Tree-like Structure Path Planning using Real-time Sensors data and Way Points

Yanning Hu¹, Decai Li², Feng Gu³, Liying Yang³, Yuqing He⁴ and Jianda Han⁴

Abstract—Inspired by the storage structure of graph search algorithm, we propose a tree-like data structure (TLDS) for environment information storage. The robot uses only some sparse information of global environment and can easily slips into the dead end or the wrong way, especially the forks in the road. Moreover, based on the TLDS, a complete planning scheme can be obtained. It can help the robot to search in each branch road it may encounter, and to return to the fork in the road for re-planning when the mobile robot know the road is wrong. Simulation results are presented along with two different kind purposes, the first Mini-Test is to verify the feasibility of the tree-like data structure, and the second Test is to verify the complete set of solution.

Keywords—mobile robot, path planning, graph search algorithm, road shape environment.

I. INTRODUCTION

PATH planning Ref. [1] is an important part of mobile robot navigation, especially for mobile robot based on local sensors. The robot based on local sensors easily trapped in local minimum, so that the planning tasks become quite complex and a lot of research has been made in the path planning of mobile robot using real-time sensors, which means there is no a priori global map for robot.

The path planning methods of mobile robot based on sensors are roughly divided into two categories. The first class is called active path planning which means the robot will initiative to obtain a precise global map when it is confronted with the new environment. A comprehensive and typical method has been proposed in Ref. [2]. The method use a Rapidly-exploring Random Tree (RRT) Ref. [3] to incremental build a reusable global road map. This kind of planning method is first to make an exploration of the environment and then carry out the planning task on the results of exploration. The efficiency of path planning based on this method is not high, when the working environment of mobile robot is frequently update.

The second class is named passive path planning, which means they accept the presence of the local minimum and alternatively use a global decision module to make robot jumps out the local minimum. This method first need an effectively local path planner to plan with local map gained by vehicle-sensors, and the second needed is a global path planner which is use the imprecise global information to guide the robot to reach its destination. This is a two layers of planning scheme which has been detailed instructions in Ref. [4]. This scheme is used for fast and robust vehicle positioning in Ref. [4], we can also apply it to path planning of mobile robot (see Figure 1). The application of this scheme reduces the dependence on global information.

![Image](image.jpg)

Fig.1 Two layers robot path planning model.

An effective local planner is required by the first layer, and many effective local planning methods have been proposed in the related works. A sensitive response to the dynamic environment local planner is proposed in Ref. [5], which is called dynamic window approach (DWA). This is a kind of incremental planning method which is perfect at local planning with partial visible or dynamic changing environmental. A similar method was put forward in Ref. [6] which is a trajectory planning method based on quartic Bézier curve.

Lots of global path planning method was proposed for this two layers of path planning scheme. Ref. [7] use Anytime D* Ref. [8] as its global path planning algorithm and Ref. [9] use a new method based on A* Ref. [10] called Hybrid A* to generate arbitrary trajectories irrespective of a specific road structure. The effectiveness of those algorithms is based on a precise global map which can incremental construction in the process of planning. So it cannot be directly applied to planning of the starting point to the target point. As an adaptive change, those algorithms usually used in the planning of current state to a local target. These global planning algorithms is not effective to make robot jump out of local minimum. Refs [7, 9, 11] use a complex Finite State Machine (FSM) to make up for the inadequacy of global path planning. There is no denying the fact that it makes the system more stable relatively. In fact, to independent the unexpected events from the planning task.
lost a lot of continuity. This kind of behavior is intuitive see a stuttering in visually. This increases the time cost of the plan. It is important to think carefully is how to embed the local path planning into the global path planning.

It is worthy of study the method that can put the environment structure integrated into planning task. Many parts of the earth’s environment had the human trail. The definition of road and the relationship between the human and the road are introduced in Ref. [12]. It is important for mobile robot has the concept of road, but the ability of robot to analyze the semantic far less than human. We just give a more relatively simple concept (e.g. road shape, road centerline) of road to the robot. Many complex problems become easier, when the task is planning in road-shape environment. Refs [7, 9, 11] put the heuristic in planning task, which looking for road shape actively. A more useful heuristic is treating the road as a tree which make the path planning problem approximate a graph search problem. TLDS is on the basis of graph search algorithms Ref. [13].

The main contributions of this paper are as follows:

1) We introduce a new data structure, TLDS inspired by graph search algorithm, to storage environment information. The ability of TLDS to memories enables the robot autonomous jump out from the local minimum and re-plan. The unique approach that embedding the environment structure into the planning framework make the planning fluent and natural and reduces the state number of system state machine.

2) We also give an idea that using the projection of waypoints instead of the original waypoints. We put the current waypoint projection to the central line of road. It is easy to determine whether the waypoint was visit successful after using projection of waypoint as current waypoint.

The rest of this paper is structured as follows. Section II describes the tree-like structure path planning using local map in more detail. Then, section III show the simulation result and a detailed analysis and description. And finally, the paper end with a conclusion In section IV.

II. METHODOLOGY

In this section we explain the mobile robot path planning base on TLDS using local map which can build by real-time sensors’ data in more detail. First, we will give some assumptions of the method proposed in this paper. Second, we will give the tree-like data structure and use an example to explain it. Finally, the details about how to apply the tree-like structure in path planning will be discuss.

A. Background Restatement and Assumptions

The method proposed in this paper mainly deal with the environment which has a basis road shape. Robot can get the local environment information and coarse global information from sensors or other approaches.

The method of this paper can be set up based on the following two assumptions was established:

- The upper bound of the GPS waypoints is known.
- The mobile robot should have the power to travel in bi-directional.

In figure 2 shows a simple example of path planning on the fork. We will use this figure to explain why the two assumptions are needed in this paper. In this paper, we presume that the robot can use the real-time sensors data to build a local map and the map can update as the robot run in the unknown environment.

![Fig. 2 A simple example for the fork road path planning.](image)

In the initial state, robot in the position of the starting point (Green point in figure 2). The task is to make the robot reaches the goal point (red point in figure 2). Robot will encounter the fork in road (Brown lines in figure 2). Then, robot will comprehensive think about the real-time sensors data and current waypoint (Dark red dots in figure 2) and make the wrong decision which will choose the way cannot achieve the goal. Assume that the robot according to the wrong way to move forward to blue dot in figure 2 and found that the road is not workable (we need find such a function which can judge whether a way is workable or not, hypothesis 1 is needed in the decision function.). robot will be returned to the fork in the road and re-planning. According to hypothesis 2, robot can reverse driving, which make the return task easier. It is also the indispensable conditions of that robot can use tree-like data structure to path planning, which will be explained in detail in the next part of this section.

B. Tree-like Data Structure (TLDS)

TLDS is the key to integrate the abnormal events into normal planning task. First of all, we would like to introduce a few definitions which are the necessary components of TLDS. Figure 3 give an example to present TLDS. The black area looks like road represent the drivable field. In the figure 3, we can see eight dots in different colors. The blue dot marked with f1 is a mark point of the fork in the road. The two white dots in figure 3 are the leading points of each branch road, with which we can go to the branch roads. The number of leading points in a fork is equal to the number of branch roads. The remaining five points are GPS waypoints including one start point, one goal point and three normal waypoints.

Before introducing the TLDS, we first give some data storage unit:

- **Visitedwaypoints**: it is the vessel to store the reached waypoints according to the order.
- **Unvisitwaypoints**: it is the vessel to store the unreached waypoints according to the order.
• *Leadingpoints*: it is the vessel to store the leading points of all branch roads.

Road in the real environment has a lot of branch roads, which like a tree has a lot of branch. However, a feasible path from starting point to target point has no branch. The robot using TLDS can through an iterative process to handle all the forks in its planning task. We treat the starting point as a special fork in the road in order to use this kind of iteration. The number of iteration is equal to the number of fork in the road. The contents of the iterative process are divided into the following three steps: (1) initialize a child trunk, (2) Search a right sub trunk in those branch roads, and (3) Combine the right sub trunk with its parent trunk as next parent trunk. The global path planning algorithm is given in Figure 4.

1) *Reset TLDS.*

Both child trunk and father trunk contains two types of data objects: Visitedwaypoints and Unvisitedwaypoints. *Leadingpoints* is a public data object of the current fork in road. We define two tree trunk objects: (1) *Child*, (2) *Father*.

We assume that the marked point of current fork in the road is *Fork*, the starting point is *Start*, the target point is *Goal*, and the Waypoints that represent the waypoints is \([P_1, P_2, \ldots, P_n]\). To initialize the object *Child* as follow:

*Visitedwaypoints* = \([ g_{start} ]\)

*Father.Visitedwaypoints* = *Start*

*Father.Unvisitedwaypoints* = \([ \text{Waypoints, Goal} ]\)

*Child.Visitedwaypoints* = \(g_{start}\)

*Child.Unvisitedwaypoints* = *Father.Unvisitedwaypoints*

The robot has the ability to return to the previous fork in the road (junction) when current road is judged as wrong way. The TLDS will be reset as long as the robot reach the previous junction. We assume that current junction has \(m\) branch roads, and robot is running on the \(i\)-th branch road which is a wrong road. Robot will back to the junction and reset TLDS as follow:

*Leadingpoints* = \([g_{i+1}, g_{i+2}, \ldots, g_m]\)

*Child.Visitedwaypoints* = \(g_{i+1}\)

*Child.Unvisitedwaypoints* = *Father.Unvisitedwaypoints*

2) *Update TLDS*

We assume the leading point set of current fork in the road is \([g_1, g_2, \ldots, g_m]\), which was defined as *LEADINGPOINTS*, and of those branch roads must be the right way to target. Due to the error in the rough global information, the robot may be give the wrong branch road a higher value than a right one, and ignore the right way at the beginning. Our method can make the robot judge a wrong way and return to the current fork in the road. The robot will make an exploration for each branch road in turn, until it found the right one. If the robot driving on the branch road encounters a fork in the road, suggests that it is a right way. If it is a wrong way, then the program to jump to step (1), and use the next element in *Leadingpoints* for the current navigation point.

If robot encounters a new fork in the road, we will combine the right sub trunk with its parent trunk as the next parent trunk which is prepare for a new iteration. Specific merging process is as follow:

*Child.Visitedwaypoints* = *Fork*

*Child.Unvisitedwaypoints* = *Father.Unvisitedwaypoints*

*Father.Visitedwaypoints* = \([\text{Father.Visitedwaypoints, Child.Visitedwaypoints}]\)

*Father.Unvisitedwaypoints* = *Child.Unvisitedwaypoints*

*Leadingpoints* = \([g_1, g_2, \ldots, g_m]\)

3) *Enrich TLDS*

We use TLDS incremental build a high precision road map to make the planning (e.g. returned to the junction, re-planning, etc.) more stable. We call this is enriching TLDS. Robot take sample according a certain frequency and join the state of sampling to TLDS when it is under normal state. It would be something as follows:

*Child.Visitedwaypoints* = \([\text{Child.Visitedwaypoints, CurrentState}]\)

C. *Road Shape Estimation*

Many feasible and useful method of road shape estimation have been developed by pre-works. The road estimator method of Ref. [7] were used in this paper, which uses an sample importance resample (SIR) Ref. [14] filter to estimate two road curves. We put their method as the first step of road shape estimation in this paper to estimate two curves which are represent the road edge lines. Then, we can use the two curves to calculate an approximation centerline of the road.

An example shown in figure 4 is use to explain road centerline estimation method. The pink dot is the current
position of robot, and the range in the crimson circle represents the local environment perceived by robot. We can sample ten light cyan dots on the two dark red curves which is the real world road edges. We use the average interval sampling method to sample those dots on each curve. The average of those light cyan points is shown as the cyan dots in figure 4, and then we can use the interpolation method or the Bézier Ref. [15] curve to calculate an approximate current centerline of the road (the yellow curve in figure 4). The robot uses the yellow curve as the referential path. Finally, we use the tangent line of the last point on the current road centerline to be the estimation of the future road centerline (the gray dotted line in figure 4).

D. Navigation Point Estimation

The robot path planning does not need an accurate global map, but some global priori information can make robot path planning of high efficiency, and the cost to get this information is relatively low. We can get some waypoints from an electronic map, Google maps for example. In most cases, this kind of waypoints and real values are biased, which need some processing. Lots of pre-work only use the original waypoints which cause some issue. A prominent problem is how to judgment whether successfully traversed a waypoint. Lots of pre-works are to use the original waypoint as the current target, and find a local target between the robot current position and the current target. It is difficult to judgment whether the waypoint is successfully passed and more things need to be considerate which make the planning task very complex. If we make one-one mapping from waypoints to the point on the centerline of the road, the problem will get a very good solution.

Fig.5 This figure shows how one waypoint mapping into a navigation point.

In figure 5, we will have a clear understanding of this method. The pink dot marked p1 is a GPS waypoint which have a deviation with real value and the upper bound of the deviation is known (The first assumption in the first part of the II section). Dotted line marked L3 estimate by the part of road shape estimation in the II section. The blue dot marked p2 is the projection point marked p1 on the centerline of the road which made up of two parts. The first part is a curve estimated by real-time sensors data. The second part is a half-line predicted by curve in the first part. The distance between waypoint and its corresponding map point is used to determine whether the road is feasible or not, which will be described in detail in the remainder part.

If we get the one-one mapping of current waypoint (The projection of the waypoint is update in real-time.), we can use this as a navigation point in robot path planning. We can assume that the robot visited this waypoint successfully, when the robot reaches the projection point of the waypoint.

E. Index Guiding Point (IGP)

We have mentioned index guiding point in the B part of II section (The white dots in figure 3.). In this part, we will elaborate on how to use it. We have to get the IGP before using it. IGP must be able to ensure the robot can drive into the corresponding branch road, when we only use this IGP as a current navigation point in robot path planning.

When the robot faced with a fork in the road, there will be corresponding index guiding points. The number of IGP equals the number of road branch. They are stored in a data storage unit, Leadingpoints, which has be described in part B of the II section. First of all, we can use the current sensors data to estimate the value of each branch road in current fork in the road. In order to guarantee high value can be the first visit, the order of IGP in the Leadingpoints according with the value of corresponding road.

Fig.6 In this figure, we can see how the robot using IGP visit each branch road.

Figure 6 give an example to explain how to use IGP. Robot will get two IGP, g1 and g2, at the blue fork marked f1 in the road. Assumes that the current sensors data indicating the value of g1 is greater than g2, the robot will visit the branch road associated with g1, which is the wrong way. The robot will be back to f1 while the robot knows the way is wrong. Robot will choose the second IGP in the Leadingpoints as the current navigation point. The green line in figure 6 represents the robot is driving and the red line reverse. The kind of storage structure treat the road as a tree-like structure and the branch roads can be accessed, in turn.

F. The Method to Confirm Abnormal State

In the part E of the II section, we have leaving behind a problem that is how to determine that the road is not feasible. There are two kinds of abnormal state, dead end and wrong way. Strictly speaking, dead end is a special case of wrong way. The two line of road edge are intersection within the effectively range of sensors means the robot had a dead end.
We will introduce the method determining a general wrong way in detail.

In the figure 7, we can see two points. The yellow dot marked $p_1$ is both robot current position and the projection of current waypoint. The dark red dot marked $p_2$ is the current waypoint. We know the waypoint error upper bound is known, which has mentioned in the part A of the II section. We can use the distance between current waypoint and its projection to determine whether the road is correct or not. If the distance is greater than a certain threshold, the robot will determine the branch road is wrong. Usually, the value range of threshold is one to two times of the upper bound of the waypoint error.

**G. Overall Framework**

The above, we divide our method into small parts for a better introduce. We will combine them into a complete planning framework at this part. TLDS based path planning framework is shown in figure 8. It is easy to tell the planning framework is composed of global path planning and local path planning. The global path planner will output the current leading point to the local path planner as its current target. The method proposed by us main effect of global path planning. The module of road state determining indicates the current road state, which is the basis of reset, update and enrich (The detailed introduction has carried out in the B part of II.) of TLDS.

**III. RESULTS**

This section, we will give two simulation experiments to verify the feasibility of TLDS, and the complete scheme of path planning based on TLDS. We use Processing that is data visualization software as the simulation platform. First of all, we use abstract road represented by several lines to verify the feasibility of TLDS. Then we use the processed real road map to verify the whole planning scheme.

**A. A Simple Experiment**

All distance space use pixel as a unit. We set the coordinates of the starting point $(20,20)$ (The first element in the brackets is the value of abscissa, the second is ordinate), the goal point is $(480,480)$ and the waypoints are $[(90,70), (180,160), (270,240), (360,340), (450,430)]$. Figure 9 shows the result of this experiment setup. The road was simple abstracted into two lines, $L_1$ and $L_2$. We use this little experiment to verify the effectiveness of TLDS.

Figure 10 shows the result of the experiment with the purpose to validate the effectiveness of TLDS. Figure 10(a) shows the run until a fork in the road. This doesn’t mean the robot will stop processing. Owing to the TLDS, the computational expense is low enough to meet the update frequency of the robot. The movement of the robot is continuous from beginning to end from the vision.

In the figure 10(b), we can see the robot choose a wrong way to move forward which means the robot fall into local optimum caused by the limitation of path planning based on sensors. Because of the global information of waypoints, the robot can find itself was on the wrong way at some point. The red line segment in the lower left corner picture of the Figure 9 shows that the robot using TLDS is able to jump out of the local optimum.

Here it is important to note that from the wrong branch back to the crossroads, when no GPS data is available in the process of robot to complete the task. No GPS data means that we can only use relative localization methods, such as visual odometry Ref. [16] and wheel odometry. The error of localization is accumulating along with the distance the robot driven. The distance run by robot based on local
sensors is longer than the distance of global optimum path. This benefits has been explained clearly in Ref. [2]. The path on the wrong way will not result in the final positioning error accumulation.

The last picture in figure 10, we can see the robot achieve the goal point successfully. It also said TLDS is very effective to the environment with obvious road shape structure. Putting the behavior layer into the planning make the robot software architecture more simply and the robot run more smoothly which can protect the hardware of the robot.

**B. A Complete Scheme of Path Planning**

We use a 1000x500(pixel) map to simulate the real environment which has shown in figure 11. The simulation environment has obvious road shape which has two forks in the road. The green dot in figure 11 is the robot initial state and the red dot is the target point. We assume that the robot is equipped with sensors can perceive the shape of the roads and the sensor’s effective perception range is a limited value which we set 50 pixel value in this simulation. The seven pink dots are inaccurate waypoints which are not on the road.

![Fig.11 A map to simulate the real environment.](image)

We use figure 12 to introduce the concrete simulation process and emphasize the methods used in the simulation, by the way. There are six pictures in figure 12, and each picture has zoom in on the details for facilitate observation.

The area covered by blue disk in figure 12 represent the area in the sensor range of perception. The white dot is current navigation point and the light purple dot is the projection point of the current waypoint. The blue rectangle represents the robot running on the road.

In figure 12(a), we can see the robot leaving the starting point to drive forward along the white navigation point which determined by the current waypoint’s projection on the road and the current sensors data. Robot will through the projection of current waypoint owing to our method which makes the waypoint through decision fairly simple. In figure 12(b), we can see the robot through the projection of current waypoint, which means that the robot will get the next waypoint as the current waypoint.

The robot has sensed a fork in the road shown as a bigger white dot in figure 12(c). At this point, the robot will use the current sensor data to calculate the leading point and a value (The possibility of a correct road.) of each branch road. The robot will choose the branch road with a higher value to explore in it. In this simulation, there are two branch roads in this fork. The robot will select the branch road on its left side as the first try as show in figure 12(d), due to the error from the waypoint to the road. But soon it will be found that this is a wrong way. The robot is driving back to the fork in the road shown in figure 12(e). Instead of U-turn, the robot just need to directly backward drive. The robot will choose a higher value road from the remainder branch road to explore, when it returns to the fork in the road. In the figure 12(f), we can see the robot turn the heading to its right branch road and continue along this road. The robot will use the same method to dispose the next fork in the road.

In this simulation, the robot reaches the target point successfully. Inaccurate way points easy to lead the robot into local minimum. But the method in this paper can make the robot find a right way.

**IV. DISCUSSION**

The experiments demonstrate that TLDS can help the robot jump out the local optimum in the road shape environment. This approach also remains in the simulation
phase, the next step work is mainly to apply this method in practical systems, and verify its effectiveness. Many details, such as fork in the road recognition, need to be considered in the actual system, but this does not affect the overall framework of our method.

We will discuss the question how to use TLDS to path planning in the environment has not road shape. We can also use graph search based storage structure at this condition, but the tree structure will degenerate into graph which is called road map. Ref. [17] give an extremely efficient method of incremental building a road map which is named Probabilistic Road Map (PRM). More research about PRM is in Refs [18-20]. We can use PRM to incremental building a road map for re-planning. Due to the characters of graph, we can use the graph search algorithms (e.g. Dijkstra Ref. [21], A* Ref. [10] and D* Ref. [22]) for re-planning, but the planning algorithm in figure 4 might need do some modification. This will be the future work.

V. CONCLUSION

We presented a data structure, TLDS based on graph search algorithm, to storage the information of the environment to simplify the path planning problems faced by the robot equipped with the sensors which can percept the local information of the environment. We experimentally analyzed TLDS with two simulations, showing effectiveness to global information uncertainty. The first simulation show the TLDS can realize automatic back to re-planning when the robot makes a mistake plan, and finish the task. The second simulation shows the complete scheme can be applied to actual system after add some details.

We also discussed how to use TLDS in path planning task in environment without road shape. The structure can be extended to the tool that can help the robot to explore in unknown environment. We will consider it in future work.

ACKNOWLEDGMENT

This work is supported by National Nature Sciences Foundation of China (Grant No.61305121, 61433016) and The state key laboratory of robotics (2015-z02).

REFERENCES