A New Hybrid Method for Mobile Robot Dynamic Local Path Planning in Unknown Environment

Peng Li, Xinhan Huang, Min Wang
Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
Email: lipeng_hubei@126.com

Abstract—In this paper, a hybrid approach for efficiently planning smooth local paths for mobile robot in an unknown environment is presented. The single robot is treated as a multi-agent system, and the corresponding architecture with cooperative control is constructed. And then a new method of information fusion namely DSmT (Dempster-Shafer Theory) which is an extension of the DST (Dempster-Shafer Theory) is introduced to deal with the error laser readings. In order to make A* algorithm suitable for local path planning, safety guard district search method and optimizing approach for searched paths are proposed. Also, the parameters of internal Proportional-Integral-Derivative (PID) controller in the goto agent are adjusted through practical experiments for the use of smoothing the path searched by optimized A* algorithm. Finally, two kinds of experiments are carried out with Pioneer 2-DXe mobile robot: one uses the hybrid method proposed in this paper, the other uses artificial potential field (APF) which is the classical algorithm for local path planning. The experimental results reveal the validity and superiority of the hybrid method for dynamic local path planning. The approach presented in this paper provides an academic support for path planning in dynamic environment with moving objects in the field.

Index Terms—path planning, multi-agent, Dezert-Smarandache Theory, A* algorithm, mobile robot.

I. INTRODUCTION

Mobile robot path planning is one of the most important topics in robotics research. Point-to-point path planning of autonomous mobile robot, defined as finding a collision-free path linking a given start configuration to a goal configuration, has been extensively explored in last two decades. Many different methods achieving varying degrees of success in a variety of conditions/criteria of motion and environments have been developed.

Path planning for mobile robot is composed of two main parts: the global path planning and the local path planning. For the global path planning, the entire environment is known for the robot, so that the robot only needs to compute the path once at the beginning and then to follow the planned path up to the target point. Oppositely for the local path planning, the robot only knows the area which has been detected itself, it only casually decides the direction to move. There are many studies on robot path planning using various approaches, such as the grid-based A* algorithm [1], road maps [2], cell decomposition [3], and artificial potential field (APF) [4]. Some of the previous approaches use global methods to search the possible paths in the workspace [5], normally deal with static environments only, and are computationally expensive when the environment is complex. Some methods suffer from undesired local minima; the robots may be trapped in some cases such as with concave U-shaped barriers. But most studies mentioned above mainly focus on global path planning and there is few valid method for local path planning.

This paper presents an efficient hybrid approach for real time collision-free path planning and grid method is adopted to depict the environment map. In traditional path planning studies, the robot is always treated as a single unit. But here we treat the robot as a multi-agent system, such as path planning agent, behavioral agent and perception agent. In this system, agents can be complex entities at the same time. Each agent achieves its task and collaborates with other agents for the same purpose—to find a rational path. When the robot is commanded to reach an appointed target, in brief, the path planning agent calculates a temporary path with limited knowledge about the surroundings with the A* algorithm and the behavioral agent smoothes the planned path using its goto agent. The error readings are wiped off by filter based on DSmT [6] which is extended from DST [7]. And the path is revised once robot finds an obstacle block the path. In the past, A* algorithm is completely used in global path planning; here it is proved to be a valid algorithm for local path planning, too. And it provides an academic support for path planning in dynamic environment (some objects moves in the environment).

II. MULTI-AGENT ROBOT SYSTEM ARCHITECTURE

It is established that multiple cooperative control in a single robot involves cooperative control and multi-agent systems. Cooperative control has a very general meaning, any complex system developed with multi-agent architectures may be considered as a cooperative approach.

The multi-agent robot system can be divided into five subsystems of agents: perception, behavior, path planning, localization and actuator (Fig.1). The behavioral agent subsystem includes goto agent and avoid agent. In
addition to all of the above, there is a client agent acting as user interface. Fig.1 also depicts the flow of information among different agents.

The perception agent obtains information about the environment and about the internal conditions of the robot. Of course, it includes an error reading filter based on DSmT. It collects data from the sensors and after getting rid of error readings, it adapts the data to provide the information requested by the other agents of the system. For example, the milemeter is in charge of obtaining the coordinates \((x, y)\) of the robot and its orientation, with reference to a fixed axis; the laser sensor collects all the laser readings of the robot.

The perception agent, sets the first target position to the goto agent. Based on this information and the actual position (obtained from the localization agent), the goto agent calculates the best linear and angular speeds to reach the target. On the other hand, based on the information provided by the localization agent and laser agent, the avoid agent calculates the linear and angular speeds to dodge the obstacle. At this point both agents (goto and avoid) negotiate in order to decide who uses the motors. But usually the avoid agent does not need to work because the path planning has find a collision free path for robot except accidents, for example, an object abruptly appears in front of the robot and the path planning yet has not calculated the new path for current situation.

The one that wins sends the desired speeds to the actuator agent. And then perception agent obtains the laser and the milemeter readings and sends them to the localization agent, correspondingly. With this new information all the agents update their internal state and new decisions can be taken. Once the target position sent by the path planning agent is reached, the next target position will be sent to the goto agent.

The robot’s status can be monitored by client agent. This agent depicts a real-time global map and controls the robot according to the information transmitted from the robot via radio network.

III. ERROR READING FILTER BASE ON DSmT

A. Simple review of DSmT

The DSmT of plausible and paradoxical reasoning proposed by the authors in recent years allows to formally combine any types of independent sources of information represented in term of belief functions [8, 9]. And it is mainly focused on the fusion of uncertain, highly conflicting and imprecise sources of evidence. DSmT is able to solve complex static or dynamic fusion problems, especially when conflicts between sources become large and when the refinement of the frame of the problem under consideration, denoted \(\Theta\), becomes inaccessible because of the vague, relative and imprecise nature of elements of \(\Theta\).

**Notion of hyper-power set \(D^\Theta\)**: One of the cornerstones of the DSmT is the notion of hyper-power set. Let \(\Theta = \{\theta_1, \ldots, \theta_n\}\) be a finite set (called frame) of \(n\) elements. The hyper-power set \(D^\Theta\) is defined as the set of all composite propositions built from elements of \(\Theta\) with \(\cup\) and \(\cap\) operators such that:

- \(a) \quad \phi, \theta_1, \ldots, \theta_n \in D^\Theta\).
- \(b) \quad \text{If } A, B \in D^\Theta, \text{ then } A \cap B \in D^\Theta \text{ and } A \cup B \in D^\Theta\).
- \(c) \quad \text{No other elements belong to } D^\Theta, \text{ except those obtained by using rules } a) \text{ or } b)\).

**Generalized belief functions**: We define a map from a general frame \(\Theta \ m(.) : D^\Theta \rightarrow [0,1]\) associated to a given source, say B, of evidence as

\[
m(\emptyset) = 0 \quad \text{and} \quad \sum_{A \in D^\Theta} m(A) = 1
\]
The quantity \( m(A) \) is called the \textit{generalized basic belief assignment (gbba)} of \( A \).

The generalized belief and plausibility functions are defined as

\[
\text{Bel}(A) = \sum_{\beta \in A} m(B)
\]

\[
\text{Pl}(A) = \sum_{\beta \in D^\theta} m(B)
\]

The Classic DSm Rule for Free-DSm Model: For \( k \) independent uncertain and paradoxical sources of information providing generalized basic belief assignment \( m_i(\cdot) \) over \( D^\theta \), the classical DSm conjunctive rule of combination \[9\] \( m_{\bigcup(i)}(A) \) is given by

\[
\forall A \neq \phi \in D^\theta, \quad m_{\bigcup(i)}(A) \bigcup [m_i(\Theta \cap \Theta \cup \Theta_1)](A) = \sum_{X_i \cap \Theta \subseteq \Phi} \prod_{i=1}^{2} m_i(X_i)
\]

\( m_{\bigcup(i)}(A) = 0 \) by definition, unless otherwise specified in special cases when some source assigns a non-zero value to it.

\[\text{A. Error reading filter base on DSmT}\]

Here the \( \Theta \) is defined as the status of each grid cell on the map which is constructed by the robot. Suppose there are two elements \( \Theta_1 \) and \( \Theta_2 \) in the frame of discernment \( \Theta \). \( \Theta_1 \) means the reading is wrong and \( \Theta_2 \) is defined as right. The hyper-power set is \( D^\Theta = \{ \phi, \Theta \cap \Theta_2, \Theta_1, \Theta_2, \Theta_1 \cup \Theta_2 \} \). Then we define \( m(\Theta_j) \) as the general basic belief assignment function (gbba) for right status; define \( m(\Theta_1 \cup \Theta_2) \) as the gbba of the wrong status; \( m(\Theta_1 \cap \Theta_2) \) is defined as the gbba of conflict mass, it is generated during the fusion; and \( m(\Theta_1 \cup \Theta_2) \) is defined as the gbba of unknown status (it mainly refers to those areas that still not be scanned at present, so the \( m(\Theta_1 \cup \Theta_2) \) of detected areas is zero and does not need join in the fusion).

The laser sensor can detect the 180° area in front of the robot. Each time it can get no more than 180 readings. It is proved that if the surrounding is roomy the amount of readings can reach nearly 180; oppositely if narrow the robot maybe can only get about 150 readings. There are two evidence sources for the filter. The first source is from the readings themselves. When the robot get a new group of laser readings, each reading is compared with its two neighbor readings (left reading and right reading) except the first and the last reading (because the first one and the last one only have one neighbor reading). For example, if a reading \( R \) is being checking, the left reading is defined as \( R_L \) and the right reading is \( R_R \), if \( R \cap R_L = R_L \cap R_R \) or \( R \cap R_R = R_L \), these two situations (shown in Fig.2) mean that the reading \( R \) probably is a wrong reading and it need to be fused with the second source which is mentioned later. Other situations mean that \( R \) is a right reading and does not need further fusion.

The belief assignments of the first source \( m_1(\cdot) \); \( D^\Theta \rightarrow [0,1] \) are constructed by authors as follows:

\[
R_e = |R_L + R_R - 2R|/2; \quad (1)
\]

\[
R_{max} = \max\{|R_L, R_R, R|; \quad (2)
\]

\[
m(\Theta_1) = \exp[-5 \times (R_e/R_{max})^2]; \quad (3)
\]

\[
m(\Theta_2) = 1 - m(\Theta_1); \quad (4)
\]

The Fig.3 shows that the smaller the \( R_e \) is, the bigger the general basic belief assignment (gbba) of \( \Theta_1 \) becomes, and the gbba of \( \Theta_2 \) is opposite.

The second evidence source is from the map which has been built by the robot. If a laser reading is in one of the two above mentioned error reading situations, the \( m(\cdot) \) calculated from the first source will be fused with the \( m(\cdot) \) of second source. The robot checks the area around of the reading point which is marked as the potential error reading. The area is a rectangle with 5×5 grids, and the reading point is the center grid. The \( m(\cdot) \) of the reading point is calculated according to the amount of occupied grid in this area. Suppose \( N \) is the amount of occupied grid in this area, the \( m(\cdot) \) of the second source is computed as:

\[
\text{Figure 2. Two situations of error readings.}
\]

\[
\text{Figure 3. m(\cdot) of Eq.(1-4).}
\]
\[ m_2(\Theta_2) = 1 - m_2(\Theta_1); \]

The fusion between \( m_1(\cdot) \) and \( m_2(\cdot) \) follows the combining rules of Proportional Conflict Redistribution Rule 2 (PCR2) \[10\] under the framework of DSmT. The PCR2 formula for \( k \geq 2 \) sources is:

\[ \forall (X \neq \emptyset) \in D^{m}, m_{PCR2}(X) = \sum_{i} \prod_{j} m_i(X_j) + C(X) \frac{\sum_{s} c_{12...s}(X)}{e_{12...s}} \]

Where

\[ k_{12...s} = \sum_{X_i \cap X_j \neq \emptyset} \prod_{j=1}^{n} m_i(X_j) \]

\[ C(X) = \begin{cases} 1, & \text{if } X \text{ involved in the conflict,} \\ 0, & \text{otherwise;} \end{cases} \]

In this formula, \( c_{12...s}(X) \) is the non-zero sum of the column of \( X \) in the mass matrix, \( k_{12...s} \) is the total conflicting mass (here it is \( m(\Theta_1 \cup \Theta_2) \)), and \( e_{12...s} \) is the sum of all non-zero column sums of all non-empty sets involved in the conflict.

After fusion, the final gbba of \( \Theta_1 \) can be calculated and if \( m_2(\Theta_1) \geq 0.8 \) that means the laser reading which is being checked is a right reading instead of an error reading.

IV. PATH PLANNING METHOD

A. Path planning agent

Method of path planning agent: The robot plans the path through path planning agent. After updating the environmental information, the robot firstly makes use of A* algorithm to re-calculate the path if need. The path from current location to the target point is decomposed into a series of turning goal point. And then the real path between every two goal points is smoothed by internal PID controller.

The A* algorithm is usually used for static global path planning. There are few applications for dynamic local path planning in real time because of its large amount of calculations. In this paper, a simple and very effectual method is proposed to make the A* suitable for dynamic local path planning in real time. In most path planning studies, the robot is abstracted as a point without acreage. But actually the robot’s radius must be considered in practical applications. The area near the planned path is the safety guard district. That means this area must be empty or the robot cannot pass. The safety guard district is shown in Fig.4.

Once the robot gets a new group of laser readings, each reading is checked to make sure whether it is in the safety guard district. If a reading is in this area, it means an obstacle blocks the way and the path must be re-calculated. Oppositely, if there is none in the area, well then the path does not need to be re-calculated. This method reduces the computations of path planning. The robot only needs to calculate the path occasionally.

Optimized A* algorithm: The idea of A* algorithm is that each node is associated with a cost function

\[ f(n) = g(n) + h(n), \]

where \( g(n) \) is the cost from initial node to the current node and \( h(n) \) is an estimated cost from the current node to the goal node. A* SEARCH generates and processes the successor nodes in a certain way. Whenever it looks for the next node to process, it employs heuristic function \( h(n) \) trying to choose the lowest cost node to process. The following algorithm summarizes A* algorithm:

1. Put the start node \( s \) on a list called OPEN and compute \( f(s) \).
2. If OPEN is empty, exit with failure; otherwise continue.
3. Remove from OPEN that node whose \( f \) value is smallest and put it on a list called CLOSED. Call this node \( n \). Resolve ties for minimal \( f \) values arbitrarily, but always in favor of any goal node.
4. If \( n \) is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise continue.
5. Expand node \( n \), generating all of its successors. If there are no successors, go immediately to 2. For each successor \( n_i \), compute \( f(n_i) \).
6. Associate with the successors not already on either OPEN or CLOSED the \( f \) values just computed. Put these nodes on OPEN and direct pointers from them back to \( n \).
7. Associate with those successors that were already on OPEN or CLOSED the smaller of the \( f \) values just computed and their previous \( f \) values. Put on OPEN those successors on CLOSED whose \( f \) values were thus lowered, and redirect to \( n \) the pointers from all nodes whose \( f \) values were lowered.

Go to 2.

The A* algorithm relies heavily on heuristic function. An appropriate heuristic function determines whether the algorithm can execute efficiently and accurately. In order to find an optimal solution, the heuristic must be admissible. To be admissible, the heuristic function must
never over-estimate the cost from one node to the goal node.

![Diagram](image)

Figure 5. Sketch map of optimizing point-path.

The path searched by A* algorithm is a group of continuous goal points. If the grid is too small, it will spend a large of memory to store the point-path; and if the goal points are placed too closely, the robot yet cannot follow the path well because of the limit of its turning radius. So here the point-path is optimized. The goal points on the same line are deleted and then the robot only needs to store the turning goal points (include start point and end point). This method markedly reduces the memory for storing point-path. The sketch map of optimized point-path is shown in Fig.5.

B. Goto agent

It is known that the turning angle of the path calculated by A* algorithm in the grid map is 45° or 90°. So the path is not smooth and sometimes the robot cannot follow the trajectory because of these stark turnings. The goto agent can solve this problem. The robot’s walking between two neighbor goal points is under the charge of goto agent.

Architecture of goto agent: The input of goto agent is the target goal point \((x_i, y_i, \theta_i)\) and the output is a group of control parameters \((v, \omega)\). \(v\) is the robot’s velocity and \(\omega\) is angular velocity for turning. The architecture of goto agent is shown in Fig.6. The variable \(d\) in Fig.6 is the distance from current robot’s position to the target goal point.

![Diagram](image)

Figure 6. Architecture of goto agent.

**Internal PID controller:** The robot uses a common Proportional-Integral-Derivative (PID) control system to adjust the PWM pulse width at the motor drivers and subsequent power to the motors.

The P term value \(K_p\) increases the overall gain of the system by amplifying the position error. Large gains will have a tendency to overshoot the velocity goal; small gains will limit the overshoot but cause the system to become sluggish. It is found that a fully loaded robot works best with a \(K_p\) setting of around 15 to 20, whereas a lightly loaded robot may work best with \(K_p\) in the range of 20 to 30.

The D term \(K_d\) provides a PID gain factor that is proportional to the output velocity. It has the greatest effect on system damping and minimizing oscillations within the drive system. The term usually is the first to be adjusted if robot encounters unsatisfactory drive response. Typically, it is found that \(K_d\) works best in the range of 600 to 800 for lightly to heavily loaded robots, respectively.

The I Term \(K_i\) moderates any steady state errors thereby limiting velocity fluctuations during the course of a move. Too large of a \(K_i\) factor will cause an excessive windup of the motor when the load changes, such as when climbing over a bump or accelerating to a new speed. Consequently, this study typically uses a minimum value for \(K_i\) in the range of 0 to 10 for lightly to heavily loaded robots respectively.

V. EXPERIMENTS

A user interface as a software platform for experiment is developed by authors with Visual Studio 2008. And Pioneer 2-DXE mobile robot which is shown in Fig.7(a) is used in experiments. Two kinds of experiments are carried out: one uses the hybrid method proposed in this paper, the other uses artificial potential field (APF) which is the classical algorithm for local path planning.

An experiment field (size: 4840×3100 mm) is created as Fig.7(b) and the real world of experiment is as Fig.7(c). The point of robot I is treated as the coordinate origin of the global map. So robot is set to the pose of (0,0,0°). The third parameter is the deflection angle of robot. And the target goal position is placed near to the right top corner.

![Robot](image)

(a) Pioneer 2-DXE mobile robot.
A. Hybrid method experiment

The effect of error reading filter: The effect of error reading filter is verified before the path planning experiment. The striking dissimilarity between mapping without filter and mapping with filter is revealed clearly in Fig.8. If there is no filter, the robot cannot find the way to the target goal because so many barrier-points block the way that there is not enough space to pass.

Path planning experiment: In order to distinguish the planned path and the final real path, the planned path is displayed during walking and the real path is displayed when the robot arrives at the target goal point. The experimental result is shown in Fig.9. The circles in Fig.9 are the turning goal points calculated by optimized A* algorithm.
B. APF experiment

The error reading filter is still used in this experiment for its important effect. The start position and the target goal position is the same as in the hybrid method experiment. Of course in this experiment the robot is not a multi-agent system any more. For this experiment does not need the behavioral agent subsystem. And then the information flow becomes unilateral. There are no reciprocities between the parts in the robot.

The APF experimental result is shown in Fig.10. The trajectory in the center is robot’s real path. The fig.10(h) shows the final path of the robot.

C. Analysis

Through these two experiments, it is obviously that the hybrid method proposed in this paper is a very effectual
method for dynamic local path planning. The experimental results of these two experiments clearly show:

1) In hybrid method experiment, the robot only needs to store several turning goal points. The planned path is changed once the robot finds new barriers block the way; this is expressly shown in Fig.9(c–d).

2) The goto agent performs so perfectly that the real path of hybrid method experiment is very smooth instead of stark turnings which is the fault of A* algorithm.

3) Actually, the start position of the robot is a concave U-shaped trap. In hybrid method experiment, the robot can easily walk out this area and the path is rational. But in APF experiment, the robot is trapped in local minima; it repeatedly follows the same trajectory and cannot get out the repetition.

4) The effect of error reading filter based on DSmT is valid. The maps built with the laser readings are clean and accurate.

5) In hybrid method experiment, the Safety guard district search method reduces the computation, so that the whole system works well without lags even crash. The experiment is carried out well.

VI. CONCLUSIONS

This paper has proposed a hybrid approach for planning smooth paths satisfying dynamic local constraints for robot in an unknown environment. The single robot is divided into several synergic agents; among them the most important agents for path planning are path planning agent and behavioral agents since their cooperation influences directly the final result. This paper also presents a valid error reading filter based on DSmT. And then in order to make A* algorithm suitable for dynamic local path planning, safety guard district search method and optimizing approach for searched path are proposed. The parameters of internal PID controller in the goto agent are adjusted through practical experiments. The results of experiments carried out with Pioneer 2-DXe mobile robot prove the validity and superiority of the hybrid method for dynamic local path planning. The application of the approach presented in this paper to path planning in completely dynamic unknown environment with moving objects around the robot will be investigated in the future.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under grant No. 60675028, which is greatly acknowledged.

REFERENCES


Peng Li, born in 1981. Since 2006, he has been a Ph.D. candidate in Department of Control Science and Engineering of Huazhong University of Science and Technology (HUST), Wuhan, China. His main interests include information fusion, robot’s map building and localization, multi-robot system.

Xinhan Huang, born in 1946. He is currently a Professor, Ph.D. candidate supervisor of Department of
Control Science and Engineering and Head of the Institute of Intelligent Control of HUST. His research interests include robots and their intelligent control, multi-sensor data fusion, pattern recognizing.

**Min Wang**, born in 1954. She is currently a Professor of the Department of Control Science and Engineering of HUST. Her research interests include robotics, intelligent control, image processing, pattern recognizing.