Tangent-based path planning with distinctive topologies in a dynamic environment

Zhuo Yao[1,*]

Abstract— **This manuscript proposes a novel tangent-graph-based method that provides all distinctive topology paths, as a set of trajectories that can not be transformed into each other by gradual bending and stretching without colliding with obstacles. As the global optimal trajectory (limited by dynamic constraints) may not have the same topology as the shortest path (almost doesn't considering dynamic constraints), it is important to provide all distinctive topology paths rather than just the globally shortest one. One of the application is provide initial paths for trajectory planning methods, such as time elastic band (TEB) and dynamic window approach (DWA).**

Considering that a mobile platform is always working in a dynamic changing environment and that update the whole graph is time consuming, a locally rectify method is proposed to update tangents according to local map provided by sensor reading. Finally, results under real grid maps are presented.

I. INTRODUCTION

Path planning is a fundalmental research area in mobile platform applications. With the expansion of requirements and navigation constraints , general single-output path planning is not suitable to cope with this problem. Online trajectory planning methods (e.g, TEB[8] and DWA[2]) are able to find solutions under complex constraints and scoring function and generate trajectories that considering dynamics, but most of them they need good initial paths as input and are unable to transform them across obstacles (in other words, the initial path and the resulted trajectory must have the same topology). Therefore, it is important to determine all distinctive topology paths rather than a globally shortest path. Compared to homotopy class paths, any two of the distinctive topology paths are belong to different homotopy classes. The idea of utilizing topologically distinctive paths for planning and navigation is not novel. Some contributional works have focused on path planning under topology constraints.

Homotopy preserving probabilistic roadmap (HPPR)[9] is a sampling-based method that operate with albeit different equivalence relations and can provide all of the solutions which belong to different homotopy classes. The algorithms are probabilistically complete to identify distinctive topology paths in large, complex environments. However, this approach lacks adaption to dynamic environments and the probabilistic road map from grid map cost substantially time, and this limited its application in complex real scenes. PDRs [3] also apply PRM to get distinctive topology paths, and expand it to free-flying and articulated robots in both 2D and 3D environments. Comparing these graph-based methods, the proposed method only needs to update the tangent graph partialy, enable itto be efficient under a dynamically changing environment.

Bhattacharya et al. [1] presented complex analysis to determine whether two paths have the same topylogy and proposed general path planning with homotopy constraint, which can be apply to both point-by-point travesal and graph search-based path plannings. And it was proved to be effective by [7]. This approach provides a close form of solution, however its computaional cost under a large scale map increases sharply. Compared to complex analysis, our method uses the geometric properties of the locally shortest path to limit the search range and is insensitive to map scale increases but sensitive to the fragment degree of obstacle boundaries.

Tangent-based path planning has been extensively studied in recent years and has been proven to be efficient at finding the shortest path in 2D maps. A tangent graph has a similar structure as a visibility graph; the major difference is that it uses the tangent to represent the visibility between the grid on the surface and avoids the limitations of polygonal obstacle shapes. Liu and Arimoto [4, 5, 6] first proposed the locally shortest path, the tangent graph and described how to find the shortest path from the graph. The major difference between the tangent graph and the visibility graph is that the tangent graph can be applied to obstacles with arbitrary boundaries. Then, we proposed a series of tangent graph-based path planning method [10, 11] which search the global shortest path in form of path tree instead of a queue, which is always used by general path planning. These approaches can always find the shortest path, and the time cost is insensitive to the map scale increase.

Based on previous work, a tangent-graph-based method that provides all distinctive topology paths was proposed in this paper, and the process is shown in Fig. 1. The rest of this paper is organized as follows. Section II introduces the relationship between the locally shortest path and the homotopy class path and how to determine the tangent graph from grid map, corresponding to Fig. 1 A. Section III presents how to search the distinctive topology paths from the tangent graph, corresponding to Fig. 1 B. Section IV presents the planning result with dynamic obstacles. Section V concludes above sections and future work directions.

II. DETERMINING THE TANGENT GRAPH

This section includs two parts: 1. The tangent and homotopy class path: clarifying the connection between tangent

*This work was not supported by any organization

[1]Zhuo Yao was with the Department of Mechatronic, Beijing Institute of Technology, Beijing 100081, China. Now he is an independent researcher. 1521232476@qq.com

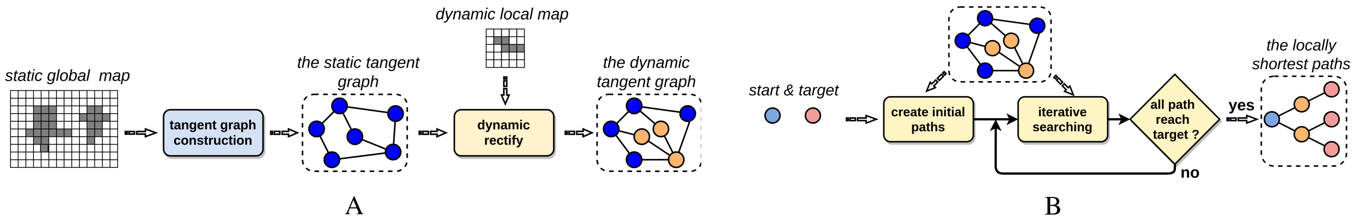

Fig. 1. A: The process of obtaining dynamically updated tangent graph. B: The process of obtaining locally shortest paths from dynamic updated tangent graph.

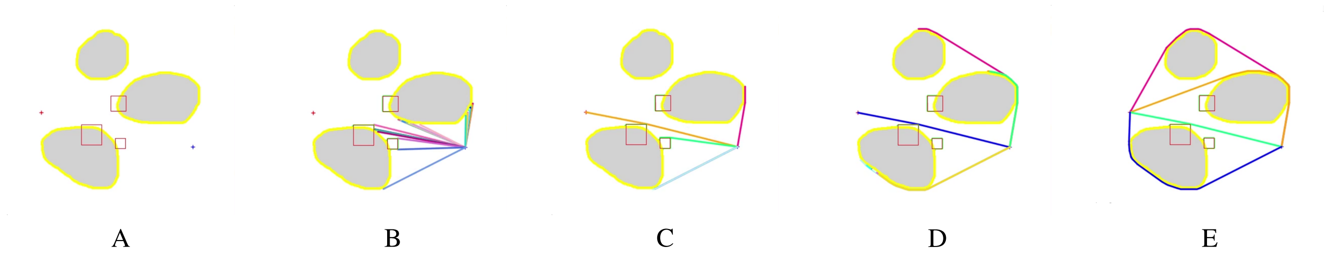

Fig. 2. These figures show an example of the process of searching distinctive topology paths, which represented by different color curves. The size of map is 450 * 600 pixels and it takes 29ms to obtain all paths. And the blue cross is the start and the red is the target, as described below.

and homotopy class paths and how to determine tangents from an arbitrary occupancy grid map. 2. The tangent graph update: describing how to update the tangent graph with a dynamic local map.

A. *Tangent and homotopy class path*

As defined in [1] and [7], homotopy class paths is define as a set of paths which can be smoothly transformed into one other without intersecting obstacles. In this manuscript, we define the locally shortest path as a path that cannot obtain a shorter path by replacing any waypoint with any point from its neightbor. Both locally shortest and homotopy class paths are defined based on increment movement of waypoints. Essentialy, a locally shortest path is one of the shortest path in a set of homotopy class paths. Therefore, if we obtain all the locally shortest paths that connect the start and the target, we can obtain paths with all possible distinctive topologies.

To obtain locally shortest paths, we propose a tangent-based path planning method. The tangent is defined as follows, which is proven to be locally shortest. Before introducing the tangent, some basic concepts should be clarified.

Neighborhood: $U(g)$ is defined as the four nearest grids of grid g.

Frontier:$Fr(g)$ is defined as the eight nearst grids of grid g.

Surface: if grid g is free to pass but there is an occupied grid in $U(g)$, define it as a grid on the surface. Denote the set consisting of all grids on surfaces as S.

Briefly, if $g \in F, \exists g' \in U(g) and \exists g' \in O$, then $g \in S$.

Tangent: 1. There is at least one end of the tangent on the surface; 2. For a tangent AB, $A \in F, B \in F$ and $l(A,B)$ (equal to line AB) cross no obstacles. 3. For any grid on the surface (A or B or both), denote it as A, and denote the other as B. There must be both $\exists g_1 \in Fr(A) \cap S$, $l(B,g_1)$ crossing obstacles locally and $\exists g_2 \in Fr(A) \cap S$, $l(B,g_2)$ does not cross obstacles at the same time.

Theorem 1. *The locally shortest path consists of tangents: if a line in a path is not a tangent line, the path is not locally shortest. The only exception is when both ends of the line are not on the surface, meaning $l(g_{start}, g_{target})$ crosses no obstacles, which is easy to distinguish.*

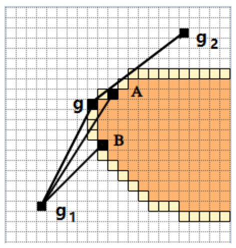

Fig. 3. This figure shows an example of a tangent and two other possibilites of the shortest path. Dark yellow grids represent obstacles and yellow grid represent boundaies.

Proof 1. *As proved in [10], the waypoints of the locally shortest path must be on surface. According to the definition of tangent and the waypoints of the shortest path must be on surface, there are two possible situations ($g_1 \rightarrow A$ and $g_1 \rightarrow B$) for a line in the shortest path except tangent ($g_1 \rightarrow g$):*

1, $\forall g' \in \{Fr(A) \cap S\}$. $l(g_1, g')$ crosses obstacles locally; which means that $l(A,B)$ also crosses an obstacle, as in point A shown in Fig. 3, which contradicts the definition of the shortest path;

2, $\forall g' \in \{Fr(B) \cap S\}$, $l(g_1, g')$ does not cross any obstacles locally, and thus, it is feasible to construct a shorter path by replacing B with a grid in $\{Fr(B) \cap F\}$, as point B shows in Fig. 3. This contradicts the definition of the locally shortest path.

Thus, the locally shortest path consists of tangents.

To search locally shortest paths more efficiently, tangents are stored in the form of tangent graph, in which vertices represent tangent nodes and edges represent tangents. Some examples of occupancy grid maps and related tangent graphs are shown in Fig. 5 A(static), B(static) and C(static), and the time cost can be found in Table. I.

B. Locally rectifying of tangent graph

The occupancy map keeps changing in real-world applications, and the static tangent graph take considerable time (several minutes) to reconstruct. Therefore, we proposed a tangent graph dynamic rectification process that partially updates the static tangent graph partially and retains the properties of the tangent graph, enabling our method get used to dynamic change environment.

This process consists of three steps:

1. Denote tangent nodes that not on surface in the new map (merged static map and dynamic local map) and tangents that cross dynamic obstacles as illegal; they will not be considered when searching paths and will be reset to legal after current planning;

2. Detect new boundary grid and store them in continuous order to prepare for determine new tangents;

3. Calculate the tangent inside the dynamic boundary and between the dynamic and static boundaries. As if a grid is tangent to another boundary, it must be tangent to the boundary it is on, and the method determine the tangent node inside the dynamic boundary first and then compute the tangent between these nodes and static tangent nodes. Finally, insert all new tangents into the tangent graph and remove all new tangents after current path planning is finished.

An example of a dynamic rectified tangent graph is shown in Fig. 4. It is obvious that tangents that cross obstacles are removed in the dynamic tangent graph and tangent of new obstacles are inserted.

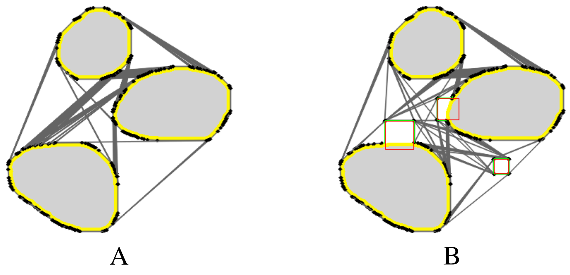

Fig. 4. These figures show an example of the static tangent graph (A) and rectified tangent graph (B). The red rectangles represent three dynamic obstacles, the gray lines represent tangents, the dark points represent tangent points, and the light gray area represents obstacles; the same represenions apply below.

III. SEARCH LOCALLY SHORTEST PATHS

A. Create initial paths

As the start and target are not in the graph, we need to determine the tangent from these points to the tangent graph as initial paths. As the tangent graph is updated, we need to consider the difference between the dynamic boundary and static boundary. In a static map, a continuous boundary belongs to a common boundary; however, in a dynamic environment, a dynamic boundary may connect to a static boundary, but they have a different boundary ID, as shown in Fig. 4 A. Therefore, we need to consider a continuous boundary including both dynamic boundary and static boundary as a unit to create initial paths. A BFS-based method is proposed to classify boundaries according to whether they are connected to others; an example is shown in Fig. 7. A group of continuous boundaries are considered to be one when creating initial paths.

We note all tangent nodes boundaries between the start and target as candidates, as we did in [10], as considering all nodes in the tangent graph is time consuming and not neccesary. Then, collision check are performed for all connections between the start and candidates, and all collision free

Algorithm 1: Search for paths from tangent graph

```
Input: $start$, $target$, $initPaths$, $G$, $O$, $F$
Output: the locally shortest paths
1  $paths = initPaths$;
2  while not all $path \in paths$ reach $target$ do
3      $newPaths = \phi$;
4      for $path \in paths$ do
5          $newPaths \leftarrow$ new paths created from $path$;
6      if only search global shortest path then
7          if any $path \in newPaths$ reach the target then
8              $maxLength$ = length of the shortest
                 finished path in $paths$;
9          else
10             $maxLength = \infty$;
11     $paths$ = $newPaths$;
12 return $paths$;
```

Algorithm 2: Create new paths from the previous paths

```
Input: $O, F, G, target, path$
Output: $newPaths$
1  // $path$: a previous path
2  // $newPaths$: new paths created from $path$
3  $newPaths = \phi$;
4  if $connection$ of the $target$ and the last point of $path$
     cross no obstacles then
5      $newPath$ = $path \leftarrow target$;
6      $newPaths \leftarrow newPath$;
7      return $newPaths$;
8  $tangentCandidates$ = tangents in $G$ that one of the
     two end points is the last point of $path$;
9  for $tangent \in tangentCandidates$ do
10     if $tangent$ meet no conditions of cutting branch
         then
11         $newPath$ = $path \leftarrow tangent$;
12         $newPaths \leftarrow newPath$;
13 return $newPaths$;
```

doi:10.20944/preprints202108.0034.v1

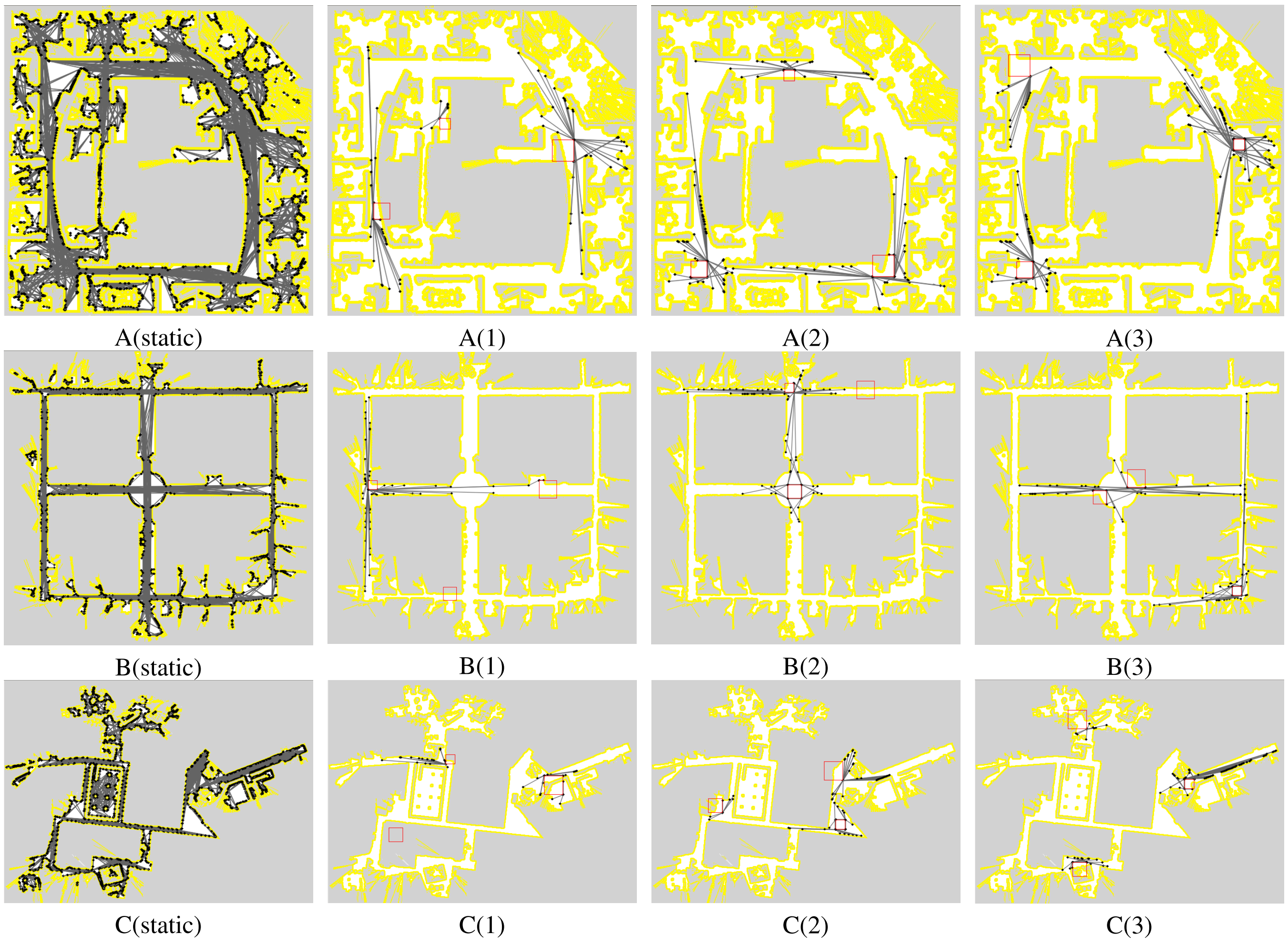

Fig. 5. These figures show the static tangent graph and three examples of dynamic tangents resulting from dynamic obstacles (red rectangle) with indices 1, 2 and 3 under grid maps A, B and C.

connections are selected as initial paths. Collision checks are performed for all connections of the target and candidates, all tangent node that belong to collision-free connections are stored as target nodes. In the iterative search (next section), when a path reaches one of the target nodes, we mark it as reach target.

B. *Path planning with distinctive topologies*

In this section, we use an adaptive breadth first search (BFS) method to obtain all locally shortest paths, as shown in Algorithm. 1 and 2. This section only considers the geometrical requirement of the shortest path, and dynamics or mechanics are not considered.

We use a path tree to replace the queue used in general BFS to record all histories of iteration. The nodes in the path tree store related tangent points and pointers to previous and next path tree nodes, and the edges represent tangents between nodes. Furthermore, it is possible for a parent node to have more than two child nodes in a path tree. When the method finds all finished paths (in other words, all leaf nodes of the path tree reach the target), we terminate searching.

To reduce the size of the path tree and accelerate the searching process, three branch cutting conditions related to the geometric nature of the shortest path are introduced:

1. The available range ([10], [11]) that limit the orientation of tangent during tangent graph searching;

2. Remove paths that contain tangents which cross static or dynamic obstacles;

3. Remove path that intersect with the itself, or containing the same tangent node twice or more. If we do not remove path containning loop, the search will never end.

If we only want the global shortest path, extra branch cutting conditions are required: if a node in the tangent graph is visited by a shorter path, then all paths created from the longer path are removed from the path tree; any paths longer than the shortest finished path are removed.

To obtain all locally shortest paths from the tangent graph, we obtain all distinctive topology paths. A simple example of search is shown in Fig.2. Examples under complex real word maps are shown in Fig. 6, and the related time cost can be found in Table II.

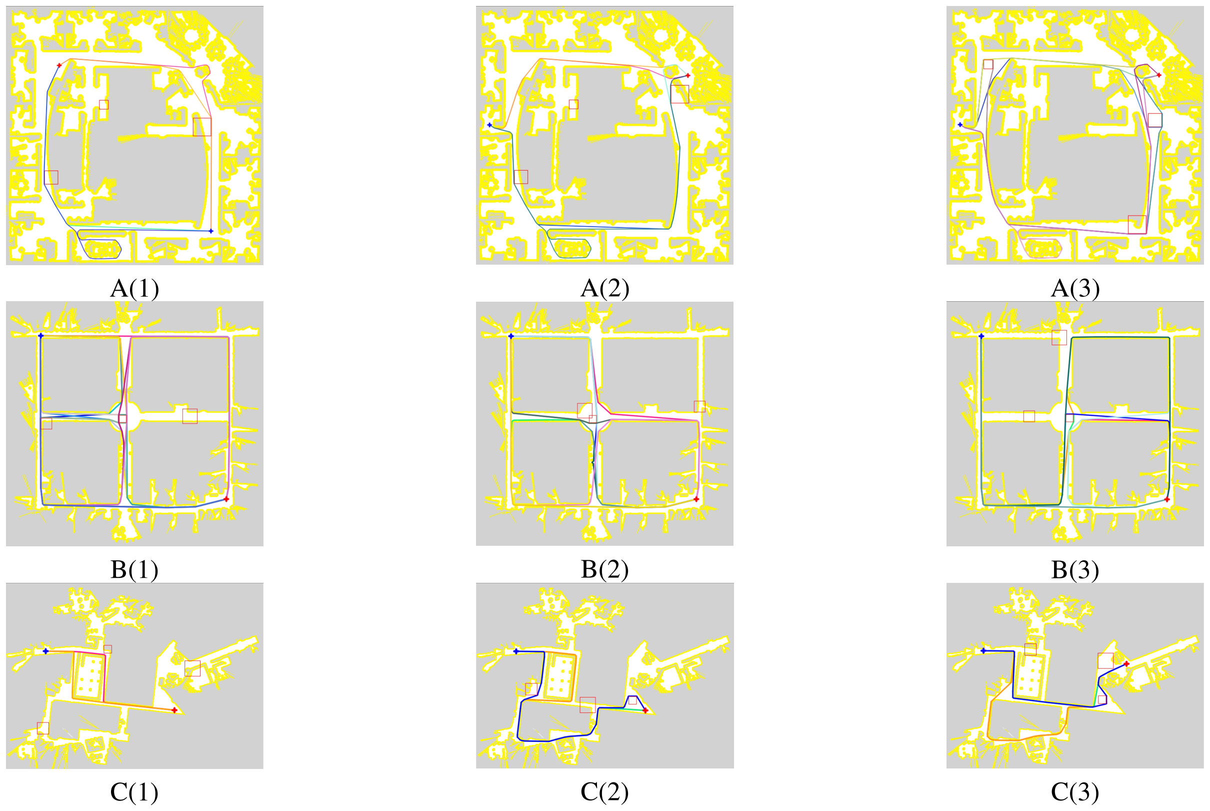

Fig. 6. These figures show the planning result with several dynamic obstacles (red rectangle) with indices 1, 2 and 3, under grid maps A, B and C. Different colors represent different paths.

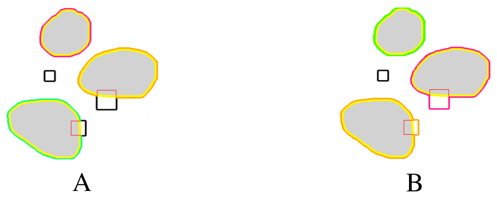

Fig. 7. These figures show an example of the difference between unclassified (A, 6 boundaries) and classified boundaries (B, 4 boundaries).

IV. RESULTS

The section mainly focuses on how the proposed algorithm works under grid maps created from real sensor data [1]. This section focuses both the dynamic rectification of the tangent graph and the distinctive topology paths, corresponding to Section III and Section IV, respectively. Considering the requirements of real applications, time costs are also considered.

All experiments were conducted on a computer with a 2.30 GHz Intel CPU and 7.3 GB RAM.

A. Locally tangent graph update

In this section we place some kinds of random dynamic obstacles three times (with indices 1, 2 and 3) on three occupancy grid maps (A, B and C) and record both the static tangent graph, the new tangents caused by dynamic obstacles and the time cost of determining them. The results are shown in Fig. 5 and statistics are presented in Table I.

TABLE I
MAP INDEX, SIZE(PIXEL) AND TIME COST

index/size	static	1	2	3
A / 579*581	186s	767.809ms	817.203ms	818.211ms
B / 703*667	73.9s	688.555ms	795.825ms	656.164ms
C / 668*482	69.3s	718ms	778ms	779ms

As shown in Table. I, it takes 1 to 3 minutes to create the static tangent graph and approximately 1 second to update the graph, which is acceptable in real-world applications. Map A is smaller than Map B but requires more time to create the static tangent graph, because Map A has more fragmented boudaries (yellow area) than Map B. The time cost of creating the static tangent graph is closely related to the degree of fragementation of obstacle boundaries, as more fragmented boundaries cause more tangents. therfore, removing unimportant noise in the grid map will significantly reduce the time cost.

B. Search distinctive topology path

This section shows the planning result under the above maps with the random dynamic obstacles, start and target. Details about planning are shown in Fig. 6 and statistics are presented in Table. II. It is noteworthy that the time

[1]https://www.ipb.uni-bonn.de/datasets/

cost includes all processes of search path, including creating initial paths and the iterative search.

TABLE II
MAP INDEX, TIME COST AND COUNT OF DISTINCTIVE TOPLOGY PATHS

index	1	2	3
A	94.0$ms/4$	110.2$ms/6$	382.95$ms/24$
B	328.9$ms/25$	112.1$ms/10$	116.4$ms/7$
C	54.1$ms/2$	94.4$ms/4$	67.7$ms/4$

As shown in Table. II, the proposed method finds all distinctive topology paths. And the time cost of search them is closely related to the total number of them, because the size of the path tree increases as the number of paths increases. And its time cost is also influenced by the number of tangents.

V. CONCLUSIONS

In this paper, we proposes an efficient tangent-based method to find all distinctive topology paths under a dynamically changing environment. Using an adaptive BFS that considers the geometrical properties of the locally shortest path, we can find all distinctive topology paths from the tangent graph in less time than general graph search method. Comparing to sampling based path planning which is probabilistic complete to find all distinctive topology paths, the proposed method is complete to find them. And as tangent based method is insensitive to map scale increasing (proved in [10]), the method is insensitive to map scale increasing too.

In results, we have experimentally demonstrated its efficiency on several real-world maps and prove that its time cost is acceptable for real-word applications. Its applications including provide initial paths for trajectory planning methods (e.g. TEB and DWA), or multi-robot path planning. In the future, it will be meaningful to transplant the method to 3D environment and the time cost will be the major difficulty we need to overcome.

ACKNOWLEDGMENT

The author thanks Lu Zhu for her encouragement and support during the consummation of this manuscript.

REFERENCES

[1] Subhrajit Bhattacharya. “Search-based path planning with homotopy class constraints”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 24. 1. 2010.

[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.

[3] Léonard Jaillet and Thierry Siméon. “Path deformation roadmaps: Compact graphs with useful cycles for motion planning”. In: *The International Journal of Robotics Research* 27.11-12 (2008), pp. 1175–1188.

[4] Y-H Liu and Suguru Arimoto. “Proposal of tangent graph and extended tangent graph for path planning of mobile robots”. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE. 1991, pp. 312–317.

[5] Yun-Hui Liu and Suguru Arimoto. “Computation of the tangent graph of polygonal obstacles by moving-line processing”. In: *IEEE Transactions on Robotics and Automation* 10.6 (1994), pp. 823–830.

[6] Yun-Hui Liu and Suguru Arimoto. “Path planning using a tangent graph for mobile robots among polygonal and curved obstacles: Communication”. In: *The International Journal of Robotics Research* 11.4 (1992), pp. 376–382.

[7] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. “Planning of multiple robot trajectories in distinctive topologies”. In: *2015 European Conference on Mobile Robots (ECMR)*. IEEE. 2015, pp. 1–6.

[8] Christoph Rösmann et al. “Efficient trajectory optimization using a sparse model”. In: *2013 European Conference on Mobile Robots*. IEEE. 2013, pp. 138–143.

[9] E. Schmitzberger et al. “Capture of homotopy classes with probabilistic road map”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. 2002, 2317–2322 vol.3. DOI: 10.1109/IRDS.2002.1041613.

[10] Z. Yao et al. “ReinforcedRimJump: Tangent-Based Shortest-Path Planning for Two-Dimensional Maps”. In: *IEEE Transactions on Industrial Informatics* 16.2 (2020), pp. 949–958. DOI: 10.1109/TII.2019.2918589.

[11] Zhuo Yao et al. “RimJump: Edge-based Shortest Path Planning for a 2D Map”. In: *Robotica* 37.4 (2019), pp. 641–655.