2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

A Safe and Efficient Timed-Elastic-Band Planner for Unstructured Environments

Haoyu Xi[1,2], Wei Li[1,3*], Fangzhou Zhao[1], Liang Chen[1], Yu Hu[1,3*]

***Abstract*— In unstructured environments with complex obstacles and obscure road boundaries, the local planner faces more severe challenges in terms of safety and real-time performance. In order to fulfill these emerging requirements, we propose a novel Timed- Elastic-Band approach for unstructured environments, abbreviated as TEB-U. This approach incorporates a free space extraction optimization module for 2D occupancy grid maps, which efficiently transforms irregular free space boundaries into polygons and restrains robots within the boundaries. Moreover, a dynamic global point adjustment module is designed to adaptively correct the trajectory points obtained from the global planner, thereby enabling robots to travel along the centerline of free space and providing a better initial trajectory for subsequent modules. To reduce the computational cost, we replace the obstacle constraint of TEB with the boundary constraint in hyper-graph optimization. We evaluate our planner in three distinct scenarios, and the results show that TEB-U improves the average success rate by 21% and reduces the planning time by 23% compared to TEB in unstructured road, which demonstrates its safety and efficiency.**

I. INTRODUCTION

Planning in unstructured environments such as rural roads and rugged terrain is challenging. These scenarios lack clear driving boundaries or lanes, and the free space is often difficult to recognize. In addition, vehicles or wheeled mobile robots often encounter irregularly shaped obstacles, for instance, shrubs, trees, and mounds, which may cause potential hazards for navigation. Therefore, existing local planning methods in structured roads are not suitable. This paper aims to address these problems and ensure the safety of off-road robots and vehicles in unstructured environments.

Global path planning methods can provide a complete trajectory from the initial point to the terminal point. However, due to the absence of the local map and incomplete environmental information, these methods cannot deal with obstacles that are not in the map and fail to incorporate real-time sensor observations. To ensure safety, local planning methods become indispensable. Neural network approaches based on sensor observations [1], [2] are supervised, and they require a large amount of training data [3], which limits their generalization to multiple terrains in unstructured scenarios. Reinforcement learning methods [4], [5] rely on interacting with the environment, but training directly in a real-world environment can cause dangers. Thus the training process is usually replaced in simulators, which may lead to sim-to-real problems. In rule-based local planning methods, the Dynamic Window Approach (DWA) [6] is a sampling-based method that samples from the motion space and complies with kinematic constraints, but DWA is not forward-looking. The Timed-Elastic-Band (TEB) [7], [8] is a multi-objective optimization algorithm that incorporates different constraints into the planning problem. TEB has the characteristics of online optimization and can adapt to different robot models. The algorithm produces smooth trajectories with temporal information, which are suitable for complex environments and motion tasks.

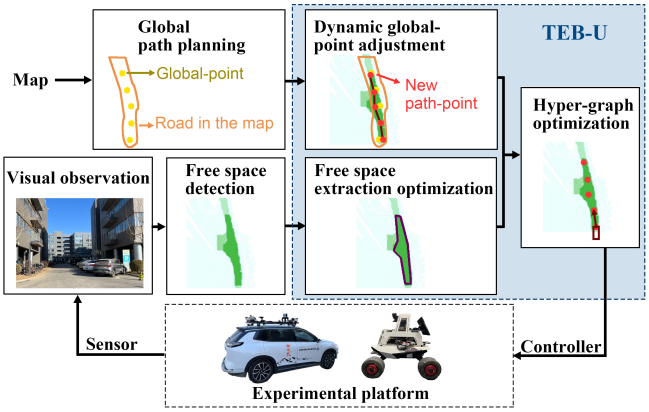


Fig. 1. This picture shows the TEB-U framework, which is contained in the blue block. TEB-U consists of three main steps: (1) free space extraction optimization, (2) dynamic global point adjustment, and (3) hyper-graph optimization. The trajectory points are generated by global path planning, and the free space is obtained through the free space detection algorithm. The dynamic global point adjustment module generates path points and candidate trajectories, while the free space extraction optimization module obtains free space boundaries. All information is incorporated into the hyper-graph optimization module as constraints. The predicted position and velocity are calculated by the optimization module and sent to the controller after being converted into control commands.

TEB requires several candidate trajectories for optimization, but the initial candidate trajectory of TEB is a straight line from the starting point to the destination [7]. If the initial trajectory intersects with obstacles, it may be unsafe and infeasible, according to the gradient optimization. To solve this problem, Rösmann et al. [9] uses a sampling-based exploration strategy called Probabilistic Road-Map (PRM) [10], [11] to generate candidate trajectories of distinctive topologies. However, this method has a high time cost in large environments because it has to search multiple trajectories. Smith et al. propose the egoTEB [12], which generates candidate trajectories through a gap-based

[1]Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
[2]Hangzhou Institute for Advanced Study, UCAS, Hangzhou 310024, China.
[3]University of Chinese Academy of Sciences, Beijing, 100049, China.
*Correspondence: Wei Li, liwei2019@ict.ac.cn, Yu Hu, huyu@ict.ac.cn

approach. The initial candidate trajectories of egoTEB are influenced by gaps and trajectory points obtained from global planner. However, due to the lack of local information, there is a deviation between the trajectory points and the road centerline, which may lead to potential danger for the initial candidate trajectories. We propose a novel method that corrects the trajectory points generated by global path planning to the centerline of free space, thereby ensuring collision avoidance. Subsequently, the theta* [13] algorithm is employed to compute an optimal path from the current position to the next trajectory point, and this path is then adopted as the new candidate trajectory.

In unstructured environments, common obstacles, such as trees and soil pits, pose challenges for local planning due to their irregular shapes. The TEB algorithm requires traversing the entire costmap and converting all obstacles into points, lines, or polygons before inputting them into the optimization. This leads to high computational costs, which makes the robot unable to adapt to dynamic changes in environments and may compromise its safety. To improve the real-time performance of the planning, Lu et al. [14] implement the layered costmaps. Each layer includes an obstacle or constraint and then modifies the master costmap, but the obstacle updates slowly when the perception range is large. The Convex Elastic Smoothing (CES) [15] is an improvement of the Elastic-Band (EB) [16] approach with better computational efficiency. It uses convex programming to handle shape and speed optimization. However, CES still has limitations in terms of both control feasibility and trajectory smoothness in some extreme cases. ConvexHull [17] and ConcaveHull [18] can transform irregular obstacles into convex or concave polygons. But for irregularly shaped free space, convex polygons may not be able to obtain the true boundaries of the free space, while concave polygons are accompanied by a large time overhead. Therefore, to make the process of TEB planning more efficient, this paper utilizes a polygon fitting method to extract the free space boundary from a 2D occupancy grid map and designs it as new TEB constraints.

To address the problems of local planning in unstructured environments, we propose TEB-U, a novel improvement of TEB that ensures safety and efficiency. The framework is shown in Fig. 1. To ensure the safe travel of the robots, TEB-U applies the dynamic global point adjustment module, adding the jerk constraint to hyper-graph optimization. To improve the efficiency, we design the free space extraction optimization module, replacing the obstacle constraint with the boundary constraint. The main contributions of this paper are:

- Present the free space extraction optimization method that efficiently obtains free space boundaries, which reduces the computational cost of the optimization.
- Propose the global point dynamic adjustment method that adaptively corrects the future trajectory points, which decreases the likelihood of getting stuck in a local minima, allowing the robot to travel more safely.
- Deploy our approach on a vehicle and a wheeled mobile robot and conduct real-world scenario experiments. Experiments demonstrate that TEB-U performs better in terms of safety and efficiency.

II. RELATED WORK

In unstructured environments, the variety of surfaces and the complexity of road conditions have a significant influence on planning. There are various approaches, including global and local planning, that attempt to address these challenges. Jiang et al. [19] use accessible areas to design a two-layer global path planning approach. The first layer uses an improved A* algorithm to generate the global path, and the second layer uses quadratic programming for path smoothing. However, when the local map is incomplete, the global path planning method cannot find the optimal path. Yang et al. [20] utilize the deterministic sampling-based method to solve sparse waypoints navigation in off-road environments. But they do not include velocity and acceleration restrictions. Feng et al. [21] use a predefined global reference line to generate multiple offsets and select the one with the minimal cost for local planning. This work allows the small-medium-sized vehicle to travel fast in off-road environments, which shows that modifying the global reference line can make the vehicle drive smoothly. Inspired by these works, this paper implements global point adjustment and leverages the free space to restrain the robot, thereby guaranteeing its safe travel.

TEB is a local planning algorithm that incorporates the dynamic constraints of robots while directly correcting the trajectories. It has been successfully applied to different tasks. Zha et al. [22] remove the time constraints and add smoothing and curvature constraints in TEB for car-like robots and post-process the generated trajectories using dynamics and velocity corrections. Gong et al. [23] propose orientation-aware timed-elastic-band (OATEB) for multi-task execution. OATEB aims to solve orientation transition tasks and position transition tasks in parallel. Schulze et al. [24] apply TEB to wheelchair robots by adding new non-holonomic and kinodynamic constraints and implement a real-time dual-arm controller for executing steering commands. Chung et al. [25] propose Distributed Timed Elastic Band (DTEB), which synergizes TEB with Priority Planning (PP) [26] for multi-robot systems. This planner can be used by the robot team to avoid imminent collisions and resolve potential deadlocks. However, improving the computational efficiency of TEB to better adapt it for complex environments is still a crucial challenge.

Collision avoidance is a common problem in local planning, which requires computing the distance between the robot model and the obstacles. In 2D grid maps, using only discrete grid points to represent the entire obstacle ignores the shape information of the obstacle and leads to a lot of redundant calculations. TEB provides some methods that can represent obstacles using geometric primitives such as points, lines, and polygons. ConvexHull and ConcaveHull can transform irregular obstacles into polygonal shapes. Both methods cluster the obstacle points first, but ConvexHull

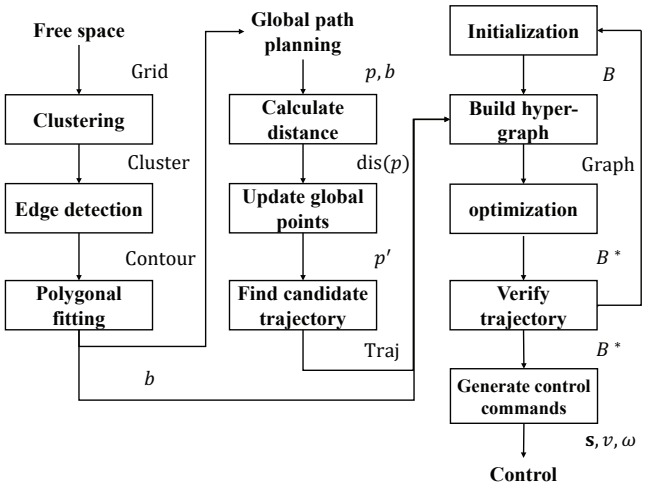


Fig. 2. TEB-U flowchart

generates convex polygons while ConcaveHull generates concave polygons. But they have a large time overhead for processing irregular obstacles in unstructured environments.

III. METHOD

This section presents the TEB-U method, which is organized as follows: Section III-A describes the overall flowchart of the TEB-U framework. Section III-B introduces the hyper-graph optimization module in TEB-U process. Section III-C explains the free space extraction optimization. Section III-D develops a description of the dynamic global point adjustment method.

A. The overall framework of TEB-U

TEB-U is designed to plan an optimal trajectory for the robot. The trajectory should satisfy constraints that include non-holonomic, minimum turning radius, velocity, acceleration, time, jerk, and boundary. The trajectory should also be computed within a limited time frame to ensure real-time planning. The TEB-U planner consists of three main modules: free space extraction optimization, dynamic global point adjustment, and hyper-graph optimization. The hyper-graph optimization receives the obstacle information and global points from the previous two modules and solves a sparse optimization problem using the g2o [27] framework. The output is the planned trajectory for the next 2 seconds, including the positions and velocities of the robot. The flowchart of TEB-U is illustrated in Fig. 2.

B. Hyper-graph optimization in TEB-U process

TEB is a method for optimizing the trajectory of the robot, and it represents the trajectory as a sequence of n intermediate robot configurations:

$$\mathbf{s}_i = (x_i, y_i, \beta_i)^T \tag{1}$$

where x_i, y_i, and β_i are the robot position and steering angle at time i. The x-axis aligns with the forward direction of the robot, while the y-axis points to the left of the robot. TEB also assigns a time interval ΔT_i to each pair of consecutive configurations, resulting in $n-1$ time intervals. The total time of the trajectory is given by:

$$T = \sum_{i=1}^{n-1} \Delta T_i \tag{2}$$

TEB aims to minimize the total travel time while satisfying the kinematic and dynamic constraints of the robot. Referring to [7], we formulate the optimization problem as follows:

$$B := \{\mathbf{s}_1, \Delta T_1, \mathbf{s}_2, \Delta T_2, ..., \mathbf{s}_{n-1}, \Delta T_{n-1}, \mathbf{s}_n\} \tag{3}$$

$$B^* = \arg \min_{B \setminus \{\mathbf{s}_1, \mathbf{s}_n\}} f(B) \tag{4}$$

where B is the set of parameters to be optimized and $f(B)$ is the objective function.

TEB-U uses the penalty function to penalize the violation of the constraints:

$$e\left(z, z_r, \epsilon, S, N\right) \simeq \begin{cases} \left(\frac{z-(z_r-\epsilon)}{S}\right)^N & \text{if } z > z_r - \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where z is associated with the state of each constraint, z_r is the reference value, ϵ is the offset, S is the scaling factor, and N is the polynomial order. The penalty function is zero when the constraint is satisfied and changes rapidly when the constraint is violated.

The non-holonomic kinematic $\mathbf{h}_i\left(\mathbf{s}_{i+1}, \mathbf{s}_i\right)$ requires the angle between the configuration $\mathbf{s}_i$ and the direction $\mathbf{d}_i$ to be equal to the corresponding angle at the configuration $\mathbf{s}_{i+1}$. $f_{\mathbf{h}}$ denotes the TEB-U non-holonomic constraint.

$$\mathbf{d}_i = \left[x_{i+1} - x_i, y_{i+1} - y_i, 0\right]^\top \tag{6}$$

$$\mathbf{h}_i\left(\mathbf{s}_{i+1}, \mathbf{s}_i\right) = \left(\begin{bmatrix} \cos\left(\beta_i\right) \\ \sin\left(\beta_i\right) \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\left(\beta_{i+1}\right) \\ \sin\left(\beta_{i+1}\right) \\ 0 \end{bmatrix} \right) \times \mathbf{d}_i \tag{7}$$

$$f_{\mathbf{h}} = e\left(\mathbf{h}_i(\mathbf{s}_{i+1}, \mathbf{s}_i), 0, \epsilon, S, N\right) \tag{8}$$

For car-like robots, the motion of the robot is further limited by the minimum turning radius between successive configurations. Let f_r denote the minimum turning radius constraint, d_r the turning radius, and d_{minr} the minimum turning radius.

$$f_r = e\left(-d_r, -d_{minr}, \epsilon, S, N\right) \tag{9}$$

The velocity vector $\mathbf{v}$ includes linear velocity v and angular velocity ω of the robot. They are computed from the sequence of configurations as follows:

$$v_i = \frac{1}{\Delta T_i} \left\| \begin{pmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{pmatrix} \right\| \tag{10}$$

$$\omega_i = \frac{\beta_{i+1} - \beta_i}{\Delta T_i} \tag{11}$$

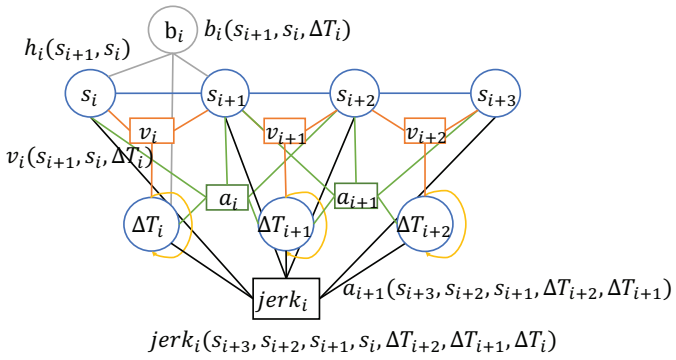


Fig. 3. TEB-U hyper-graph

The acceleration vector $\mathbf{a}$ can denote as linear acceleration and angular acceleration. They are computed from the sequence of configurations as follows:

$$\mathbf{a}_i = \frac{2\left(\mathbf{v}_{i+1} - \mathbf{v}_i\right)}{\Delta T_i + \Delta T_{i+1}} \tag{12}$$

In order to ensure the time for the whole trajectory is as short as possible, TEB-U minimizes the square of the sum of all the time differences.

$$f_{time} = \left(\sum_{i=1}^{n-1} \Delta T_i\right)^2 \tag{13}$$

In order to prevent the robot from generating a series of instabilities, such as sudden sprints and braking, we introduce the jerk constraint. The jerk can be expressed as:

$$\mathbf{jerk}_i = \frac{\mathbf{a}_{i+1} - \mathbf{a}_i}{(0.25\Delta T_i + 0.5\Delta T_{i+1} + 0.25\Delta T_{i+2})} \tag{14}$$

consider the penalty function:

$$f_{\mathbf{jerk}} = e\left(\mathbf{jerk}, \mathbf{jerk}_{max}, \epsilon, S, N\right) \tag{15}$$

$f(B)$ is a objective function that incorporates various constraints. The cost function can be written as the weighted sum of different terms, such as:

$$\begin{aligned} f(B) = {} & W_r * f_r + W_{\mathbf{h}} * f_{\mathbf{h}} + W_{time} * f_{time} \\ & + W_{\mathbf{v}} * f_{\mathbf{v}} + W_{\mathbf{a}} * f_{\mathbf{a}} + W_{\mathbf{jerk}} * f_{\mathbf{jerk}} \\ & + W_b * f_b \end{aligned} \tag{16}$$

where W is the weight and f is the penalty function corresponding to the constraint. b denotes the boundary, and f_b denotes the boundary constraint, they are introduced in Section III-C. The hyper-graph for the whole method is shown in Fig. 3.

C. Free space extraction optimization

To better represent obstacles in unstructured environments, this work proposes the free space extraction optimization method. The goal is to treat the free space boundaries as polygons to optimize obstacle processing. This method takes 2D occupancy grid maps as input and produces a set of polygon points as output. The implementation of the free space extraction method is described below:

1) Grid point clustering: for the free space 2D occupancy grid map, obtain the set of points at the edge of the free space and use the density-based spatial clustering of applications with noise (DBSCAN) [28] algorithm to cluster all the points into several clusters.

2) Contour extraction: for each cluster, apply the edge detection algorithm to identify the edges, and then record these edges as contours based on the edge data.

3) Polygon fitting: for each contour, use the Douglas-Peucker (DP) algorithm [29] to get new polygons, and the resulting polygons are used as the boundary constraint.

This work abandons the idea of using minimum convex or concave packages to extract obstacle boundaries and instead uses the DP algorithm. DP algorithm can fit the edges well, and the precision can be adjusted. Compared to ConvexHull and ConcaveHull, there is no need to add extra time overhead, and the time complexity is $O(nlogn)$, where n denotes the number of points.

The boundary constraint is designed to make better use of the free space boundary information. This constraint considers both the velocity and the distance from the extracted boundary of the robot, based on the fact that higher velocity increases the risk of lane departure. Therefore, it prevents the robot from driving out of the road boundary and keeps it within free space. The flow of the free space extraction optimization algorithm is shown in Algorithm 1. d_b is the distance from each polygon edge to configuration $\mathbf{s}_i$ separately. Set a threshold d_{minb}, if d_b is greater than or equal to d_{minb}, then establish a boundary constraint. The boundary constraint is also associated with the linear velocity v of the robot. Calculate v_i from $\mathbf{s}_i$, $\mathbf{s}_{i+1}$, ΔT_i. For internal obstacles in free space, d_b is related to the distance of the polygon center from the robot. There is the constraint:

$$f_b = v * e\left(-d_b, -d_{minb}, \epsilon, S, N\right) \tag{17}$$

D. Dynamic global point adjustment

The dynamic global point adjustment is proposed to ensure the safe travel of the robots. The inputs of this method consist of boundaries b, which are the outputs of free space extraction optimization, and the trajectory point set g. For each trajectory point p in g, adjust it to the centerline of free space to get the new trajectory point p'. Referring to experiments by [30], we use the theta* algorithm to search out a path from the current position to p' as the optimal candidate trajectory of TEB-U. Considering that theta* has a high time overhead, we separate theta* from the planning module and run them in parallel.

To better adapt to the changing road conditions in unstructured environments, the value of threshold $dis_{threshold}$ is adaptively adjusted based on historical information about previous distances dis. We use $dis_{threshold}$ to constrain the distance of global point adjustment, which is initially set with a value. If p' is simply placed in the centerline of free space, then the trajectory point might shake dramatically from side to side. Large shakes in the x-y plane can lead to large deviations of the robot from the direction of the

Algorithm 1: Free space extraction optimization

Input: list GridList, node list Poses, d_{minb}
Output: list PolygonList

Initialize: empty list EdgeNodeList, PolygonList;
for *node* nd *in GridList* **do**
  if nd *is the boundary or* nd *is the obstacle inside the free space* **then**
    Append nd into EdgeNodeList;
  end
end
2D-array Clusters = DBSCAN($EdgeNodeList$);
for *array* $cluster$ *in Clusters* **do**
  list Contours = EdgeDetection($cluster$);
  list Polygons = DP($Contours$);
  list NewPolygons ← Delete repeat edges in Polygons;
  for *node* $pose$ *in Poses* **do**
    d_b = Distance($pose$, $NewPolygons$);
    if $d_b \geqslant d_{minb}$ **then**
      Add boundary constraints f_b;
    end
  end
  Append Polygons into PolygonList;
end
return PolygonList;

Algorithm 2: Dynamic global point adjustment

Input: list PolygonList, $maxnum$, dis_{min}, $dis_{threshold}$
Output: list NewTrajectory

Initialize: empty list NewTrajectory, point p';
$num = 0$, $dis = 0$;
p is the next trajectory point;
dis_c is the distance of the robot from p;
while $num \leqslant maxnum$ *and* $dis_c \geqslant dis_{min}$ **do**
  for *list Polygons in PolygonList* **do**
    Find the point p_1 in the Polygons that is closest to p;
    Extend p_1p and the other interaction with Polygons is p_2;
    Take the centre of line segment p_1p_2 as p_{tmp};
    dis = GetDistance(p, p_{tmp});
  end
  if $dis \leqslant dis_{threshold}$ **then**
    Update $p' \leftarrow p_{tmp}$;
  else
    Draw a circle with p as the centre and $dis_{threshold}$ as the radius;
    p_3 is the intersection of the circle with the line segment $p_{tmp}p$;
    Update $p' \leftarrow p_3$;
  end
  $dis_{threshold} \leftarrow$ average dis of the previous 10 steps;
  $num = num + 1$;
  Update distance dis_c through new p';
  NewTrajectory = ThetaStar(p');
end
return NewTrajectory;

road. This deviation may affect the segmentation of the road ahead, which may lead to an increasingly large error.

We stop adjusting a trajectory point when the distance of the robot from the p' is less than dis_{min} or the number of adjustments exceeds the maximum number. The purpose is to prevent situations that a trajectory point is repeatedly adjusted and ends up too far from the desired position in some environments. The process of the algorithm shown in Algorithm 2.

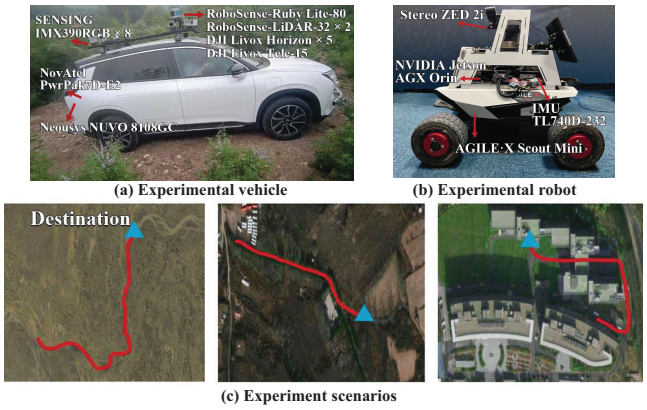
IV. EXPERIMENTS

This section validates the efficiency and safety of our proposed method. We measure the time overhead of different algorithms for free space and obstacle processing, and conduct the real-world driving tests.

A. Experimental platform and environment

Our experimental platforms include a sport utility vehicle and a wheeled mobile robot. The vehicle is equipped with LiDARs and cameras, and the planning methods are executed on a laptop equipped with the AMD Ryzen 7 6800H with Radeon Graphics@3.20GHz, 16G RAM, connected to the vehicle through a switch. The robot has a stereo camera ZED2i and an NVIDIA AGX Orin on-board computer. We use the Robot Operating System (ROS) in Ubuntu Linux for communication. Experimental platforms and environments are shown in Fig. 4.

We conduct our experiments in three types of scenarios: Gobi, Forest, and unstructured road in a sci-tech park (abbreviated as Park in the following text):



(a) Experimental vehicle (b) Experimental robot

(c) Experiment scenarios

Fig. 4. (a) Experimental sport utility vehicle, (b) Experimental wheeled mobile robot, (c) Experiment scenarios, from left to right: Gobi, Forest, and unstructured road in a sci-tech park. Red lines indicate the vehicle paths.

1) The experimental path in Park scenario is about 200m long and includes two right-angled turns, flower beds, and fences. It also contains a car park on the left side, as well as obstacles such as stone bricks and soil mounds.

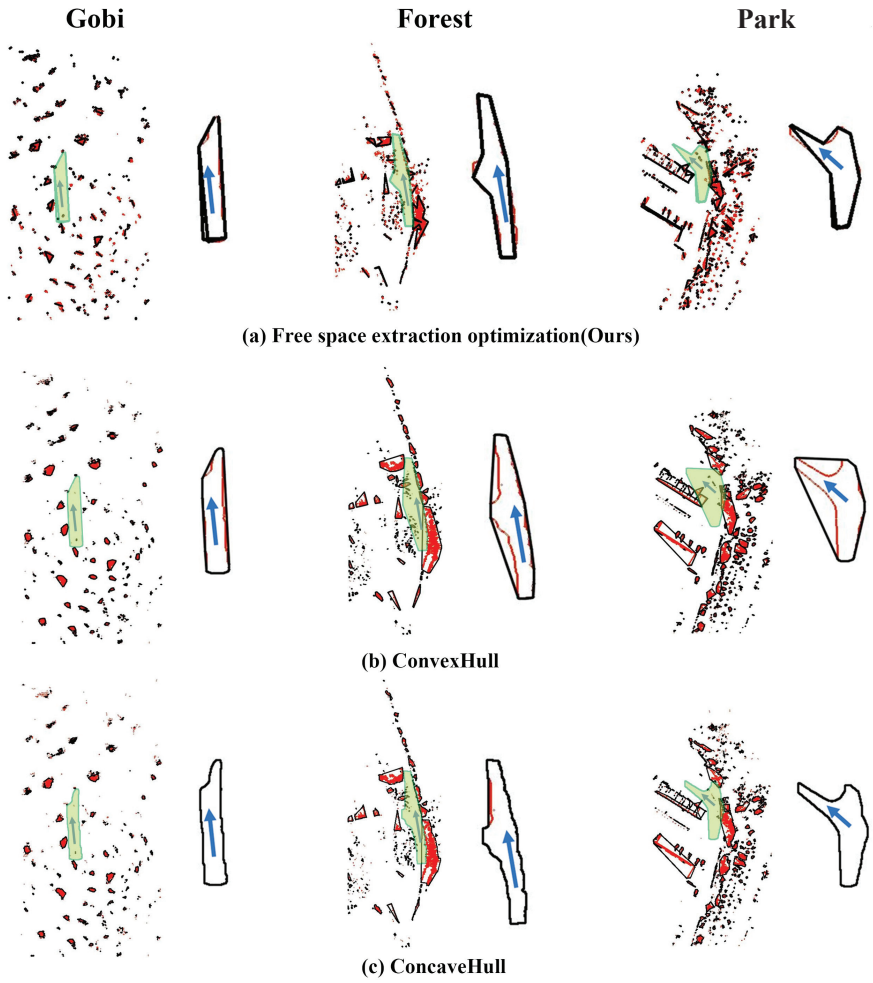


Fig. 5. Obstacles and free space extraction examples in Gobi, Forest, and Park respectively. (a) using free space extraction optimization (Ours), (b) using ConvexHull [17], and (c) using ConcaveHull [18]. In the left image, red points indicate obstacles, and black lines indicate the obstacle boundaries obtained by the algorithm; in the right image, red points indicate edges of free space, and black lines indicate the boundaries of free space obtained by the algorithm. The blue arrow indicates the driving direction of the vehicle. The view of images is a bird's-eye view.

2) The experimental path in Gobi scenario is about 2km long, with large and small sand pits around the road. We also artificially place some obstacles on the road.

3) The experimental path in Forest scenario is about 1km long, with trees and shrubs on both sides of the road. It also contains stones and other obstacles on the road.

B. Experiment result analysis

The experimental vehicle obtains a 2D occupancy grid map of size 500 × 750 with a resolution of 0.2m from the perception module. The grid map covers 100m in front, 50m behind, and 50m to the left and right of the vehicle. We run our free space extraction method, ConvexHull, ConcaveHull, at 4 Hz and the planning frequency is also 4 Hz. In the driving experiments, we set up several fixed groups of start and finish points and conducted 20 experiments for each method. The maximum velocity of the vehicle is set to $1m/s$. Other relevant parameters are as follows: $d_b = 8.0m$, $d_{minr} = 6.0m$, $dis_{threshold} = 5.0m$, $\mathbf{jerk_{max}} = [2.0m/s^3, 1.0rad/s^3]$, $dis_{min} = 8.0m$, $maxnum = 30$.

1) Obstacle representation analysis

We explore the effect of different methods to represent the obstacle using free space boundaries, or obstacle boundaries, and compare our proposed free space extraction method with ConvexHull and ConcaveHull. Three methods are applied to process them in each of the three scenarios. The results are shown in Fig. 5. ConvexHull fails to obtain free space boundaries. ConcaveHull has a good representation of free space boundaries, but both ConvexHull and ConcaveHull lose some details of obstacle boundaries. They generate polygons that obscure parts of the road when encountering curves, thus treating that part of the road as the obstacle. Our free space extraction method can accurately and efficiently represent the boundaries. Table I shows the obstacle representation average processing time of each method. In three scenarios, the processing time of using free space is lower than using obstacles. Our proposed method has the lowest time overhead for extracting free space boundaries. Therefore TEB-U uses the free space. ConvexHull has the lowest time overhead for extracting obstacle boundaries, but it does not work well for their representation at corners.

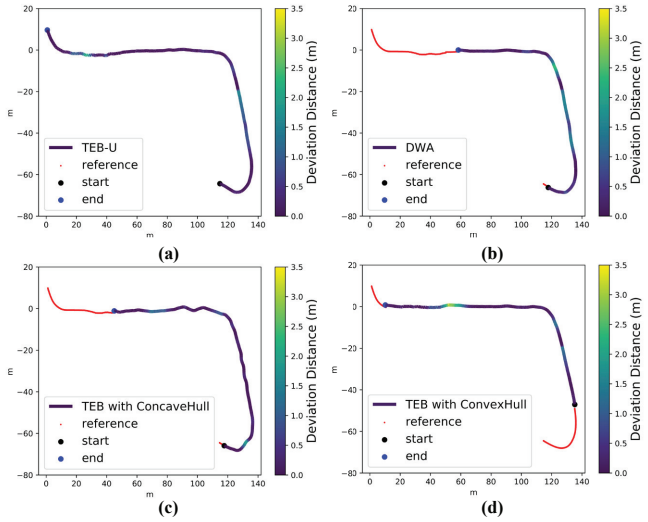


Fig. 6. Driving trajectories. In Park scenario, (a) TEB-U trajectory, (b) DWA trajectory, (c) TEB with ConcaveHull trajectory, and (d) TEB with ConvexHull trajectory. The trajectory color indicates the deviation distance from the trajectory of human driver. The red points are the reference human trajectory.

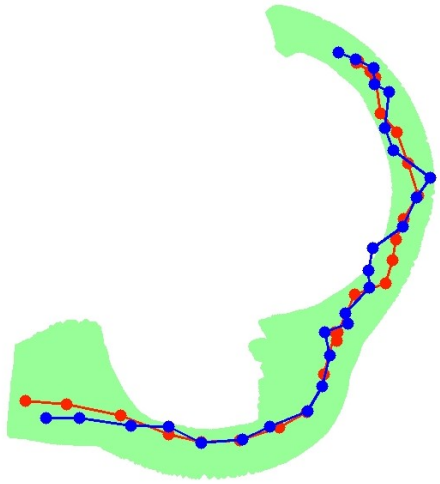


Fig. 7. Results of dynamic global point adjustment in Park scenario. The green area denotes the free space, the blue points are the trajectory points obtained from the global planner, and the red points represent the new trajectory points generated by the dynamic global point adjustment. Points of the same color are connected by lines.

TABLE I

OBSTACLE REPRESENTATION PROCESSING TIME (UNIT: MS)

		ConvexHull	ConcaveHull	Ours
Obstacle	Gobi	**17.88**	355.62	124.73
	Forest	**12.92**	283.95	172.45
	Park	**9.87**	473.21	60.97
Free space	Gobi	7.02	237.40	**4.53**
	Forest	6.31	86.93	**5.62**
	Park	4.28	149.94	**3.73**

TABLE II

AVERAGE PLANNING TIME (UNIT: MS)

		TEB with ConvexHull	TEB with ConcaveHull	Ours
Obstacle	Gobi	674.54	1652.10	**514.77**
	Forest	1439.36	1923.67	**965.42**
	Park	1023.45	1746.32	**719.35**
Free space	Gobi	249.62	663.98	**182.73**
	Forest	230.06	653.33	**216.02**
	Park	328.29	569.34	**253.68**

TABLE III

DRIVING EXPERIMENTS

Method	Success Rate(%)	Path Length(m)	RMSE
DWA	55	64.10	0.79
TEB with ConvexHull	65	60.96	0.98
TEB with ConcaveHull	70	90.27	0.75
TEB-U	**85**	**134.67**	**0.63**

2) Planning time analysis

We compare the performance of three methods: TEB with ConvexHull, TEB with ConcaveHull, and TEB-U (ours), in terms of their average planning time in three scenarios. Three methods are used to deal with the free space and obstacles in each scenario. Average planning time results are shown in Table II. TEB-U has the lowest planning time for dealing with both free space and obstacles. We also find that using free space to replace the obstacle reduces the planning time compared to using obstacle representation directly. Free space can better satisfy the requirements of real-time planning. From Table II, when processing free space, TEB-U reduces the time by 27%, 6%, and 23% in the three scenarios compared to TEB with ConvexHull. Planning time is influenced by the number of vertices and edges in the g2o hyper-graph. Among these, the edges associated with obstacles have the most significant impact. We observe that TEB-U generates the fewest edges when dealing with obstacles and free spaces, thereby leading to a reduction in planning time.

3) Driving experiment analysis

We use experimental vehicles to conduct real-world driving experiments in Park scenario and evaluate the performance of DWA, TEB with ConvexHull, TEB with ConcaveHull, and TEB-U methods. We use free space instead of obstacles. The deviation of trajectories generated by each method from the reference trajectory is measured using the Root Mean Square Error (RMSE). The reference trajectory is obtained by a human driver. Table III shows the success rate, average successful driving distance, and RMSE for the four methods. This indicates that TEB-U driving process is safer than other comparison methods. Fig. 6 shows the longest successfully driving trajectory for each method, along with the corresponding deviation distance. The trajectories generated by our method are closer to the reference trajectory. Fig. 7 illustrates historical information from the dynamic global point adjustment module in a portion of Park scenario. The new trajectory points generated by the dynamic global point adjustment are closer to the centerline of the free space. For more vehicle and wheeled mobile robot real-world driving experiments, please refer to the attached video.

V. CONCLUSIONS

In this paper, we propose TEB-U, a novel planner for unstructured environments that can efficiently generate safe trajectories. Our planner employs a free space extraction optimization method to reduce the TEB computation time and a dynamic global point adjustment method to correct future trajectory points to the centerline of free space. TEB-U also introduces a new candidate trajectory method that uses theta* to plan a feasible path for the adjusted global points. We demonstrate the effectiveness of our planner through real-world experiments in three unstructured scenarios, and the results show that our proposed method is better in terms of safety and real-time performance. For future work, we plan to extend our planner to handle dynamic obstacles.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant No. 62003323 and No. 62176250, and Beijing Natural Science Foundation (L243008).

REFERENCES

[1] Samuel Triest, Matthew Sivaprakasam, Sean J. Wang, Wenshan Wang, Aaron M. Johnson, and Sebastian Scherer. Tartandrive: A large-scale dataset for learning off-road dynamics models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2546–2552, 2022.

[2] Tsun-Hsuan Wang, Alexander Amini, Wilko Schwarting, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. Learning interactive driving policies via data-driven simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7745–7752, 2022.

[3] Ahmad Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017:70–76, 2017.

[4] Daniel W. Carruth, Clayton T. Walden, Christopher Goodin, and Sara C. Fuller. Challenges in low infrastructure and off-road automated driving. In *2022 Fifth International Conference on Connected and Autonomous Driving (MetroCAD)*, pages 13–20, 2022.

[5] Philémon Brakel, Steven Bohez, Leonard Hasenclever, Nicolas Heess, and Konstantinos Bousmalis. Learning coordinated terrain-adaptive locomotion by imitating a centroidal dynamics planner. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10335–10342, 2022.

[6] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

[7] Christoph Rösmann, Wendelin Feiten, Thomas Woesch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6, 2012.

[8] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686, 2017.

[9] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153, 2017.

[10] Erwin Schmitzberger, Jean-Louis Bouchet, Michel Dufaut, Didier Wolf, and René Husson. Capture of homotopy classes with probabilistic road map. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2317–2322, 2002.

[11] Léonard Jaillet and Thierry Siméon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research*, 27(11-12):1175–1188, 2008.

[12] Justin S. Smith, Ruoyang Xu, and Patricio Vela. egoteb: Egocentric, perception space navigation using timed-elastic-bands. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2703–2709, 2020.

[13] Kenny Daniel, Alex Nash, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39:533–579, 2010.

[14] David V. Lu, Dave Hershberger, and William D. Smart. Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–715, 2014.

[15] Zhijie Zhu, Edward Schmerling, and Marco 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)*, pages 835–842, 2015.

[16] M. Khatib, H. Jaouni, R. Chatila, and J.P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 2920–2925 vol.4, 1997.

[17] A.M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[18] J.-S Park and S.-J Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering*, 29:379–392, 2013.

[19] Junkai Jiang, Zeyu Han, Jinhao Li, Yuning Wang, Jianqiang Wang, and Shaobing Xu. Global path planning of ugvs in large-scale off-road environment based on improved a-star algorithm and quadratic programming. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7, 2023.

[20] Tian Yang, Guangming Xiong, Yu Zhang, Lei Yang, Bo Tang, Mengze Wu, and Jianwei Gong. A practical planning framework and its implementation for autonomous navigation in off-road environment. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1571–1576, 2019.

[21] Zhiqi Feng, Mengyin Fu, Wenjie Song, Xiaohui Tian, Yi Yang, and Meiling Wang. Decision making and local trajectory planning for autonomous driving in off-road environment. In *2020 3rd International Conference on Unmanned Systems (ICUS)*, pages 1180–1186, 2020.

[22] Tianjun Zha, Jian Wen, Yang Li, and Lei Sun. A local planning method based on graph optimization framework. In *2021 6th International Conference on Control, Robotics and Cybernetics (CRC)*, pages 80–84, 2021.

[23] Cheng Gong, Zirui Li, Xingyu Zhou, Jiachen Li, Junhui Zhou, and Jianwei Gong. Orientation-aware planning for parallel task execution of omni-directional mobile robot. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6891–6898, 2021.

[24] Martin Schulze, Friedrich Graaf, Lea Steffen, Arne Roennau, and Rüdiger Dillmann. A trajectory planner for mobile robots steering non-holonomic wheelchairs in dynamic environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3642–3648, 2023.

[25] Yiu Ming Chung, Hazem Youssef, and Moritz Roidl. Distributed timed elastic band (dteb) planner: Trajectory sharing and collision prediction for multi-robot systems. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10702–10708, 2022.

[26] Lynne E Parker. Path planning and motion coordination in multiple mobile robot teams. *Encyclopedia of complexity and system science*, pages 5783–5800, 2009.

[27] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page 226–231, 1996.

[29] John Edward Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proceedings, 5th International Symposium Spatial Data Handling*, pages 134–143, 1992.

[30] Niklas Persson, Martin C. Ekström, Mikael Ekström, and Alessandro V. Papadopoulos. On the initialization problem for timed-elastic bands. *IFAC-PapersOnLine*, 56(2):11802–11807, 2023.