. Zobolas, P. Repoussis, and M. Giannakis, "Lean manufacturing systems in the area of industry 4.0: A lean automation plan of agvs/iot integration," *Production planning & control*, vol. 34, no. 4, pp. 345–358, 2023.

[12] B. Farooq, J. Bao, H. Raza, Y. Sun, and Q. Ma, "Flow-shop path planning for multi-automated guided vehicles in intelligent textile spinning cyber-physical production systems dynamic environment," *Journal of manufacturing systems*, vol. 59, pp. 98–116, 2021.

[13] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2182–2189, 2017.

[14] X. Wang, B. Li, X. Su, H. Peng, L. Wang, C. Lu, and C. Wang, "Autonomous dispatch trajectory planning on flight deck: A search-resampling-optimization framework," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105792, 2023.

[15] B. Li, Y. Zhang, Y. Ouyang, Y. Liu, X. Zhong, H. Cen, and Q. Kong, "Fast trajectory planning for agv in the presence of moving obstacles: A combination of 3-dim a search and qcqp," in *2021 33rd Chinese Control and Decision Conference (CCDC)*. IEEE, 2021, pp. 7549–7554.

[16] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3263–3274, 2016.

[17] S. Shi, Y. Xiong, J. Chen, and C. Xiong, "A bilevel optimal motion planning (bomp) model with application to autonomous parking," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 370–382, 2019.

[18] K. Bergman and D. Axehill, "Combining homotopy methods and numerical optimal control to solve motion planning problems," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 347–354.

[19] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Two-stage trajectory optimization for autonomous ground vehicles parking maneuver," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3899–3909, 2018.

[20] Y. Guo, D. Yao, B. Li, H. Gao, and L. Li, "Down-sized initialization for optimization-based unstructured trajectory planning by only opti-mizing critical variables," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 709–720, 2022.

[21] B. Li, T. Acarman, X. Peng, Y. Zhang, X. Bian, and Q. Kong, "Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach," in *2020 European Control Conference (ECC)*. IEEE, 2020, pp. 1993–1998.

[22] Y. Li, S. Liang, J. Gao, Z. Chen, S. Qiao, and Z. Yin, "Trajectory optimization for the nonholonomic space rover in cluttered environments using safe convex corridors," *Aerospace*, vol. 10, no. 8, p. 705, 2023.

[23] J. Lian, W. Ren, D. Yang, L. Li, and F. Yu, "Trajectory planning for autonomous valet parking in narrow environments with enhanced hybrid a* search and nonlinear optimization," *IEEE Transactions on Intelligent Vehicles*, 2023.

[24] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *2015 54th IEEE conference on decision and control (CDC)*. IEEE, 2015, pp. 835–842.

[25] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[26] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 109–124.

[27] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1476–1483.

[28] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[29] B. Li, Z. Yin, Y. Ouyang, Y. Zhang, X. Zhong, and S. Tang, "Online trajectory replanning for sudden environmental changes during automated parking: A parallel stitching method," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 748–757, 2022.

[30] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[31] M. Buehler, K. Iagnemma, and S. Singh, "Junior: the stanford entry in the urban challenge," *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56, pp. 91–123, 2009.

[32] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[33] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: a survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[34] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5954–5961.

[35] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowledge-Based Systems*, vol. 86, pp. 11–20, 2015.

[36] J. H. Mathews, K. D. Fink *et al.*, *Numerical methods using MATLAB*. Pearson prentice hall Upper Saddle River, NJ, 2004, vol. 4.

[37] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[38] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290.

[39] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[40] A. HSL, "collection of fortran codes for large-scale scientific computation," *See http://www. hsl. rl. ac. uk*, 2007.

[41] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.