IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2024, Yokohama, Japan.
Cite as T-ASE paper.

Improve Computing Efficiency and Motion Safety by Analyzing Environment with Graphics

Qianyi Zhang, Shichao Wu, Yuhang Jia, Yuang Xu, and Jingtai Liu, *Senior Member, IEEE*

***Abstract*—Exploring topologically distinctive trajectories provides more options for robot motion planning. Since computing time grows greatly with environment complexity, improving exploration efficiency and picking the optimal trajectory in complex environments are critical issues. To this end, this paper proposes a Graphic- and Timed-Elastic-Band-based approach (GraphicTEB) with spatial completeness and high computing efficiency. The environment is analyzed utilizing computer graphics, where obstacles are extracted as nodes and their relationships are built as edges. Three contributions are presented. 1) By assembling directed detours formed by nodes and segmented paths formed by edges, a generalized path consisting of nodes and edges derives various normal paths efficiently. 2) By multiplying two vectors starting from the obstacle point closest to the waypoint and the boundary point farthest from the waypoint, an novel obstacle gradient is introduced to guide safer optimization. 3) By assigning edges with asymmetric Gaussian model, a trajectory evaluation strategy is designed to reflect the motion tendency and motion uncertainty of dynamic obstacles. Qualitative and quantitative simulations demonstrate that the proposed GraphicTEB achieves spatial completeness, higher scene pass rate, and fastest computing efficiency. Experiments are implemented in long corridor and broad room scenarios, where the robot goes through gaps safely, finds trajectories quickly, and passes pedestrians politely.**

***Note to Practitioners*—The motivation stems from the fact that our daily cruising robot occasionally gets trapped in a corridor with piled obstacles or in a complex dynamic crowd due to the lack of a reliable trajectory. The solution is to search for more topologically distinctive trajectories and pick the optimal one. Considering that existing open-source approaches are either incomplete or highly time-consuming, a method for clustering and searching trajectories in the obstacle-occupied regions is proposed to achieve spatial completeness and high computing efficiency. In addition, an optimization technique and a trajectory selection strategy are proposed to improve motion safety. However, at present, the search is incomplete in the temporal-spatial dimension when dynamic obstacle are moving fast. How to perform a complete and fast search in temporal-spatial space will be developed in the future.**

***Index Terms*—Motion planning, Computer graphics, Timed Elastic Band (TEB), Homology class of trajectories**

I. Introduction

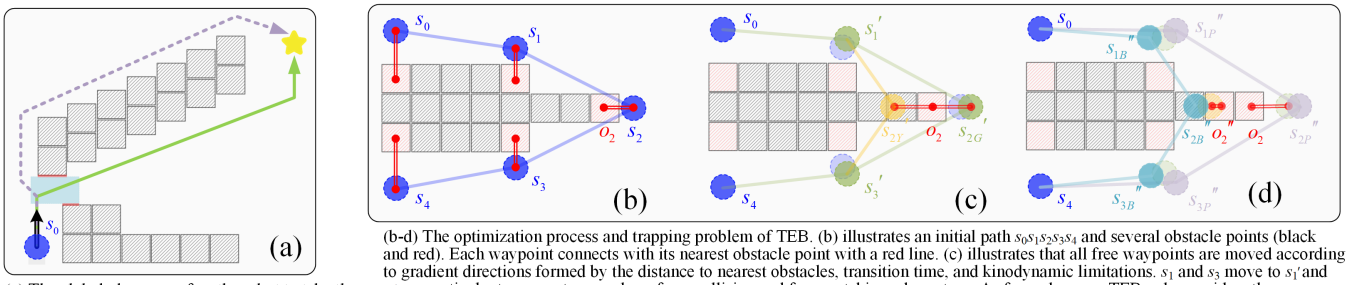
MOBILE robots are increasingly integrated into human life [1], [2]. The robot receives sensor data and computes kinodynamic-constrained trajectories in real-time [3]. This motion planning process can be completed by velocity-based methods that sample candidate trajectories in velocity space, such as Dynamic Window Approach [4] and Reciprocal Velocity Obstacle [5], or state-based methods that fit trajectories by candidate goal states, such as lattice planner [6]. However, they suffer from either a short horizon or tracking error. As a result, an optimization-based Timed Elastic Band (TEB) [7], [8], [9] forms a time-optimal trajectory in a long horizon and guarantees safe separation from obstacles.

However, as an inevitable problem of quadratic optimization, TEB could get stuck in a local minima due to the presence of obstacles, as illustrated in Fig.1(a): To reduce time-consuming, global path planning typically simply focuses on minimizing path length while ignoring the robot's dynamic constraints [10]. Given the constraints and energy costs, the ideal optimal trajectory may expect the global path to leap over certain obstacles during optimization, which is impossible due to the non-convexity of the obstacle distribution. As a result, trajectories are blocked near obstacles and fall into local minima. One solution is to find all topologically distinctive trajectories and select the best of them for the robot to execute.

The issue coming with "find all" is the requirement for computing efficiency. The robot keeps executing the previous command until the next planning is completed. Low planning frequency results in slow robot response and a high collision risk. Some approaches conserve computing time by sacrificing completeness [8], [11]. LRpoint-based TEB clusters obstacles along the vector from the robot to the target point and samples waypoints around the obstacles specifically. PRM-based TEB samples waypoints at random within the area restricted by the robot and the target. Their time complexities are $O(n)$. They are practical, but are not guaranteed to find all key waypoints to generate all topologically distinct trajectories. Others use Voronoi diagram to search for all waypoints [8], [12]. Although theoretically complete, the time complexity is up to $O(nlogn)$. In contrast, this paper achieves completeness in $O(n)$ by analyzing the obstacle distribution with computer graphics. Unlike existing methods for finding waypoints in collision-free regions, we extract obstacles as large nodes surrounded by special waypoints, taking both completeness and efficiency into account.

To search for topologically distinctive paths among waypoints, depth-first search is widely used although its searching time grows greatly with waypoint quantity. We speed up this procedure by searching among nodes instead of waypoints. A generalized path consisting of nodes can derive several

The authors are with the Institute of Robotics and Automatic Information System, Nankai University, Tianjin, 300350, China. {zhangqianyi, 1120200175, 2013628}@mail.nankai.edu.cn; liujt@nankai.edu.cn. This work is supported in part by National Key Research and Development Project under Grant 2019YFB1310604 and in part by National Natural Science Foundation of China under Grant 62173189.



(a) The global planner prefers the robot to take the shortest path (green), while the blue area requires the robot to slow down, and approaching the red lines are dangerous. The time-optimal path (purple) can be a better choice.

(b-d) The optimization process and trapping problem of TEB. (b) illustrates an initial path $s_0s_1s_2s_3s_4$ and several obstacle points (black and red). Each waypoint connects with its nearest obstacle point with a red line. (c) illustrates that all free waypoints are moved according to gradient directions formed by the distance to nearest obstacles, transition time, and kinodynamic limitations. s_1 and s_3 move to $s_1'$and s_3', respectively, to prevent s_1s_2 and s_2s_3 from collision and from s_2 taking a large turn. As for s_2, because TEB only considers the undirected distance between the waypoint and its nearest obstacle point, for two points $s_{2Y}'$and s_{2G}'having the equivalent distance to o_2, $s_{2Y}o_2$ and $s_{2G}o_2$ form the same gradient direction compared to the original s_2o_2. As $s_0s_1's_{2Y}'s_3's_4$ is shorter and spends less time, TEB optimizes the original $s_0s_1s_2s_3s_4$ into $s_0s_1's_{2Y}'s_3's_4$ instead of the expected $s_0s_1's_{2G}'s_3's_4$. In the following iterations like (d), $s_0s_1's_{2Y}'s_3's_4$ based trajectory $s_0s_{1B}''s_{2B}''s_{3B}''s_4$ is trapped into obstacles, while $s_0s_1's_{2G}'s_3's_4$ based trajectory $s_0s_{1P}''s_{2P}''s_{3P}''s_4$ can be increasingly better.

Fig. 1. Two issues of TEB: being stuck in local minima and being trapped in obstacles.

normal paths consisting of waypoints. The time consumption decreases from $O((\alpha n)!)$ to $O(n!)$, where α is the average number of waypoints around a node, usually large. Each path is assigned an H-signature [13], [14] according to the relationship to obstacles. Paths belonging to the same homology class have the same unique H-signature, and redundant ones will be deleted since they present the same result after optimization. The remained paths are topologically distinctive.

The paths are then optimized into trajectories, considering the time to the goal, the distance to obstacles, and the robot kinodynamic [8]. As for the obstacle item, the Euclidean distance between each waypoint and its nearest obstacle point is optimized to ensure safety. However, this local vision method that only considers a single obstacle point for each waypoint ignores the overall distribution of obstacles and may trap the trajectory into dense obstacles, as shown in Fig.1(b-d). To overcome this problem, this paper designs an obstacle gradient, and prevents the terrible trap.

All trajectories are scored based on the quality, and the best one will be executed by the robot. This traditional method only considers the trajectory itself and ignores the relationship between obstacles. The safety and stability of the trajectory cannot be fully guaranteed. Instead, this paper considers the motion uncertainty of dynamic obstacles and integrate robot planning and obstacle prediction to improve safety by a designed trajectory evaluation strategy [15].

In summary, our best contribution is an obstacle extraction method for analyzing the environment using computer graphics. On this basis, the computational efficiency and the motion safety are improved through three designs:

- A efficient waypoints sampling and topologically distinctive paths exploring approach.
- A robust trajectory optimization technique to improve success rate of the optimization.
- A trajectory evaluation strategy to select safer homology class trajectory in dynamic environment.

The remainder of the paper is organized as follows: Sec.II presents preliminaries and related work on TEB and trajectory homology class. The entire process from receiving global path to yielding a trajectory for the robot includes waypoints sampling, potential paths generation, trajectories optimization, and final trajectory selection. These four procedures are detailed in Sec.III, IV, VA, and VB, respectively. Sec.VI demonstrates the contributions of the former two in static scenarios and the latter in dynamic scenarios by qualitative comparison. Sec.VII verifies the proposed approach in the real world. Sec.VIII concludes the paper and discusses further work.

II. Preliminaries and Related Work

A. Timed Elastic Band

As a local motion planning method, TEB takes in a path and map information, and yields a collision-free and kynodynamic-feasible trajectory for the robot. Originally, a planning algorithm Elastic Band (EB) [16] regards the path as a sequence of waypoints $s_i = [x_i, y_i, \beta_i]^T \in R^2 \times S^1$: $Q = \{s_i\}_{i=0...n}$, where $n \in N$, (x_i, y_i) denotes the position of the waypoint, and β_i denotes the orientation of the waypoint. EB incrementally adjusts the waypoints with an internal force to contract the path and an external force to repeal the path from obstacles. TEB [7] augments the sequence by incorporating the time interval ΔT_i between adjacent waypoints, forming a new sequence: $\tau = \{\Delta T_i\}_{i=0...n-1}$. Each time internal ΔT_i indicates the transition time from s_i to s_{i+1}. Two sequence forms a trajectory tuple: $\gamma = (Q, \tau)$. TEB iteratively optimizes γ to minimize the time to the goal, while satisfying kinodynamic constraints of the robot and maintaining a safe separation from obstacles:

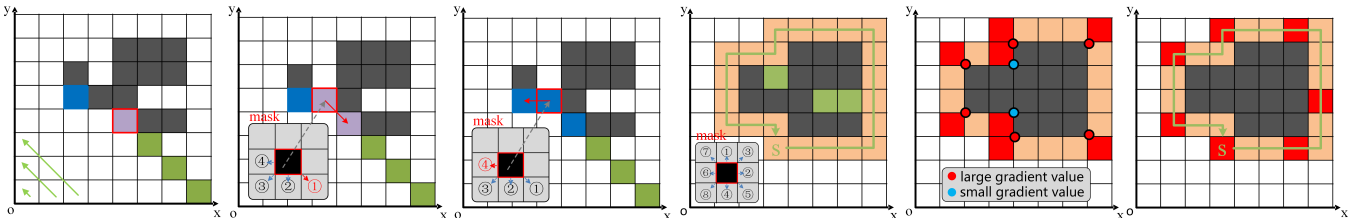
$$f(\gamma) = \sum_{i=0}^{n-1} \Delta T_i^2, \gamma^* = \arg\min_{\gamma} f(\gamma) \tag{1}$$

subjects to:

$$\begin{aligned} &s_0 = s_s, s_n = s_f, \Delta T_i > 0, h_i(s_{i+1}, s_i) = 0, o_i(s_i) \geq 0 \\ &v_i(s_{i+1}, s_i, \Delta T_i) \geq 0, \alpha_i(s_{i+2}, s_{i+1}, s_i, \Delta T_{i+1}, \Delta T_i) \geq 0 \end{aligned} \tag{2}$$

γ^* is the optimized trajectory. Initial waypoint s_0 and target waypoint s_n are fixed by given states s_s and s_f respectively. $h_k(s_{k+1}, s_k)$ constraints adjacent waypoints with the robot kinodynamic. o_k sustains a safe separation from obstacles. v_k and α_k limits velocities and accelerations respectively.

The plenty of hard constraints could render the optimization problem unsolvable. So the proposer of TEB [8]



(a) The last and current iterations generate two obstacle groups (blue and purple) respectively (b) The obstacle point outlined in red is merged into the purple obstacle group by the first point of the mask (c) The purple obstacle group is merged into the blue group by the fourth point of the mask (d) Closing operation fills small gaps (green) within the obstacle group. The boundary points are sorted by the mask (e) The corner points are identified (circle) and matched to boundary points (red) (f) The corner points are resampled to max the total gradient value

Fig. 2. Illustration of understanding environment with computer graphics. (a-c) extract and identifies adjacent obstacle points as an obstacle group. (d) extracts the boundary of the obstacle group and sorts boundary points counterclockwise. (e-f) explore the key corner points of the obstacle group.

transformed them into soft constraints with large weights by Lagrangian multipliers.

$$f(\gamma) = \sum_{i=0}^{n-1} (\Delta T_i^2 + w_1 h_i^2 + w_2 o_i^2 + w_3 v_i^2 + w_4 \alpha_i^2) \quad (3)$$

Each transition time ΔT_i is calculated by assuming that the robot runs from s_i to s_{i+1} at the maximum speed, so the trajectory is actually optimized by adjusting all the free waypoints $\{s_i\}_{i=1,\dots,n-1}$. Fig.1(b-d) illustrates an optimization process. In each iteration, waypoints move according to the gradient directions formed by the nearest obstacles, transition times, velocities and accelerations until the gradients are zero or the maximum number of iterations is reached. The figure also illustrates a remained issue that TEB understands the obstacle distribution only from local perspective, and could trap the trajectory into obstacles. Because the waypoints only care about nearby environment and adjacent waypoints, the problem has a sparse structure that can be efficiently solved by a ROS-friendly optimization framework g2o [17]. Due to the extensibility of the optimization framework of TEB, it has been applied to various fields. Yin [18] constructed special velocity constraints and jerk constraints to generate a smoother forward trajectory without backward for a wheeled-differential robot. Kenny [19] created a critical corner detector and incorporates the detected blind spots into TEB, improving safety when the robot approaches a non-visible area with hidden moving objects. Hoang [20] proposed a human-approaching robot navigation framework, the goal-oriented timed elastic band (GTEB). It generated a socially optimal trajectory for the robot by estimating the socially optimal approaching pose of the robot.

B. Homology Class of Trajectories

Exploring topologically distinctive paths works between global path planning and local trajectory planning. As the present of obstacle could trap the optimization into local minima, the exploration can provide multiple different initial paths for TEB and prevent local minima by optimizing all the paths in parallel. Researchers explore topologically distinctive paths by introducing the concept of homology class [13], [14], which is defined as: *Two paths Q_1 and Q_2 connecting the same start and goal points s_s and s_f, respectively, are homologous if and only if $Q_1 \sqcup -Q_2$ forms the complete boundary of a $2D$ manifold embedded in C not containing and intersecting any obstacle.* In short, the two paths are topologically distinctive if they do not belong to the same homology class.

There are three steps for generating topologically distinctive trajectories. The first is to generate sufficient waypoints. Kuderer [12] used Voronoi diagram [21] to sample waypoints, which ensures to find all the topologically distinctive paths (completeness), but the approach is high time-consuming and impractical for experimental application. Rösmann [8], [11] clustered obstacle points into huger obstacles [22] and sampled waypoints on the left and right of them. He further used probabilistic roadmaps (PRM) to randomly sample waypoints in the local map to make the algorithm more generalized. Zhou [23] proposed a path-guided optimization (PGO) to solve infeasible local minima, sampling points as guards when the existing path is not visible to them, and connecting guards to paths or other guards when the path is visible. Smith [24] proposed egoTEB to explore the impact of using egocentric perceptual spatial representations on local planning maps. The approaches are low time-consuming but cannot guarantee completeness. The second step is to search for paths within waypoints, and depth-first search is widely used at present. Each path is assigned an H-signature [13], [14] based on its relationship to the obstacles. Paths with the same H signature belong to the same homology class, and present the same result after optimization. So the redundant paths will be pruned. The third step is that TEB assigns time intervals for the remained paths and optimizes these trajectories.

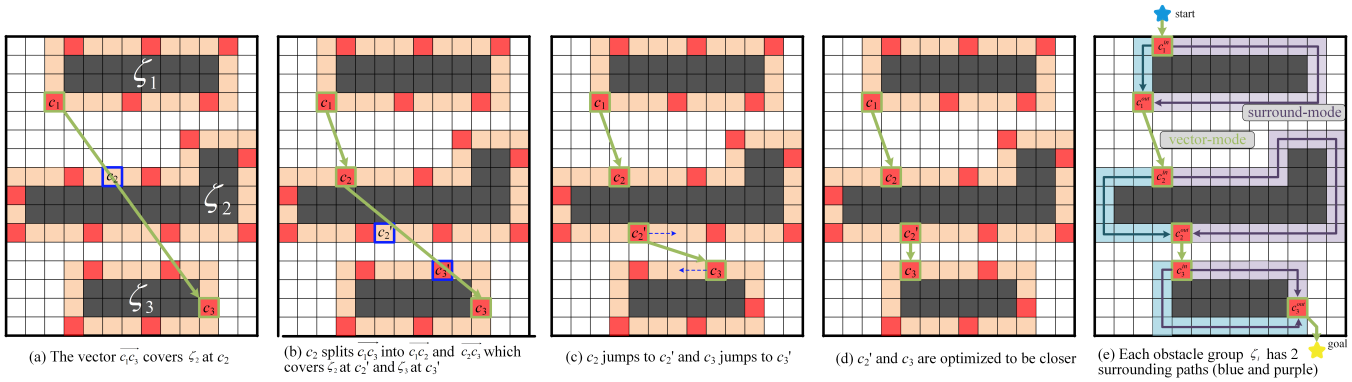
The key issue lies in the former two steps for exploring all topologically distinctive paths in a short time. Most existing algorithms explore paths in collision-free regions, resulting in high time-consumption or incompleteness. In contrast, this paper focuses on obstacle-occupied regions and explores trajectories with low time-consumption.

III. Environment Extraction with Graphics

This section clusters adjacent obstacle points as a large node $\mathcal{N}_i$ consisting of an obstacle group ζ_i, an ordered boundary B_i, and an ordered corner list C_i. The cluster replaces the traditional waypoints generation.

A. Problem Formulation

The robot state is assumed as $s_s = \{x_s, y_s, \theta_s, v_s, \omega_s\}$, and the local goal is $s_f = \{x_f, y_f, \theta_f, v_f, \omega_f\}$. The costmap



(a) The vector $\overrightarrow{c_1c_3}$ covers ζ_2 at c_2 (b) c_2 splits $\overrightarrow{c_1c_3}$ into $\overrightarrow{c_1c_2}$ and $\overrightarrow{c_2c_3}$ which covers ζ_2 at c_2' and ζ_3 at c_3' (c) c_2 jumps to c_2' and c_3 jumps to c_3' (d) c_2' and c_3 are optimized to be closer (e) Each obstacle group ζ_i has 2 surrounding paths (blue and purple)

Fig. 3. Illustration of topologically distinctive paths generation. (a-d) construct a graph with 3 nodes. It provides piecewise paths for vector-mode. (e) sets the start and the goal points, and 2^3 topologically distinctive paths are generated directly with the help of surround-mode.

$M \in \mathbb{R}^2$ is dilated by the robot radius R, and the map size is $\{W \times H\}$. For each point $p = \{(x,y)|_{x\in[0,W],y\in[0,H]}\} \in \mathbb{R}^2$ in the costmap, $M(p) = 1$ represents the point is obstacle-occupied while $M(p) = 0$ means collision-free. In this paper, we define each point as one of four states: $\{F, \zeta_k, B_k, C_k\}$. $F = \{p|_{M(p)==0}\}$ denotes the point is collision-free. $\zeta_k = \{p_i|_{M(p_i)==1;\exists p_j\in\zeta_j,||p_i-p_j||_\infty\leq 1}\}$ represents an obstacle group, which is formed by a series of adjacent obstacle-occupied points. When a point cannot be connected with any other obstacle-occupied point, a new group ζ_{k+1} is established. $B_k = \{p_i|_{p_i\in F;\exists p_j\in\zeta_j,\exists p_l\in F,||p_i-p_j||_\infty\leq 1,||p_i-p_l||_\infty\leq 1}\}$ represents the boundary of ζ_k. It briefly outlines ζ_k and is the limit that the robot can reach when approaching ζ_k. C_k represents the corners of ζ_k which simplifies the boundary B_k by reserving only a few points with most effective information.

B. Obstacle Group Extraction and Identification

We exploit the continuity of the costmap to explore obstacle groups. The exploration detects the obstacle points diagonally from the lower left of the map.

$$\{(x,y)|_{x+y=k,y^{++},k\in[0,W+H]}\} \tag{4}$$

For an obstacle point p, we detect its four adjacent points in turn as the mask in Fig.2(b)-(c) shows. If an adjacent point p' already belongs to an obstacle group ζ_k, the current point (and the obstacle group ζ_i to which it belongs) will be merged into ζ_k, otherwise it will open a new obstacle group.

$$\{p \in \zeta_k|_{p\notin\forall\zeta_i;\exists p'\in mask(p),p'\in\zeta_k}\} \tag{5}$$

$$\{\zeta_k = \zeta_k \cup \zeta_i|_{p\in\zeta_i;\exists p'\in mask(p),p'\in\zeta_k}\} \tag{6}$$

Besides, a map M_{obs} labeled by obstacle groups is generated, where $M_{obs}(p) = j$ if $p \in \zeta_j$, else $M_{obs}(p) = 0$.

C. Boundary Extraction and Sorting

Boundary extraction utilizes the closing operation in computer graphics. The operation consists of a dilation and a erosion. The dilation extends obstacle groups by one step, forming a new map with a safety margin:

$$\{M_{dil}(x,y) = j|_{\exists M(x+\Delta x,y+\Delta y)=j,j\neq 0,\Delta x,\Delta y\in\{-1,0.1\}}\} \tag{7}$$

Erosion shrinks M_{dil} and fills gaps compared with original M:

$$\{M_{ero}(x,y) = 0|_{\forall M_{dil}(x+\Delta x,y+\Delta y)=0,\Delta x,\Delta y\in\{-1,0.1\}}\} \tag{8}$$

By comparing M_{dil} and M_{ero}, a new map M_{boun} labeled by safety boundaries is generated:

$$\{M_{boun}(x,y) = M_{dil}(x,y)|_{M_{dil}(x,y)\oplus M_{ero}(x,y)==1}\} \tag{9}$$

We traverse the map in row-major order. Once a boundary point $p = \{(x,y)|_{M_{obs}(p)=0,M_{boun}=j}\}$ is found, it creates a new boundary group B_j, and its adjacent points $p' = \{(x',y')|_{p'\in mask(p),M_{boun}(p')=j}\}$ will be explored in turn as the mask in Fig.2(d) shows to iteratively extend the boundary group. Finally, a boundary group of waypoints arranged in a counter-clockwise direction is formed.

D. Corner Extraction and Resampling

The corner points in the picture contain more information than normal ones. Regarding the costmap as a picture, Harris corner identification [25] extracts corner points from boundaries further as Fig.2(e)-(f) shows. Each corner is assigned a response function R, which reflects how much the gradient around it varies. The original corner points are all obstacle points, and their response functions have different values in different directions. The original corner points are guided to move along this direction by one step, so that the new corner points are all boundary points. Due to uneven distribution of corner points, e.g. a huge rectangle may have only 4 corners that are far apart, while an obstacle with bumpy boundary may have many adjacent corners, there may be plenty of redundant corner points. So we resample the corner points with a dynamic window. We assume an obstacle boundary can be represented by n corners. n is constrained by a max threshold n_c and a min step size k_{step}. For a boundary with

$|B_j|$ points, a corner points combination C_j^k starting from the $k^{th}|_{,k\in[0,|B_j|]}$ point $B_j[k]$ has a likelihood value:

$$L(k) = \sum_{i\in[0,n)} B_j[k + (k_{step} * i)\%|B_j|] \tag{10}$$

The combination with the maximum likelihood value is the result of resampling, and the obstacle group can be represented by the corner points in this combination. Since the boundary points are ordered, the corner points are naturally ordered.

IV. Topologically Distinctive Paths Generation

A. Piecewise Path Formed by Surround and Vector Modes

Different from the traditional method of sampling waypoints in collision-free regions and searching for paths among waypoints, we treat each obstacle as a node, and a generalized path T is defined by the *visibility graph* algorithm [26] as the connection between nodes:

$$T = \{\overrightarrow{s_s c_1^{in}}, \overset{\frown}{\zeta_1(c_1^{in}, c_1^{out})}, \overrightarrow{c_1^{out} c_2^{in}}, ..., \overrightarrow{c_m^{out} s_f}\} \tag{11}$$

where, the first item is the connection between the robot and a visible corner point $c_1^{in} \in C_1$. The second item is a path (part of the boundary B_1) surrounding the obstacle group ζ_1. It starts from c_1^{in} and ends at c_1^{out} once another obstacle group ζ_2 is visible. ζ_1 and ζ_2 are connected by a vector from c_1^{out} to c_2^{in} as the third item. Then the surround-mode and the vector-mode are alternated until the goal s_f is visible for ζ_m at c_m^{out}.

For each surround-mode, utilizing the continuity of the boundary, we can directly obtain two different paths from c_i^{in} to c_i^{out}: clockwise or counter-clockwise. For each generalized path containing m obstacle groups, at most 2^m topologically distinctive paths can be extended directly. An illustration can be seen at Fig.3.

For vector-mode, the relationship between n obstacle groups form a graph $\mathcal{G}$ of $n \times n$. Each element $\mathcal{G}(i,j)|_{i\neq j}$ contains a vector $\overrightarrow{c_i^{out} c_j^{in}}$ where the two corner points can be connected without collision and should be as close as possible. The graph can be constructed as following rules: for two arbitrary obstacles ζ_i and ζ_j $(i \neq j)$, two arbitrary corner points c_i and c_j are selected, respectively. The relationship between the vector $\overrightarrow{c_i c_j}$ and the first obstacle ζ_k it intersects at b_k has three cases: (a) k==i or k==j means that c_i or c_j is on the back of ζ_i or ζ_j, respectively. To jump out of this local minima, c_i or c_j will be replaced with a corner point c_k which is closest to the intersected boundary point b_k. (b) k==$null$ means the two corner points can be connected directly without collision. They will be then iteratively optimized to shorten the distance. Assuming the indexes of c_i and c_j are ϵ_i in C_i and ϵ_j in C_j, respectively, the optimizing directions subject to:

$$dir_i^*, dir_j^* = \underset{dir_i, dir_j \in \{-1,0,1\}}{\arg\min} ||C_i[\epsilon_i + dir_i] - C_j[\epsilon_j + dir_j]||_2 \tag{12}$$

(c) Otherwise, the vector intersects with another obstacle ζ_k and this connection process is split into two new ones $\overrightarrow{c_i c_k}$ and $\overrightarrow{c_k c_j}$, where $c_k = \underset{c_k \in C_k}{\arg\min}(||c_k - b_k||_2)$.

B. Topologically Distinctive Paths Searching

To obtain all topologically distinctive paths, a depth-first search is used to search in the graph. For an environment with n obstacle groups, s_s can connect at most n ones at the first layer, and then $n-1$ left alternatives at the second layer, etc. The search is ended until the goal s_f is visible. For each path, we borrow from [14] to calculate its H-signature, and only the path with the shortest length in the same homology class will be reserved. In this case, all the topologically distinctive paths can be obtained completely in time complexity time $O(n!)$, while the original Voronoi-based method [12] is $O((\alpha n)!)$, where α is the average waypoints related to each single obstacle and it's usually a large number.

V. Trajectory Optimization and Selection

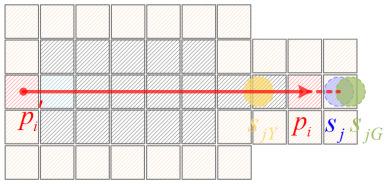


Fig. 4. Illustration of obstacle gradient. For a given waypoint s_j with the nearest obstacle point p_i, traditional TEB considers only undirected Euclidean distance, and two potential waypoints s_{jY} and s_{jG} have equivalent cost difference $s_{jY}p_i - s_jp_i = s_{jG}p_i - s_jp_i$. In contrast, the obstacle gradient $\overrightarrow{p_i' p_i}$ makes cost difference from s_j to s_{jG} negative $C_{s_{jG}}^{dis} - C_{s_j}^{dis} < 0$ and that from s_j to s_{jY} positive $C_{s_{jY}}^{dis} - C_{s_j}^{dis} > 0$. The safer s_{jG} is preferred.

A. Trajectory Optimization with Obstacle Gradient

Traditional TEB defines *distance to the closest obstacle* item in the subjective function as Euclidean distance. This understanding obstacles from local perspective could trap the trajectory inside obstacles as Fig.1(b-d) shows. Based on the obstacle extraction, this paper proposes an *obstacle gradient* to solve the problem. For a obstacle point $p_i = (x_i, y_i)$ belonging to ζ_i and a waypoint s_j on the initial path, the gradient of p_i is decided by another boundary point p_i' on the back of ζ_i:

$$\overrightarrow{g(p_i)} = \{\overrightarrow{p_i' p_i}|_{p_i' = s_j p_i \wedge B_i, max(s_j p_i')}\} \tag{13}$$

where, p_i' is the farthest intersection of a line $s_j p_i$ and the boundary B_i from s_j. The cost of s_j is defined as:

$$\mathcal{C}_{s_j}^{dis} = -\alpha \frac{\overrightarrow{g(p_i)}}{|\overrightarrow{g(p_i)}|} \cdot \overrightarrow{p_i s_j}, \ \alpha = \begin{cases} \alpha_1 & M(s_i) \neq 0 \\ \alpha_2 & otherwise \end{cases} \tag{14}$$

where, $\alpha_1 > \alpha_2 > 0$ are two constant parameters. The trajectory will be greatly penalized by a positive cost and a larger gain factor α_1 when it covers obstacles. Otherwise, the cost is negative with a smaller gain factor (just like traditional methods). An example of calculating obstacle gradient is shown in Fig.4.

B. Selection Criteria for the Best Trajectory

The objective function focuses on the quality of the trajectory itself, balancing various costs (e.g. safety, length, time) and optimizing the trajectory within the homology class. It is also used as a criterion by traditional approaches [8], [9] to select the best among the optimized topologically distinctive trajectories. We argue the criteria is unreasonable, e.g. a trajectory passing through a human group may reach the goal quickly, but it will disturb humans, and the sensor noise and human motion uncertainty could increase the risk of collision, especially in complex dynamic environments. Another alternative that detours the group could be safer and have also a competitive time cost. To realize this preference, a high-level cost item named *uncertainty* is introduced to score safety.

Since the optimized trajectory will intersect with the piecewise paths generated by vector modes, the piecewise paths are great tools to reflect the uncertainty and tendency of the obstacle movement, and further to judge the safety of the trajectory. We assume each dynamic obstacle group ζ_i has a central position p_i and an orientation θ_i. Considering the vector-mode $\overrightarrow{c_i c_j}$ between two arbitrary obstacle groups ζ_i and ζ_j, the cost of a point on the piecewise path $p_{ij} \in \overrightarrow{c_i c_j}$, is projected by two obstacle groups:

$$\mathcal{C}_{ij}^{safe}(p_{ij}) = f(p_{ij} - \zeta_i) + f(p_{ij} - \zeta_j) \tag{15}$$

Define the direction of p_{ij} w.r.t. the orientation of the obstacle as θ and the distance from p_{ij} to the obstacle border as ρ:

$$f(\rho, \theta) = \begin{cases} A\ exp(-\frac{\rho^2 cos^2\theta}{2\sigma_f^2} - \frac{\rho^2 sin^2\theta}{2\sigma_{lr}^2}) & \theta \in [-\pi/2, \pi/2] \\ A\ exp(-\frac{\rho^2 cos^2\theta}{2\sigma_b^2} - \frac{\rho^2 sin^2\theta}{2\sigma_{lr}^2}) & otherwise \end{cases} \tag{16}$$

where, A is a constant parameter. $\sigma_f, \sigma_b, \sigma_{lr}$ are proportional to the obstacle velocity v_i with $\alpha_f > \alpha_b = \alpha_{lr}$. f models the uncertainty and tendency of obstacles using a 2D asymmetric Gaussian distribution as obstacles tend to maintain velocity and have uncertainty to change directions.

Kalman Filter predicts the future positions and velocities of obstacles. For each timestamped position $s = (x, y, t)$ on the trajectory, piecewise paths between every two obstacles at t are assigned. If the trajectory passes through a piecewise path $\overrightarrow{c_i c_j}(t)$, $\mathcal{C}_{ij}^{safe}(s)$ is assigned, otherwise, the cost is 0. The total cost consist of three parts: time, length, and uncertainty.

$$\mathcal{C}^{total}(\gamma) = w_1 \sum_{k=0}^{n-1} \Delta T_k + w_2 \sum_{k=0}^{n-1} \Delta s_k + w_3 \max_{s \in \gamma} \mathcal{C}_{ij}^{safe}(s) \tag{17}$$

The workflow of our method GraphicTEB is shown in Algorithm 1. Lines 3-6 use Sec.III to analyze the environment and extract nodes for the graph. Lines 7-10 use Sec.IV to build the graph and generate topologically distinctive paths by searching in the graph. Lines 11-15 and 16-18 use Sec.V to optimize the trajectories and select the optimal trajectory for the robot to execute, respectively.

Algorithm 1 Workflow of GraphicTEB

1: **Input:** robot s_s, global path P_g, global costmap M_g
2: **Return:** local trajectory γ^*, velocity command (v, w)
3: Extract local costmap $M(M_g)$
4: Extract obstacle groups $\{\zeta_1, ..., \zeta_{n_1}\}(M)$
5: **for all** $\zeta_i \in \{\zeta_1, ..., \zeta_{n_1}\}$ **do**
6: extract and sort boundary points $B_i(\zeta_i)$
7: extract corner points $C_i(B_i)$
8: Pick local goal $s_f(P_g, M)$
9: Build graph $\mathcal{G}(s_s, s_f, \{(B_1, C_1), ..., (B_n, C_{n_1})\})$
10: Depth-first Search for generalized paths $\{T_1, ..., T_{n_2}\}(\mathcal{G})$
11: Derive normal paths $\{\gamma_1, ..., \gamma_{n_3}\}(T)$
12: **for** $k = 0$ **and** $k++ < iterionMax$ **do**
13: **for all** $\gamma_i \in \{\gamma_1, ..., \gamma_{n_3}\}$ **do**
14: **for all** $x_j \in \gamma_i$ **do**
15: find closest obstacle p_j and gradient $\overrightarrow{g(p_j)}$
16: optimize $\gamma_i(\{s_1, ...s_{n_4-1}\}, \{\overrightarrow{g(p_1)}, .., \overrightarrow{g(p_{n_4-1})}\})$
17: **for all** $\zeta_i \in \{\zeta_1, ..., \zeta_{n_1}\}$ **do**
18: assign asymmetric Gauss value for edges
19: Select the best trajectory γ^* and related velocity (v, w)

VI. Simulation

The following scenarios demonstrate the performance of the proposed method GraphicTEB on completeness, computing efficiency, optimization, and safety. Fig.5 and Fig.6 analyze GraphicTEB in static environment with qualitative comparisons and quantitative comparisons, respectively, while Fig.7 and Fig.8 show its performance in dynamic environments with qualitative comparisons and quantitative comparisons, respectively.

A. Simulation Setup

The robot in the simulation and experiment is a two-wheel differential robot[1]. The radius of the robot is 0.25 m, the maximum speed is 0.5 m/s, the maximum acceleration is 0.5 m/s^2, the additional safety margin is 0, and the map resolution is 0.1. The navigation framework is extended from an open source work[2], in which the global path planning is the route guidance in the long corridor and A* in the broad room, respectively. Karto is used for SLAM, while AMCL is used for robot localization. Algorithms are executed on a machine running Ubuntu 18.04 with a 5.0 Ghz Intel i7 CPU. Some important parameters are listed here. $w_1 = 1000$, $w_2 = 100$, $w_3 = 2$ and $w_4 = 1$ in Equ.3, $k_{step} = 3$ in Equ.10, $\alpha_1 = 3$ and $\alpha_2 = 1$ in Equ.14, $\sigma_f = 0.5 + 1.0v_s$, $\sigma_b = 0.05 + 0.2v_s$, $\sigma_{lr} = 0.05 + 0.2v_s$, and $A = 2.0$ in Equ.16, $w_1 = 2$, $w_2 = 1$, and $w_3 = 4$ in Equ.17. For more details, please visit the c++ source code[3].

The simulation and experiment are tested in two scenarios of the long corridor and the broad room [2], [27], [28], [29], [30]. The former is defined as a corridor with a long length and a small width. The space constraints lead to a

[1]https://www.robotrunner.com
[2]https://github.com/ros-planning/navigation
[3]https://github.com/Chris-Arvin/GraphicTEB

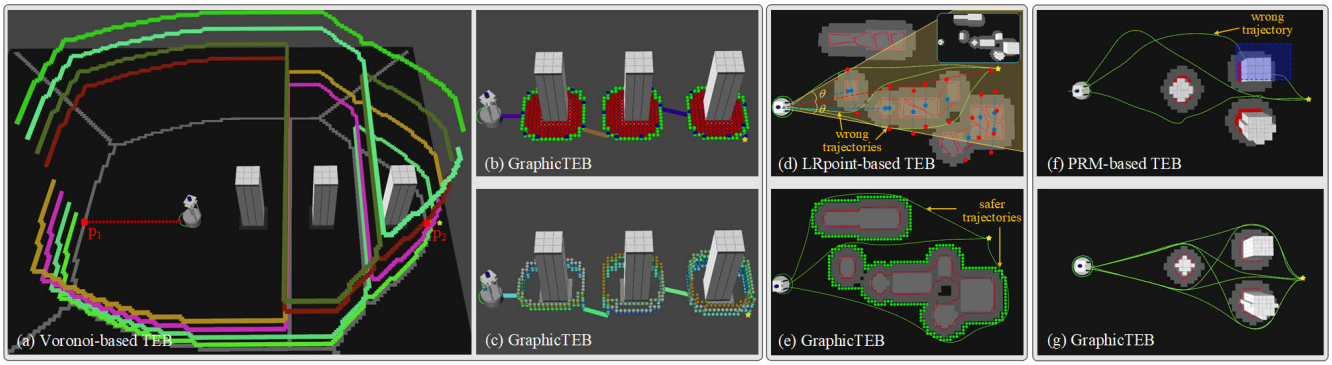


Fig. 5. Qualitative comparison of LRpoint-based TEB, Voronoi-based TEB, PRM-based TEB, and GraphicTEB (ours) in static environments. The three scenarios demonstrate the advantage of GraphicTEB on obstacle extraction, topologically distinctive paths searching, and trajectory optimization, respectively.

small feasible space for the robot, and the robot often stops overly conservatively. This scenario challenges the motion efficiency of the robot. The latter is defined as a broad area with static obstacles and dynamic pedestrians. The human-robot interaction leads to a small feasible space for the robot, and the robot needs to frequently decide how to pass people. This scenario challenges the motion safety of the robot.

B. Static Scenarios

Three methods of Voronoi-based TEB, LRpoint-based TEB and PRM-based TEB are compared. Voronoi-base TEB can find all topologically distinctive paths, but has low computing efficiency. The latter two find all paths probabilistically, but take short time. GraphicTEB is compared to the strengths of the three methods respectively.

Graphics & Computing Efficiency: Fig.5(a-c) compare the computing efficiency of Voronoi-based TEB [8], [12] and GraphicTEB. The robot and the goal (yellow pentagram) are located on opposite sides of the map, intervening with three rectangular obstacles.

Voronoi-based TEB first constructs a Voronoi diagram to extract all voronoi points shown in grey in Fig.5(a). Each voronoi point has equivalent distance to its nearest two obstacle points, and the voronoi points are adjacent. The nearest voronoi point to the robot is p_1 and the nearest point to the goal is p_2. All topologically distinctive paths from p_1 to p_2 are founded by Depth-First search. Plus the piecewise paths from the robot to p_1 and the goal to p_2, the final paths are generated.

The issue is that the number of voronoi points are too many (up to 5742 in the example). As the Depth-First search has a time complexity $O(n!)$, voronoi-based TEB always cannot work in real-time when the environment is complex.

In contrast, GraphicTEB clusters each obstacle as a node, including an obstacle group (red), an ordered boundary (green), and an ordered corner (blue), as shown in Fig.5(b). The Depth-First search works on a few nodes instead of numerous waypoints. Once a generalized path consisting of n nodes is found, at most 2^n normal paths consisting of waypoints can be directly derived without consuming additional computing resources. Fig.5(b) illustrates $n = 3$ nodes and $2^3 = 8$ paths indicated with distinct colors. While both approaches find all the 8 paths, Voronoi-based TEB takes 874 ms and GrapahicTEB takes 1.1 ms.

Graphics & Completeness: Fig.5(d-e) compare the capacity of the environment extraction of LRpoint-based TEB [8] and GraphicTEB. The environment shown at the up-right of Fig.5(d) contains several separated obstacles. Expanding the obstacles according to the radius of the robot, they are actually jointed into two new ones.

LRpoint-based TEB clusters obstacles by [31] which clusters adjacent obstacle points on the original map as large polygons. To reduce the error, the max size of the polygon is limited. Each polygon is outlined in red in Fig.5(d), and its center is labeled as a blue circle. LRpoint-based TEB defines the goal heading (red dotted vector) and the obstacle heading (blue dotted vector) as the directions from the robot to the goal and the obstacle center, respectively. When an obstacle heading is less than $\theta = \pi/9$ with respect to the goal heading, the left and right waypoints (blue circles) perpendicular to the obstacle headings are sampled around the obstacles. Then topologically distinctive trajectories (green trajectories) are searched among the waypoints by depth-first search.

This raises two issues. The first is that the clustering must indicate the maximum size of the obstacle in advance, resulting in clustered obstacles are separated and smaller than the actual obstacles. Some waypoints that should have been deleted because their distances to the obstacles are less than the safety threshold will be reserved, invalidating the path consisting of them, as indicated by *wrong trajectory* in Fig.5(d). The second is that some topologically distinctive trajectories, such as the *safer trajectories* in Fig.5(e), will be overlooked since the sampling is limited near the obstacle headings.

In contrast, the cluster of GraphicTEB is not limited by a max size. It works on the map expanded by robot radius and merges all adjacent obstacle points into obstacle groups with green outlines. The expansion avoids wrong trajectories. Because the closing operation of graphics fills in the gaps, even after we shrink the obstacle groups by the robot radius,

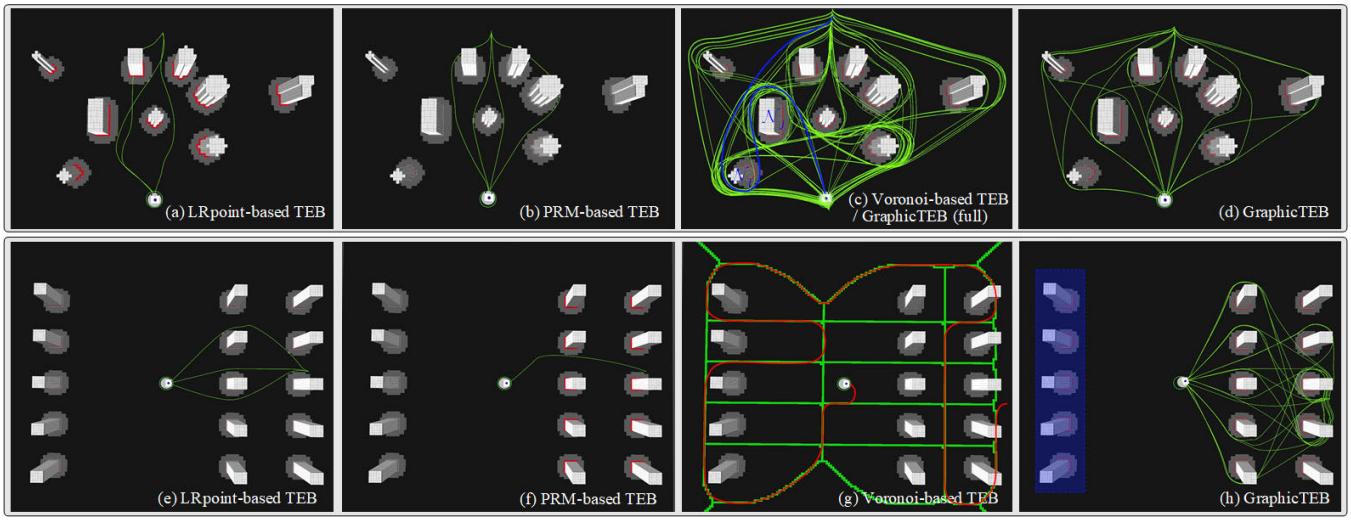


Fig. 6. Two examples of quantitative comparisons of GraphicTEB, GraphicTEB(full), Voronoi-based TEB, LRpoint-based TEB, and PRM-based TEB in static environments.

as indicated by the red outlines in Fig.5(b), they are still adjacent enough. Additionally, along the borders of obstacle nodes, all the three safe paths are figured out directly.

Obstacle Gradient & Optimization: Fig.5(f-g) compares the trajectory optimization of PRM-based TEB [8] and GraphicTEB. Three obstacles are distributed in a triangle, and the robot and the goal are located on opposite sides of the map.

PRM-based TEB generates a sampleable map constrained by the robot and target locations. A preset number of waypoints are sampled in the map, and the Depth-First search finds paths among waypoints. Like LRpoint-based TEB, PRM-based TEB is probabilistically complete and can finish search quickly because of the small number of waypoints.

The issue in Fig.1(b-d) occurs in the blue region of Fig.5(f). Two waypoints are sampled on upper and right sides of the obstacle. They are wrongly optimized in obstacles in a certain optimization iteration because PRM-based TEB only considers the undirected Euclidean distance between a waypoint and its nearest obstacle point. In subsequent iterations, the trajectory is trapped in obstacles up to the max iteration.

In contrast, GraphicTEB considers the directed distance as illustrated in Fig.4. It penalizes the collision more strictly and guides the trajectory out of the trap by obstacle gradient.

Quantitative Comparisons with Random Obstacles: Quantitative comparisons in static environments are performed in broad rooms with randomly distributed obstacles, random robot initializations, and random robot goals. The map size is 15 m $\times$ 15 m and the resolution is 0.1 m. Three kinds of obstacles, circular, rectangular, and T-shaped, are randomly distributed on the map. The radius of the circular obstacle is randomly selected in the range of [0.1 m, 1.5 m], the side length of the rectangle is randomly selected in the range of [0.1 m, 1.5 m], and the T-shaped obstacle is composed of two intersecting rectangles. The number of obstacles in each test is randomly selected in the range of [5, 15]. The robot and the goal are randomly located in non-obstacle areas and their distance is required to be greater than 15 m. According to the above-mentioned rules, 100 test environments are randomly generated to compare five algorithms: GraphicTEB, GraphicTEB (full), Voronoi-based TEB, LRpoint-based TEB, and PRM-based TEB. The approaches given in this paper serve as the foundation for GraphicTEB and GraphicTEB(full). The difference is that GraphicTEB is more practical by adding a limitation during the graph searching: The node $\mathcal{N}_j$ will not extend its adjacent node $\mathcal{N}_i$ if $\mathcal{N}_i$ also can be extended by the father nodes of $\mathcal{N}_j$. GraphicTEB(full), on the other hand, has no such limitation. Other algorithms utilize open-source codes and default parameters[4]. In all algorithms, [14] assigns the path a unique H-signature.

Table.I depicts the results of the quantitative experiments using four measures. (a) path coverage: the ratio of paths found by an algorithm to those found by Voronoi-based TEB. GraphicTEB(full) achieves the same performance as Voronoi-based TEB in terms of completeness. GraphicTEB with limitations is not complete, but the limitation is justified because the deleted paths are either too long or involve unnecessary detours. This is another benefit of the paper's proposition. It interpretably prunes paths rather than just prunes waypoints by the goal heading like LRpoint-based TEB and PRM-based TEB, which cannot find a solution when the robot has to go backward and detour in Back scenario (detailed in Fig.7). (b) CPU time for total paths: the time required to sample waypoints and search among them for all topologically distinctive paths. GraphicTEB (full) is about 90 times quicker than Voronoi-based TEB in terms of computational efficiency, whereas GraphicTEB with limitations reaches competitive performance compared with the fastest LRpoint-based TEB. (c) CPU time for per path: the average time required to find a path throughout the

[4]https://github.com/rst-tu-dortmund/teb_local_planner

TABLE I
RESULT OF QUANTITATIVE COMPARISONS IN 100 STATIC ENVIRONMENTS

	Path coverage [%]	CPU time for total paths [ms]	CPU time for per path [ms]	Valid rate [%]
LRpoint-based TEB	3.9	**5.9 ± 17.5**	2.9	79
PRM-based TEB	9.7	70.2 ± 88.2	14.1	82
Voronoi-based TEB	**100**	6410.0 ± 18332.2	120.5	92
GraphicTEB(full)*	**100**	72.0 ± 184.4	**1.7**	**100**
GraphicTEB*	35.4	**7.5 ± 23.0**	**0.9**	**100**

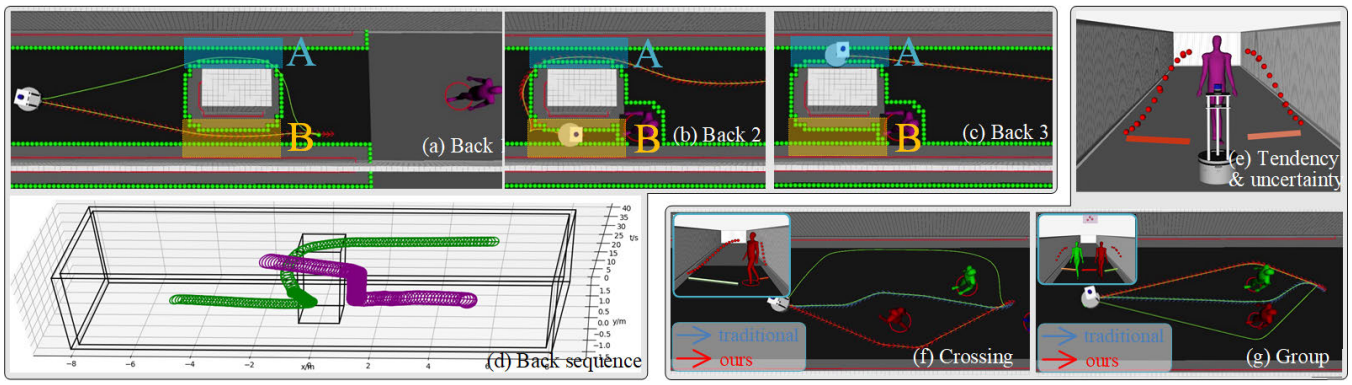


Fig. 7. Qualitative experiment and analysis of GraphicTEB in dynamic environments. (a-d) illustrates the process of the robot to back off and pass the obstacle by exploring alternative trajectories in a *back* scenario. (e) shows how the cost item *uncertainty* works. (f) and (g) compare the best trajectories under different evaluation strategies in *crossing* and *group* scenarios, respectively.

whole process. The item intuitively reflects the advantages of Graphic-based algorithms, that is, the average time to find each path is the shortest among all approaches. (d) valid rate: the proportion of valid paths compared to total paths. When an optimal trajectory crosses over an obstacle, the trajectory is invalid. Graphic-based algorithms avoid collision with 100% accuracy in terms of optimization performance (valid rate). Because the initial waypoints are far away from obstacles, the Voronoi-based TEB reaches 92%. The remainders have lower valid rates.

Fig.6 illustrates two challenging examples of quantitative experiments. (a-d) show an environment with 9 obstacles. (a) LRpoint-based TEB and (b) PRM-based TEB find 3 trajectories in 3.0 ms and 4 ones in 25.9 ms, respectively. (c) Voronoi-based TEB finds 48 trajectories in 6220.0 ms, while GraphicTEB(full) finds the same 48 ones in 44.2 ms. The completeness of GraphicTEB(full) is guaranteed compared to LRpoint-based and PRM-based TEB and it takes very little time for GraphicTEB(full) to search all paths compared to Voronoi-based TEB. (d) GraphicTEB finds 14 trajectories in 3.4 ms. The limitation technique of GraphicTEB prevents many meaningless detours as shown in Fig.6(b) with blue trajectory. GraphicTEB finds numerous meaningful topologically distinctive paths compared to Voronoi-based TEB, and it reaches competitive computing time compared to LRpoint-based and PRM-based TEB. (e-h) show another environment with 15 obstacles. (e) LRpoint-based TEB and (f) PRM-based TEB find 3 trajectories in 9.5 ms and 1 trajectory in 29.4 ms, respectively. (g) Voronoi-based TEB finds 562 trajectories in 10220.4 ms while GraphicTEB(full) is 256.4 ms. The waypoints generated by the Voronoi diagram are shown in green, and a terrible trajectory with numerous meaningless detours is shown in red. This trajectory will not be chosen by the robot to execute under any circumstances, but consumes computing resources. (h) GraphicTEB quickly finds 20 trajectories in 9.7 ms. It intelligently ignores meaningless paths associated with obstacles behind the robot (blue area) as they can directly connect with the robot.

C. Dynamic Scenarios

Fig.7 illustrates three qualitative experiments of robots avoiding pedestrians in corridors. This is the daily working scene of hotel service robots. Traditional global path planning methods (such as A*) are redundant and meaningless in this case, since the destination is far away from the robot, and only dynamic pedestrians near the robot can be detected. In contrast, the global path planning in the corridor scenario is usually the centerline of the road, and the local target of the robot is dynamically set as a point at a certain distance in front of the robot on the centerline. Local motion planning collects real-time sensor data and determines a trajectory to avoid obstacles. Since narrow corridors place higher requirements on the robot's avoidance ability, it is important to consider safer trajectory selection strategies in dense crowds under the uncertainty of pedestrian motion. This subsection specifically evaluates the ability of GraphicTEB to select safe trajectories.

Since PRM-based TEB is superior to LRpoint-based TEB in terms of computing efficiency and completeness, as shown in Table.I, and Voronoi-based TEB cannot perform real-time calculations in pedestrian-dense corridors, so only PRM-based TEB and GraphicTEB are compared in this subsection.

Graphics & Completeness: Fig.7(a-d) shows a *back* scenario in which the robot must move backward and select

another alternative. As (a) shows, a static obstacle is placed in the middle of the corridor, forming two narrow gaps $\mathcal{A}$ and $\mathcal{B}$ on both sides, through which the robot must select one to pass. Two green topologically distinctive trajectories are discovered as the robot approaches the obstacle. The robot follows the red trajectory through $\mathcal{B}$ in accordance with the society norm of walking on the right (in China). At this moment, we manually control the pedestrian to purposely breach social rules and walk to block $\mathcal{B}$ as (b) shows.

In this situation, because of the obstacle heading pruning technique of LRpoint-based TEB and probabilistic samplings of PRM-based TEB, they neither can find an alternative trajectory to make the robot back off and then head for $\mathcal{A}$ to solve the dead-lock situation as (c) shows. In contrast, the completeness of GraphicTEB makes alternatives discoverable, and this feature is also the basis of safe trajectory selection strategies. The whole process of GraphicTEB is shown in (d), where the green circles are robot trajectories, and the purple circles are pedestrian trajectories.

Evaluation Strategy & Safety: Fig.7(e) shows the cost item *uncertainty*. The walls on both sides form two piecewise paths with the pedestrian, respectively. The static walls have no uncertainty, while the pedestrian assigns the piecewise paths using Equ.16. Discrete cost values are illustrated in red, and the magnitude of the value is proportional to its height. In this situation, passing from the left or right of the pedestrian has an nearly equivalent *uncertainty* cost.

Fig.7(f) shows a *crossing* scenario where the red pedestrian moves laterally such that the piecewise path in front of him is more costly and the piecewise path behind him is less costly. There are three potential trajectories (green, blue, red) for the robot to choose from. Traditional trajectory selection strategies prefer the blue trajectory since it is shorter and takes less time to reach the goal. However, since traditional optimization of TEB focuses on minimizing time cost, the optimal trajectories are always next to pedestrians in temporal-spatial space. The blue trajectory causes the robot to pass closely in front of the pedestrian, which is particularly risky in the presence of sensor noise. In contrast, GraphicTEB, directs the robot to pass behind the pedestrian as the red trajectory shows. Even with sensor noise, the robot can stop at any time without interfering with human movement or causing an accident. The robot has a greater safety margin.

Fig.7(g) shows a *group* scenario where the gap between two pedestrians is just narrow enough for the robot to pass. There are three potential trajectories (green, blue, red) for the robot to choose from. Traditional trajectory selection strategies allow the robot to pass between pedestrians by the blue trajectory. However, since the robot requires time to calculate a velocity command and continuously executes the previous command during the calculating interval, the robot cannot perfectly track the planned trajectory, and there is a high risk of colliding with pedestrians. The new item *uncertainty* solves this issue by raising the cost between the people. Both dynamic pedestrians project Gauss value on the piecewise path between them, resulting in a higher cost there. Eventually, the robot directed by GraphicTEB chooses to pass by the group's side by the red trajectory.

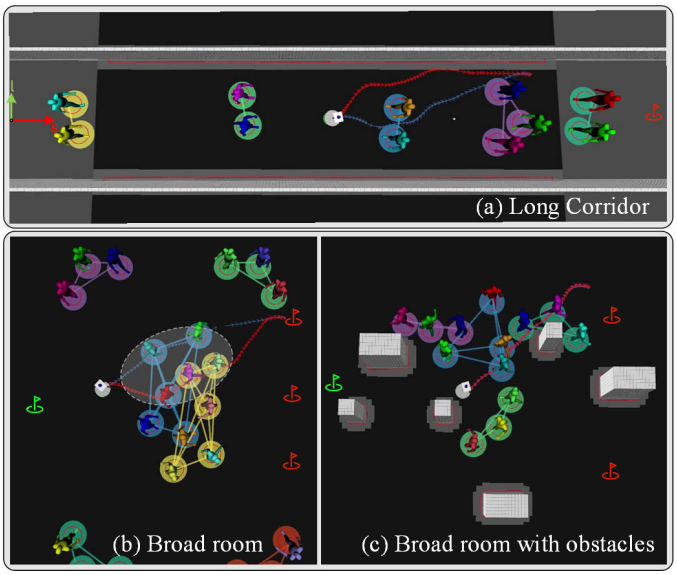


Fig. 8. Examples of quantitative experiments. The red and blue trajectories are best trajectories under proposed criteria and traditional criteria, respectively.

Quantitative Comparisons with SFM-driven Pedestrians: Quantitative comparisons in dynamic environments are performed in corridors and broad rooms with randomly distributed obstacles, random robot initializations and goals, and random pedestrian initializations and goals. A social force model (SFM) [32] is employed to drive pedestrians to simulate human reactions. SFM drives pedestrians with four forces: the attractive force of the destination, the repulsive force of static obstacles, the attractive force of pedestrians in the same group, and the repulsive force of other moving objects. In order to rigorously measure the robot's obstacle avoidance ability, the robot's repulsion force against pedestrians is omitted. Each of these three scenarios is tested 50 times. The test is successful if the robot reaches the target without collision.

Fig.8 illustrates three examples of quantitative experiments in dynamic scenarios. The corridor in Fig.8(a) is infinite on the horizontal axis s and 4.0 m wide on the lateral axis l. The radius of the pedestrian is 0.3 m. People with the same colored circles under their feet are members of the same group. The robot is not expected to move through the human group as it would disturb humans. The initial location of the robot is $(s = 0, l = 0)$ and the target is $(15, 0)$. Five human groups are randomly distributed in the corridor, with two groups behind the robot and three groups in front of the robot. The initial central positions of these groups are $(s_{rand}, 0)$, where s_{rand} is randomly generated in the range of $[-10, 10]$. The number of pedestrians in each group is randomly selected in the range of $[1, 3]$, and the positions of the pedestrians are randomly deviated from the center of the group by $(ds \in [-2, 2], dl \in [-1, 1])$. In traditional TEB, the same criteria are considered for trajectory optimization and trajectory selection, including the

TABLE II
QUANTITATIVE COMPARISONS IN DYNAMIC ENVIRONMENTS

	Corridor	Broad room	Broad room with obstacles
PRM-based TEB	94%	92%	92%
GraphicTEB	76%	74%	60%

shortest time, shortest distance, and safe separation from obstacles. As mentioned above, since only the distance of the nearest obstacle point is considered from a local perspective, the interaction relationship of pedestrians and the uncertainty of human motion are ignored, resulting in the blue trajectory being preferred. In contrast, the selection strategy in GraphicTEB especially measures the tendency and uncertainty of human motion through a Gaussian model. It prefers the red trajectory which is safer and also has a competitive time like the blue trajectory.

Fig.8(b) depicts an open environment with moving pedestrians. Six human groups are located at the upper left, middle left, lower left, upper right, middle right, and lower right of the map. Their targets are the initial sequence of the groups in reverse order. The pedestrians are still driven by SFM. As a result, complicated interactions between pedestrians appear at the map's center to verify the robot's obstacle avoidance ability. The robot starts from the green position and randomly chooses a target from the three red positions. A* provides the global path and the local target for the robot. The traditional trajectory selection strategy ignores the motion tendency and uncertainty of the five pedestrians enclosed within the white ellipse. In this situation, the blue trajectory is preferred by the traditional strategy. As the purple and cyan pedestrians keep going forward, and the blue pedestrian moves near the blue trajectory, the blue trajectory is dangerous, and the robot could frequently change decisions and oscillate in the future. In contrast, our trajectory selection strategy prefers the safer red trajectory from the beginning. As time progress, the purple, cyan, and red pedestrians move away from the red trajectory, allowing the robot to smoothly and safely move to the target.

Fig.8(c) randomly adds static obstacles to (b), resulting in a more complex environment. Three human groups are distributed at the upper left, upper right, and lower right of the map, and one group hovers at the center. The robot also starts from the green position and selects the target at random from the red positions. The pedestrians' driven model and the robot's global planning are the same as (b). Since there are multiple obstacles and pedestrians and it is costly to go around all of them, the robot must move through the crowd. It's dangerous and rigorously tests the trajectory selection strategies. As shown in Table.II, the success rates of GraphicTEB has achieved better results in all the three scenarios. The difference appears especially in the scenario of broad room with obstacles. The reason is that GraphicTEB can find all the topologically distinctive trajectories and our evaluation strategy can choose the safer trajectories, while PRM-based TEB may ignore these safe trajectories, and the robot has to take dangerous trajectories.

VII. EXPERIMENT

A. Experiment Setup

The experiment focuses on two complex scenarios, *long corridor* and *broad room*, including static obstacles, dynamic pedestrians, and a hidden obstacle that is not placed during pre-SLAM. The robot and algorithm parameters are the same as the simulation.

To reduce the error of pedestrian detection and tracking in the experiment, static obstacles and dynamic pedestrians are set to different heights. All the static obstacles are either circular with a height of 0.34 m or cube boxes with a height of 0.6 m, which are detected by a single-line laser at the bottom of the robot. All the dynamic obstacles are people with a height greater than 1.5 m, which are detected by the 16-line laser on the top of the robot. The human is detected and tracked by an improved open source work Hdl_People_Tracking[5] [33], [34], [35]: To realize pedestrian detection, Euclidean clustering and DPmeans are used to cluster raw gross point cloud data as various circles, and the AdaBoost classifier is used to judge the confidence that the circle is human. To realize pedestrian tracking, Kalman Filter predicts the positions of pedestrians at the next moment, Global Nearest Neighbor calculates Mahalanobis distances between predicted positions and observed positions, and Interacting Multiple Model matches pedestrian IDs between adjacent moments. The velocity v_i of any pedestrian is obtained by calculating the position difference between adjacent moments. A person out of the robot's field of vision is assumed to be moving at a constant speed within the following two seconds.

The pedestrian detection and tracking module runs at 5-10 Hz. Pedestrian speeds are limited to less than 0.5 m/s to weaken the effect of the pedestrian's position lagging behind the real position. Kalman Filter maintains pedestrian movement to reduce the impact of missed human detection, and the number of pedestrians in the experiment is limited to a maximum of two.

B. Long Corridor Scenario

Fig.9(a) shows the long corridor scenario [36], [37]. The size of the environment is 15 m $\times$ 2.5 m. The long sides of the scenario are outlined by parallel boards to simulate a corridor. As shown in Fig.9(b), three critical tests are indicated in blue, green, and orange, respectively. The blue area detailed in Fig.9(c) investigates the algorithms' computing efficiency. The two circular obstacles on both sides of the robot are very close, leaving the robot with only a 0.15 m safety margin from each obstacle. Because the robot will continue to execute the previous command until the current planning is finished, a longer time-consuming of the motion planning results in a higher risk of collision. Since Voronoi-based TEB requires online calculation of Voronoi diagrams, deep search paths in Voronoi points, optimization of all paths, and

[5]https://github.com/koide3/hdl_people_tracking

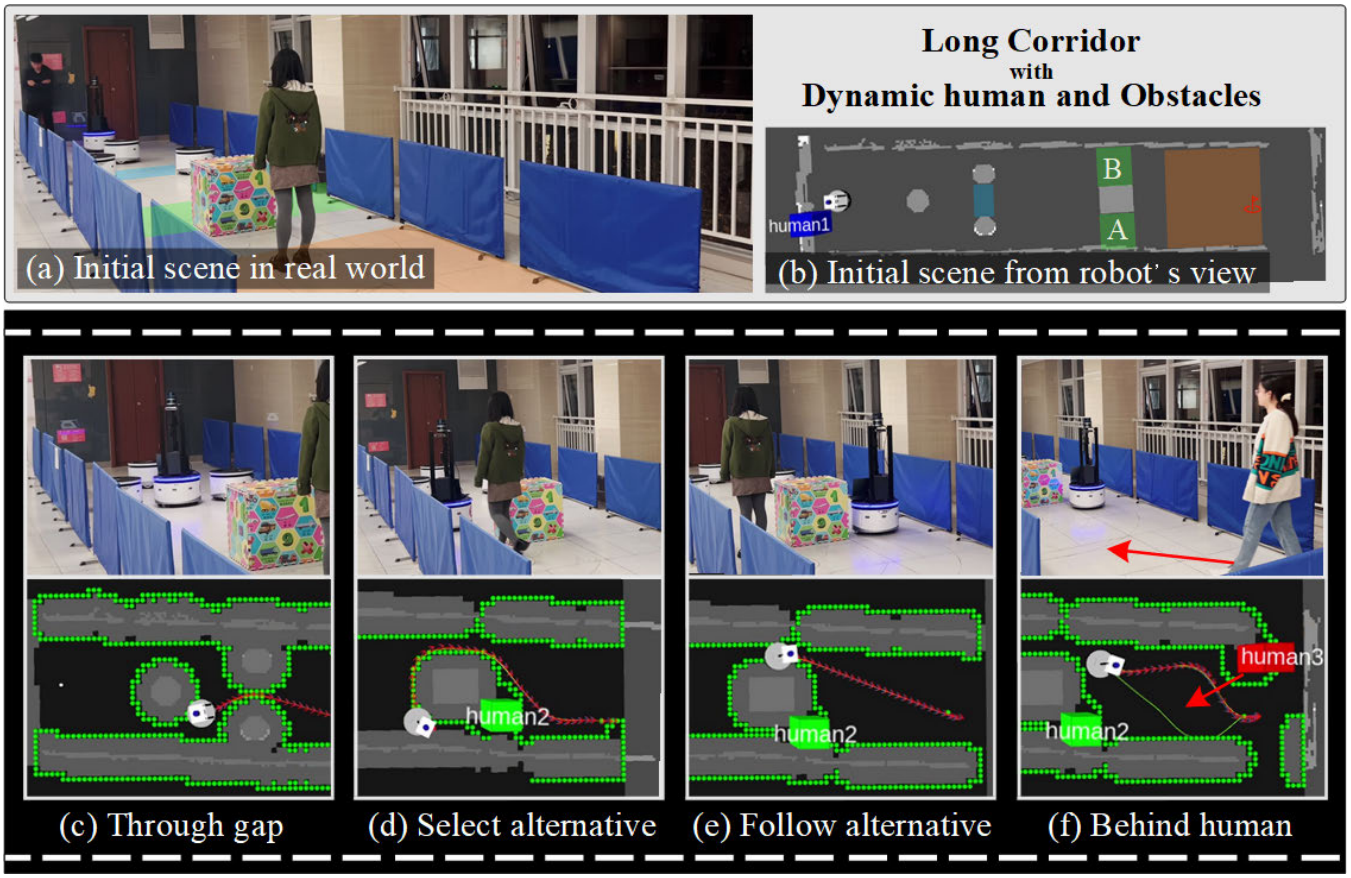


Fig. 9. Performance of GraphicTEB in long corridor scenario. (a-b) highlight three key tests, (c-f) show the movement of the robot at different times. The robot first passes through the middle of the two obstacles at (c), then backs up to give way to the person at (d-e), and finally passes behind the person at (f). For more details, visit: https://youtu.be/SzZGKdbzH9Q.

selection of trajectories, a lot of time is consumed during each planning period. In 10 replicate tests, the average planning time of Voronoi-based TEB is 0.34 s, and the running speed of the robot is kept at about 0.5m/s. At this planning time and speed, the robot can easily hit obstacles on both sides, and it collides in 5/10 tests. In contrast, the average planning time of GraphicTEB is 0.11 s, and the robot in 10/10 tests.

The green region verifies completeness, and Fig.9(d-e) depicts the decision-making process of the robot. An obstacle in the middle of the corridor gives the robot two options, $\mathcal{A}$ and $\mathcal{B}$. A person initially stands behind the obstacle, and both $\mathcal{A}$ and $\mathcal{B}$ allow the robot to pass without collision. The robot walks towards A in accordance with the (Chinese) social norm of walking on the right. However, when the robot enters $\mathcal{A}$, the pedestrian walks deliberately towards $\mathcal{A}$, blocking the gap. The robot is trapped since PRM-based TEB and LRpoint-based TEB cannot find an alternative trajectory. In contrast, graphic TEB can direct the robot to switch to $\mathcal{B}$.

The orange region investigates the trajectory selection criteria. When crossing the pedestrian with white sweater, GraphicTEB predicts her motion tendency with Kalman Filter and considers the collision risk with Gaussian-based selection criteria, as shown in Fig.9(f). Instead of passing in front of the person by the green trajectory, the robot ends up passing behind her by the red trajectory for higher safety.

C. Broad Room Scenario

Fig.10(a) shows the broad room scenario [2], [38]. The environment is 12 m $\times$ 7 m in size and is outlined by several boards. Only circular obstacles and blue boards are present when mapping the environment with slam in advance, as shown in Fig.10(a). Ideally, the robot should first choose the red local trajectory from various green topologically distinctive trajectories, as illustrated in (c), and the robot should finally approach the global target according to red trajectory in (a).

However, when a person (green) and a hidden obstacle (orange) occur, as shown in Fig.10(b), two difficulties arise and affect the robot's decision-makings. Fig.10(d) focuses on the movement of the pedestrian increases the uncertainty and collision risk of the green zone area in (c). In this case, different from Fig.10(c), the evaluation strategy of GraphicTEB enables the robot to move against the wall, ensuring safety through this small detour. Fig.10(e-g) focus on the replanning ability of the robot when encountering a hidden obstacle. When the robot is about to reach the goal along the shortest trajectory, a hidden obstacle is detected and blocks the way. The completeness of GraphicTEB allows the robot to find an alternative trajectory (red trajectory), and then take a great detour to reach the goal.

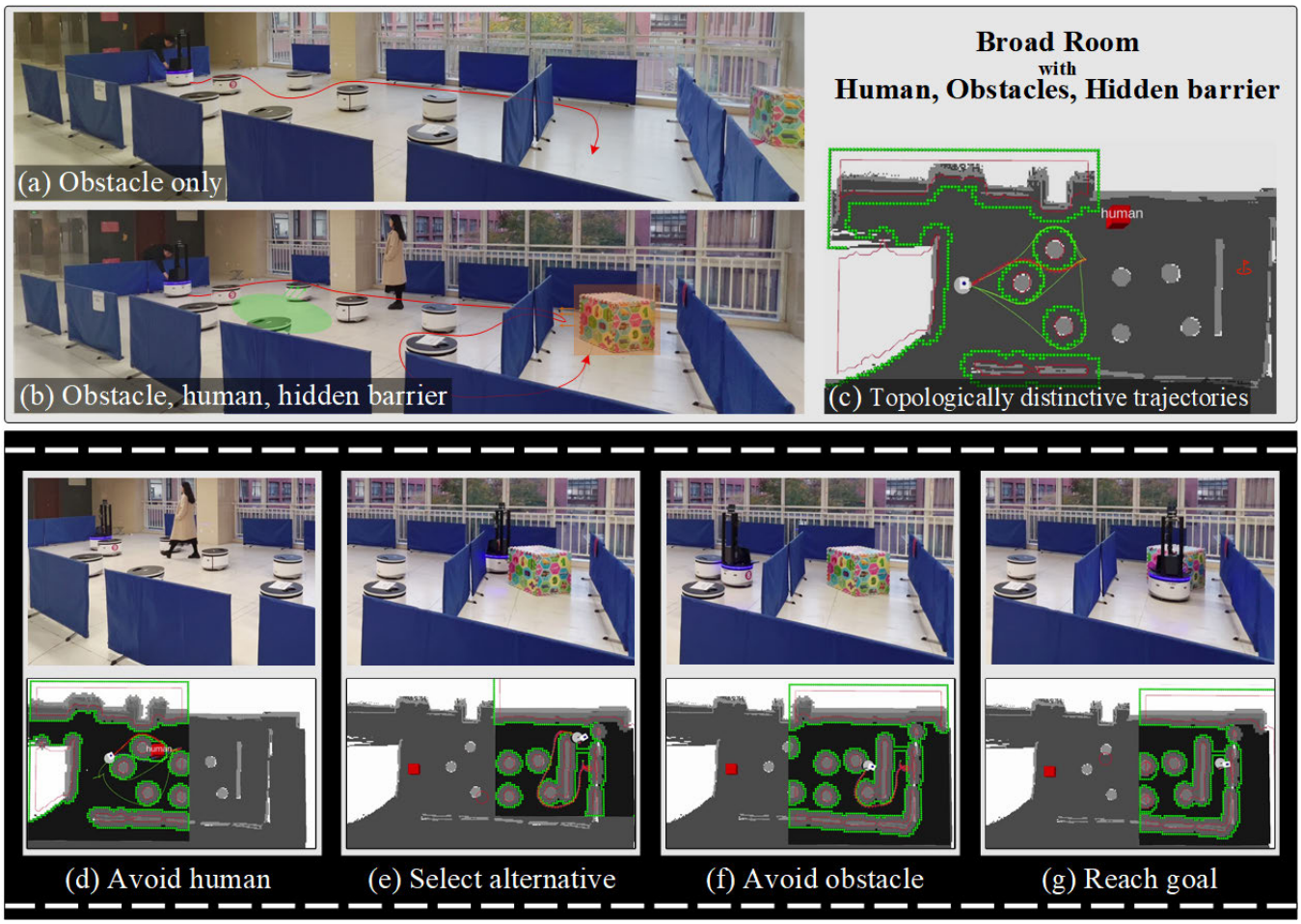


Fig. 10. Performance of GraphicTEB in broad room scenario. The robot maps the environment at (a), and a pedestrian and a hidden obstacle occur at (b). The robot first follows the shortest trajectory in (c), and changes to a safer trajectory in (d) after finding a moving pedestrian. When the robot is approaching the goal at (e), it finds a hidden obstacle and then detours at (f-g).

D. Discussion and Further Work

The above two sections compare GraphicTEB with three baselines, Voronoi-based TEB, LRpoint-based TEB, and PRM-based TEB. While GraphicTEB gaurantees to find all the topologically distinctive trajectories in the shortest time and choose safer trajectories for the robot, some limitations exist and will be our future research directions.

Shortening technique of GraphicTEB: When shortening the edge in Section.IV, once the edge covers an intervening obstacle, two new edges will be triggered, and the old edge will be defined as the combination of the shortening results of the two new edges. This method allows edges to be generated quickly without compromising the completeness of GraphicTEB(full). However, for GraphicTEB with limitations, because this technology cannot guarantee that two nodes that can be directly connected will be directly connected, a few trajectories with unnecessary detours still persist (and thus computing source is wasted to optimize them). This situation is caused by the deviation of the edge generation or the design of the shortening technique. Improving one of them or finding a better way to combine them would be a valuable issue.

Distinguish of homology classes: To explore topologically distinctive paths, the first thing to do is to generate numerous candidate paths, and assign an H-signature to each path according to the relationship between the path and obstacles. The calculation of each H-signature needs to traverse all obstacles once. This time cannot be ignored and it even approximates the time for searching paths. In this case, reducing the number of initial candidate paths can reduce the number of calculations of H-signature, thus speeding up TEB. As the generation of paths in GraphicTEB has already considered the relationship between paths and obstacles, a certain property of the graph can favor the reduction of initial candidates. For example, the red person and the green person in Fig.7(f) are two nodes connected by an edge, and the blue path can be generated by the clockwise surroundings of the red person as well as the counter-clockwise surroundings of the green person. This simple phenomenon that two connected nodes can generate a path with the same H-signature is ubiquitous. We believe it can speed up GraphicTEB, even though we have not found a satisfactory utilization yet.

Completeness in temporal-spatial space: The trajectory search of GraphicTEB is complete according to the definition in [8] (which is also widely recognized). However, this

completeness is only satisfied in two-dimensional physical space. How to achieve completeness in the three-dimensional temporal-spatial space will be our research direction.

Sensor noise and detection error: The accuracy and frequency of pedestrian detection and tracking are not good enough, resulting in errors in people's position and velocity. The inaccuracy of upstream detection will inevitably affect downstream planning. Robust motion planning that takes noise disturbances and detection errors into account will be investigated in the future.

VIII. Conclusion

This paper focuses on the exploration and selection of topologically distinctive trajectories in complex environments, and proposes a fast complete approach GraphicTEB. Different from traditional searching waypoints in collision-free regions, we focus on the collision-occupied regions, and realize graph building in $O(n)$ and path searching in $O(n!)$. In addition, an obstacle gradient is proposed to guide more efficient optimization of paths, and a safer trajectory selection strategy is proposed to reduce the uncertainty on the trajectory. Simulation demonstrates that GraphicTEB achieves complete 100% path coverage, and a fastest average search time of 0.9 ms. The trap rate of the optimization is reduced to zero, and the executed trajectory enables the robot to obtain the highest scene pass rate compared with state-of-art approaches. High computational efficiency allows the real robot to safely pass through gaps between obstacles. It can find alternative trajectories for reaching the goal and tends to pass behind people for safer running.

IX. Acknowledge

The authors gratefully acknowledge the Associate Editor and all the reviewers for their great efforts in helping to improve the quality and presentation of this paper.

References

[1] S. Proia, R. Carli, G. Cavone, and M. Dotoli, "Control techniques for safe, ergonomic, and efficient human-robot collaboration in the digital industry: A survey," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1798–1819, 2022.

[2] Y. Che, A. Okamura, and D. Sadigh, "Efficient and trustworthy social navigation via explicit and implicit robot–human communication," *IEEE Transactions on Robotics*, vol. PP, pp. 1–16, 01 2020.

[3] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ros-based mobile robot," in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 104–111.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[5] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[6] M. Pivtoraiko and A. Kelly, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2611–2616.

[7] C. Rösmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, 2012, pp. 1–6.

[8] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[9] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5681–5686.

[10] J. Wen, X. Zhang, H. Gao, J. Yuan, and Y. Fang, "E3mop: Efficient motion planning based on heuristic-guided motion primitives pruning and path optimization with sparse-banded structure," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2762–2775, 2022.

[11] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *2015 European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.

[12] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard, "Online generation of homotopically distinct navigation paths," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6462–6467.

[13] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, ser. AAAI'10. AAAI Press, 2010, p. 1230–1237.

[14] S. Bhattacharya, M. Likhachev, and V. R. Kumar, "Identification and representation of homotopy classes of trajectories for search-based path planning in 3d," in *Robotics: Science and Systems*, 2011.

[15] A. Vega, L. Manso, D. Macharet, P. Bustos, and P. Núñez, "Socially aware robot navigation system in human-populated and interactive environments based on an adaptive spatial density function and space affordances," *Pattern Recognition Letters*, vol. 118, no. 1, pp. 72–84, Feb. 2019.

[16] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807 vol.2.

[17] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*, 2013, pp. 138–143.

[18] L. Yin, M. Zhang, J. Wang, X. Gao, and J. Shen, "An improved method of dynamic path modification for nonholonomic mobile robot," in *2021 3rd Asia Pacific Information Technology Conference*, ser. APIT 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 96–100.

[19] K. Schlegel, P. Weissig, and P. Protzel, "A blind-spot-aware optimization-based planner for safe robot navigation," in *2021 European Conference on Mobile Robots (ECMR)*, 2021, pp. 1–8.

[20] V. B. Hoang, V. H. Nguyen, T. D. Ngo, and X.-T. Truong, "Socially aware robot navigation framework: Where and how to approach people in dynamic social environments," *IEEE Transactions on Automation Science and Engineering*, pp. 1–15, 2022.

[21] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1986.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. 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*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[23] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1208–1214.

[24] J. S. Smith, R. Xu, and P. Vela, "egoteb: Egocentric, perception space navigation using timed-elastic-bands," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2703–2709.

[25] L. Yi-bo and L. Jun-jun, "Harris corner detection algorithm based on improved contourlet transform," *Procedia Engineering*, vol. 15, pp. 2239–2243, 12 2011.

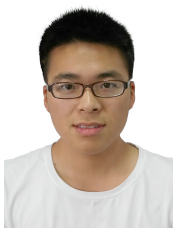
[26] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, p. 560–570, oct 1979.

[27] X.-T. Truong and T. D. Ngo, "Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1743–1760, 2017.

[28] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time dependent planning on a layered social cost map for human-aware robot navigation,"

in *2015 European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.
[29] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1662–1668.
[30] P. Teja S. and R. Alami, "Hateb-2: Reactive planning and decision making in human-robot co-navigation," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 179–186.
[31] D. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in *AAAI*, 2006.
[32] B. Okal and K. O. Arras, "Towards group-level social activity recognition for mobile robots," in *IROS Assistance and Service Robotics in a Human Environments Workshop*. International Society of Biomechanics Chicago, IL, USA, 2014.
[33] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, "Pedestrian recognition using high-definition lidar," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 405–410.
[34] M. Häselich, B. Jöbgen, N. Wojke, J. Hedrich, and D. Paulus, "Confidence-based pedestrian tracking in unstructured environments using 3d laser distance measurements," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4118–4123.
[35] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419841532, 2019.
[36] H. Khambhaita and R. Alami, "Assessing the social criteria for human-robot collaborative navigation: A comparison of human-aware navigation planners," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 1140–1145.
[37] Z. Wang, Y. Zhuang, Q. Gu, D. Chen, H. Zhang, and W. Liu, "Reinforcement learning based negotiation-aware motion planning of autonomous vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4532–4537.
[38] P. T. Kyaw, A. V. Le, P. Veerajagadheswar, M. R. Elara, T. T. Thu, N. H. K. Nhan, P. Van Duc, and M. B. Vu, "Energy-efficient path planning of reconfigurable robots in complex environments," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2481–2494, 2022.

Qianyi Zhang received the B.S. degree in intelligent science and technology from Nankai University, Tianjin, China, in 2021. He is currently working towards the Ph.D. degree in artificial intelligence, under the supervision of Prof. Jingtai Liu, with the Institute of Robotics and Automatic Information Systems, Nankai University, Tianjin, China. His research interests include the motion planning of mobile robot, decision-making of automated vehicle, and human-robot game.



Shichao Wu received the B.S. degree in automation from the Southwest University of Science and Technology, Mianyang, China, in 2017, and the M.S. degree in robot science and engineering from Northeastern University, Shenyang, China, in 2020. He is currently pursuing the Ph.D. degree at Nankai University, Tianjin, China. His research interests include context-based emotion recognition, acoustic events classification and localization.

Yuhang Jia is currently working towards the B.S. degree in computer science and technology, under the supervision of Prof. Jingtai Liu, with the Institude of Robotics and Automatic Information Systems, Nankai University, Tianjin, China. His research interests include motion planning of mobile robot and computer vision.

Yuang Xu is currently working towards the B.S. degree in computer science, under the supervision of Prof. Jingtai Liu, with the Institude of Robotics and Automatic Information Systems, Nankai University, Tianjin, China. His research interests include the deep learning and computer systems.

Jingtai Liu received his B.S. degree and the M.S. degree from Tianjin University in 1983 and 1986 respectively. And he received his Ph.D. degree from Nankai University in 1998. In 1986, he joined Nankai University and co-founded the Robotics Laboratory. He was once a member of Expert Group of the Intelligent Robots Subject of the High-tech Research and Development Program of China. During 2006-2007, he was a visiting scholar with the Laboratory for Automation Science and Engineering, University of California, Berkeley.