A Goal-Oriented Approach to Robot Way-Finding in Environments with Local Dead-Ends

Omid Motlagh¹, S. H. Tang², N. Ismail³
Mech. and Manf. Engg. Dept., University Putra Malaysia, Serdang, Selangor, Malaysia

Abstract - A new behavior-based method is proposed for way-finding of a mobile robot in unknown environments with dead-ends. A standard fuzzy logic (FL) algorithm directs the robot towards its target i.e.; actual or virtual, while avoiding obstacles by means of steering and velocity control. At top of the FL controller, a switching strategy is developed for virtually redirecting the target when the robot gets trapped in local dead-ends. The target shifting algorithm proved to direct the robot out from any type of traps e.g.; barriers forming simple corners, U-shape, snail shape, maze, and other complicated concave dead-ends. Trajectory results of a Pioneer robot are demonstrated using ActiveMedia Robotics simulator, and compared to other related works to prove the robustness of the proposed algorithm.

Keywords: Mobile robot; Virtual target; Local minimum.

1 Introduction

Real-time local path planning in unknown and dynamic environments with uncertainties is one of the most challenging problems in robotics. It requires capability of sensing the environment, interpreting the information to obtain the knowledge of position and the environment, planning a real-time route from an initial position to a target with obstacle avoidance, and controlling the robot direction and velocity to reach the target [1]. Some of the approaches to path planning include wall following [2], artificial potential field [3], [4], virtual target approach [5], [6], landmark learning [7], virtual wall algorithm [8], edge detection, memory grid [9], graph-based methods [10], vector field histogram, dynamic window, neural network, fuzzy logic [11], [12], and many others. Since introduction of fuzzy logic control (FLC) by Lotfi A. Zadeh of the University of California at Berkeley [13], this approach to mobile robot navigation and obstacle avoidance has been investigated by several researchers. Fuzzy systems have the ability to treat uncertain and imprecise data using linguistic rules. They offer possible implementation of human knowledge and do not require a precise analytical model of the environment.

1.1 The problem of limit cycles

Despite the great advantages of FLC in local navigation, obstacles forming a loop shape or dead-end traps are its most common causes of failure. The local minimum situation occurs when a robot navigating past obstacles towards a target with no prior knowledge of the environment gets trapped in a loop. This happens if the environment consists of concave obstacles, and the like. Fig.1 shows a robot with pure fuzzy logic navigator getting trapped in a U-shape dead end. Here, rules that are fired for target attractor and obstacle repulsor modules give output actions that neutralize each other. Therefore the robot gets into an infinite loop or local minimum [7].

Initially the robot moves directly toward the target due to target seeking behavior until point “A” where it detects an obstacle at direct front. Then it makes a right turn due to obstacle avoidance behavior which results to wall following until point “C”. This is because until this point, both the target and obstacle are at the left hand side of the robot. But as the robot is passing by point “C” the target is going to be at right hand side of the robot, while the obstacle is still at the left hand side. Therefore at point “D”, the robot goes back toward the target due to both target seeking and obstacle avoidance behaviors. The result is that the robot wanders indefinitely in the dead-end trap that is called limit cycle problem. In the most related works in the area of local navigation, the limit cycle problem has been investigated and several effective solutions have been developed.
2 Background

In the memory state method of Zhu, and Yang [11], the limit cycle problem is resolved by using a state memory strategy (Fig.2). The variables from which this method makes ultimate decisions i.e.; to keep turning around the obstacle (states 1 or 2) or to reach the target (state 0), are the robot current distance to the target (Dc) and the robot initial distance to the target (Dm). These distance variables were memorized when the obstacle was detected for the first time (first change from state 0 to 1 or 2). But this distance-based decision making results in poor trajectories in situations like from point “D” toward the target. Since Dm was memorized when the robot was traveling from “A” to “B”, and the condition for changing the state to 0 (target seeking) is \( Dc \leq Dm \), therefore from point “D” the robot can not go straight toward the target. It has to keep turning around the obstacle until point “E” where Dc becomes shorter than Dm.

The method of Krishna, and Karla [7] is a real-time collision avoidance algorithm with the local minimum problem resolved by classifying the environment based on the spatio-temporal sensory data sequences. Although this method has a good result in Fig.3 (a), it highly depends on the landmark recognition and therefore needs exact coordination localization. In addition, it is difficult to choose a correct way to follow the wall boundary as shown in Fig.3 (b) [12].

The minimum risk method of Wang and Liu [12], is based on trial-return behavior phenomenon shown in Fig.4 (a) and (b). Therefore when the nearest exit is blocked by a long wall, the nearest exit will be the opening at the right hand side where the wall ends. Same principle makes the robot quit the dead end of Fig.4 (b).

The minimum risk approach seems robust in traps of this type. However the problems with trial-return motion are; high power consumption and the time spent from start to target. These are the two important issues that are totally ignored in this approach. This is shown in Fig.5 with an example of a mobile robot struggling to find its way out of a simple dead end.

The method of Xu, and Tso [5], has good properties in minimum avoidance (Fig.6). The target is virtually relocated to points directed away from the true target. Therefore after passing through the critical points “C” or “F”, the robot maintains the same tendency of turning to the left from C → I → J → K or to the right from F → L → M → N. The virtual target orientation is given by \( \theta_r = - (\pi - \theta_{r0}) \) following the point “C” or \( \theta_r = \pi - \theta_{r0} \) following the point “F” where “\( \theta_{r0} \)” and “\( \theta_r \)” represent the real and virtual target orientations respectively.

The condition for target switching is given by \( |\theta_{r1} - \theta_{r0}| > TR \), where “\( \theta_{r0} \)” and “\( \theta_{r1} \)” are the real target orientations at the two consecutive reaction instants, and TR is a threshold for the abrupt change in the target orientation. Due to the actual to virtual target switching, the robot avoids limit cycles. Even in a loop shape like in Fig.7 (a), still robot can find the opening at point “C” due to the fuzzy logic target seeking and obstacle avoidance behaviors. This method shows its major weakness when it
fails to reach the goal in such a concave and recursive U-shape environment shown in Fig.7 (b). “This is because the robot encounters another local minimum at locations B and C when it is still working under the influence of previous virtual sub goal [location A]” [12].

First at point “A”, a counter-clockwise shift of the target makes the robot follow the front wall until point “B” where another wall is encountered. At point “B” because of presence of the two walls at both front and right, the target again will be gradually shifted 90 degree to left and towards point “C”. Therefore the robot keeps following the circular motion of the virtual target. The counter-clockwise shift of the target results in wall following behavior of the robot until the wall ends and the robot detects the opening at point “D”.

At this point the robot must decide to either switch back towards the real target or to keep following the walls. Here, if the sum of turning angles throughout the way is near to 0 degree, the robot will decide to go toward the real target. If the total amount of turning angles is positive, this means the robot generally has had a clockwise motion, and therefore it has to compensate it with a counter-clockwise motion until the sum becomes equal to zero. Similarly for negative amounts at the opening point, a clockwise compensatory motion is required.

For the example shown in Fig.8, until the opening at point “D”, the robot has had a total of -360 degrees turning angles. Therefore, instead of going towards the real target which results in limit cycles, the robot turns to right due to the virtual target shift, and continues to wall following. This way the robot turns 180 degrees clockwise at the opening (point “D” towards “E”), and another 90 degrees clockwise from point “E” to point “F”. The last compensation is done in turning right at point “F” where the total of the clockwise turning angles becomes equal to +360 degrees. In other words the sum of turned angles becomes zero. Therefore at point “F” where the robot detects the opening, it switches back from the virtual target in a straight direction towards the real target direction on the right and through point “I” straight goes until reaches the target.

3 Smooth target switching strategy

In this method, unlike other virtual target approaches, target switching is a smooth clockwise or counter-clockwise shift from the actual target direction to a direction tangent to the wall to be followed. Therefore the robot turns and follows the walls inside the dead-end until escapes the trap. The target switching approach adopted here has no conflict with the fuzzy controller which is controlling the two main tasks of obstacle avoidance and target seeking regardless to the nature of the target i.e.; actual or virtual. Fig.8 shows an overall performance of the robot under proposed algorithm in an ideal situation i.e.; ignoring wheel slippage, and sonar errors.
3.1 Integration of FLC and target switch conditions

Fig. 9 shows the algorithm structure consisting of two layers of FLC and conditional reasoning. Basically a fuzzy logic controller makes decisions for the amounts of turning angle, the robot speed, and also the next target shift. After the robot makes turn, the amount of actually turned angle will be added to the sum of the turning angles so far the robot has turned. The sum of turned angles then will be fed back to the upper layer which makes the decision for target shifting. The decision will be either to go towards the next virtual target, or to shift the virtual target towards the actual one by compensating in the opposite direction, or to immediately switch back to the actual target. The input data to the algorithm are from sonar readings, as well as the robot’s self positioning and localization sensors i.e.; wheel counters, and compass for X, Y coordinates and heading angle measurement.

3.2 The fuzzy logic controller

Like most of other algorithms which are based on FL potential fields, the FL navigation strategy adopted here is to make a compromise between target reaching behavior i.e.; the target attracting the robot, and obstacle avoidance behavior i.e.; the obstacles repelling the robot. This standard fuzzy logic controller (in contrast with preference-based structure [14]), is applied to coordinate the two behaviors of the robot in wayfinding in unknown environments. Therefore, at the lower layer of the navigation algorithm, the FL controller designates obstacle positions as repulsive potentials and the goal position as an attractive potential. It must be noted that the FL controller does not care about nature of the target whether it is actual or virtual target. It just makes the robot move towards the goal while avoiding obstacles.

By dividing the work space (S) into six subspaces with over laps, the fuzzy rules can be defined accordingly. The subspaces for obstacle position are chosen as; right (R), right-front (RF), front (F), left-front (LF), and, left (L), each with 45 degrees of overlap with the next one, and, finally no-obstacle. Also there are three subspaces for the target orientation namely; right (ur), front (uf), and, left (ul). Therefore there are eighteen if-then rules generated by which the FL controller makes decision for the amount of outputs which will be the amount of change to the robot’s speed, and heading angle.

Fig. 10 shows how the robot space can be divided into the six sub-spaces just mentioned. The array of eight sonar sensors (S0-S7) are shown on the robot Circumference at -90, -50, -30, -10, +10, +30, +50, +90 degrees. For more accuracy in obstacle detection in the simulation software, the maximum radius of sight for reliable obstacle detection is set to 30 cm. Therefore the robot can not detect obstacles further than this distance which is considered as “no-obstacle” sub-space.
Fig. 11 depicts the robot in the environment. The inputs to the fuzzy controller are the obstacles orientation angle relative to the robot heading \( \theta \), and the target orientation which is defined as the angle between the robot heading direction and the robot-to-target direction \( u \). These two angles at any time are obtained from sensors from which the robot perceives its environment. The outputs are the steering angle \( Y_t \), and robot linear velocity \( V \) to be applied to the robot’s wheel actuators.

### 3.2.1 Input and output membership functions

The input and output functions are normalized to \([0-1] \). The amounts of overlap between the regions are decided according to the position of the sensors. The 8 sonar sensors are grouped as ; left sensors \((S0, S1)\), left-front \((S0, S1, S2, S3)\), front \((S2, S3, S4, S5)\), right-front \((S4, S5, S6, S7)\), and finally right sensors \((S6, S7)\). The FLC operation includes namely ; fuzzification, inference, aggregation, and defuzzification. The fuzzification converts the input variables into input grades by means of the membership functions shown in Fig. 12. The inference and aggregation generate a resultant output membership function with respect to fuzzy rules, and finally the defuzzification finds the center of gravity of the output membership functions as the robot steering angle and speed. Fig. 13 shows the output membership functions.

![Fig. 11](image1.png)

An obstacle is detected at right front

![Fig. 12](image2.png)

Input membership functions. (a) Actual-virtual target direction and (b) Obstacle position, both relative to the robot current heading angle

### 3.2.2 Defining the fuzzy rule set

Robot navigation is actually governed by rules in the fuzzy controller which represents qualitatively the human driving heuristics. Each of the rules gets a value according to position of target and obstacle at any time. For example if an obstacle is exactly at right front of the robot, and the target is exactly in front of the robot, then \( RF = 1 \) and \( ur = 1 \) will be obtained from the input membership functions in Fig. 12. Therefore the rule \#2 gets the amount of 1 (rule \#2= \( ur \times RF \)), while the others get 0. The fuzzy rules are given in Table 1 where \( ur, uf \) and \( ul \) are the weights of the target at right, front and left. For the obstacles they can be either at right, right-front, front, left-front or at left all relative to the current heading angle of the robot. In case there is no obstacle around, the state will be “NO” or “No Obstacle”. Since the target (actual or virtual) takes 3 states and the obstacle takes 6 states therefore 18 rules have been developed.

Another difference made in this method is the very small number of rules. This has the advantage of short execution time and high frequency of sensor readings. This is especially important when the environment is dynamic and fast response to changes is required.

![Table 1](image3.png)

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Current virtual target direction</th>
<th>Obstacle Position</th>
<th>THEN: Steering direction</th>
<th>Robot velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ur &amp; R</td>
<td></td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>ur &amp; RF</td>
<td></td>
<td>LF</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>ur &amp; F</td>
<td></td>
<td>R</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>ur &amp; LF</td>
<td></td>
<td>R</td>
<td>N</td>
</tr>
<tr>
<td>5</td>
<td>ur &amp; L</td>
<td></td>
<td>R</td>
<td>H</td>
</tr>
<tr>
<td>6</td>
<td>ur &amp; NO</td>
<td></td>
<td>R</td>
<td>H</td>
</tr>
<tr>
<td>7</td>
<td>uf &amp; R</td>
<td></td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>uf &amp; RF</td>
<td></td>
<td>LF</td>
<td>L</td>
</tr>
<tr>
<td>9</td>
<td>uf &amp; L</td>
<td></td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>10</td>
<td>uf &amp; LF</td>
<td></td>
<td>RF</td>
<td>L</td>
</tr>
<tr>
<td>11</td>
<td>uf &amp; NO</td>
<td></td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>12</td>
<td>uf &amp; L</td>
<td></td>
<td>F</td>
<td>H</td>
</tr>
<tr>
<td>13</td>
<td>ul &amp; R</td>
<td></td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>14</td>
<td>ul &amp; RF</td>
<td></td>
<td>L</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>ul &amp; F</td>
<td></td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>16</td>
<td>ul &amp; LF</td>
<td></td>
<td>RF</td>
<td>L</td>
</tr>
<tr>
<td>17</td>
<td>ul &amp; L</td>
<td></td>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>18</td>
<td>ul &amp; NO</td>
<td></td>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>
Defuzzified output for turning angle can be obtained from Eq. (1) where \( c_r, c_{rf}, c_f, c_{lf}, \) and \( c_l \) are constants of the output membership function in Fig. 13 (a) whose values take +20, +10, 0, -10 and -20, respectively. Although these constants are adjustable during programming, the given amounts were proven to be more efficient through experimental results and by considering the robot’s actual limits for steering. Another alternative to improve the membership functions would be optimization of their variables using any optimization algorithm. For example in a GA-based method proposed in [15], 10 bits of the chromosome were utilized to represent the membership function distributions for each of the variables.

\[
\text{Turning Angle } (Y) = \frac{c_r \sum R + c_{rf} \sum RF + c_f \sum F + c_{lf} \sum LF + c_l \sum L}{\sum R + \sum RF + \sum F + \sum LF + \sum L}
\]  

(1)

Similarly, defuzzified output for changes of the robot velocity \((\Delta V)\) can be obtained from Eq. (2) where \( v_h, v_n, \) and \( v_l \) are constants of the output membership function in Fig. 13 (b) whose values take +50, 0, and -50, respectively. During simulation work in the proposed method, the robot normal speed \( V_N \), was set to 50 mm/sec for highest efficiency and with respect to speed limits and power consumption. However this parameter can be optimized along with others by using evolutionary algorithms.

\[
\text{Robot Velocity } (\Delta V) = \frac{v_h \sum H + v_n \sum N + v_l \sum L}{\sum H + \sum N + \sum L}
\]  

(2)

4 Experimental results

A Pioneer robot platform was used to show the performance of the proposed algorithm within the scope of the research. In simulation investigations, the start location of the robot, and the location of target were given for each navigation task. The robot was enabled with self-localization sensors i.e.; azimuth, and distance measurement sensors. Therefore at any time instance, the robot had a global knowledge of its current position, and direction relative to both target and start positions. For sensing the environment locally, the robot was equipped with an array of 8 sonar range finders (2 at sides and 6 at front with 20 degrees interval). A standard simulation software from ActiveMedia Robotics was used for graphic representation and programming of the robot. The programming language was Colbert which is kind of C++ added with special functions for robot motion and self-localization.

Fig. 14 depicts the expected results as previously discussed in Section 3. Unlike the memory state method [11], here the robot does not have extra wall following and goes directly toward the target as soon as it is out of the dead-end trap. And because the motion is not based on trail and return, the robot makes more logical trajectories compared to the minimum risk approach. The fact that the algorithm does not require any kind of global map e.g.; memory map or grid of discrete cells [16], is advantageous in shorter runtime. However integration of a global map would surely improve the overall performance of the robot.

Fig. 14. Robot wayfinding, compared with related works (Fig. 2, 3.a, 5, 7.b)

A more complicated trap of this type is the snail shape shown in Fig. 15. Here, after straight motion from “A” to “B”, the robot encounters the first wall and therefore follows the walls from “B” to “C” in hope of finding an opening. As soon as it detects the opening at point “C” most logical decision is to follow the walls in the opposite direction from “C” to “D” where the robot switches to actual target seeking behavior. More example of this type is shown in Fig. 16, to prove the robustness of the algorithm in different multiple dead end situations.

Fig. 15. Snail shape trap

Fig. 16. Robot trajectories in more sample environments
5 Conclusion

A real-time FL algorithm for mobile robot reactive navigation was presented. The inputs to the fuzzy controller included; obstacle position relative to the robot heading and the target orientation which was defined as the angle between the robot heading direction and the robot-to-target direction. While the fuzzy logic body of the algorithm performed the main tasks of obstacle avoidance, target seeking and speed control, an actual-virtual target switching strategy enabled the robot for wall following behavior when needed. This significantly resulted in resolving the problem of limit cycles. This work is in progress for enabling local minimum avoidance in dynamic space where cul-de-sac obstacles change their shapes dynamically. It is also intended to include more obstacle features for better path prediction especially for minimum avoidance and efficient action selection e.g.; steering and velocity. Another extension of this research will be on integration of an indoor communication system using wireless technology. Such a system is employed to supply global knowledge to the robot while it is locally exploring the environment.

6 References


