Robust Local Obstacle Avoidance for Mobile Robot based on Dynamic Window Approach

Piyapat Saranrittichai[1], Nattee Niparnan[2], Attawith Sudsang[3]

Department of Computer Engineering, Chulalongkorn University
254 Phyathai Road, Wangmai, Pathumwan, Bangkok, Thailand, 10330

[1]charn_pay@hotmail.com

[2]nattee.n@chula.ac.th

[3]attawith@cp.eng.chula.ac.th

***Abstract*— To travel within the real world safely, an autonomous mobile robot has to perform obstacle avoidance. One of the most popular approaches to solve local obstacle avoidance is Dynamic Window approach. This approach is to search optimal control command of the robot in velocity space directly. The command which maximizes objective function is then selected as output command. In the original Dynamic Window approach, in order to avoid obstacles, it concerns only obstacles on robot's trajectory. Therefore, the robot might crash with some obstacles which are near the trajectory but not on the trajectory. Therefore, in this work, we propose field Dynamic Window approach (F-DWA) which objective function is modified to consider the obstacles near the trajectory as well. This can be done by the use of histogram grid representation of obstacles to estimate the crashing probability of the trajectory. The objective function of Dynamic Window approach also depends on the crashing probability as well as other traditional factors. To test the robustness of proposed algorithm, experiment in real world is conducted. The result is the robot can travel safer when it concerns near-trajectory obstacles.**

***Keywords*— Local Obstacle Avoidance, Dynamic Window Approach, Differential Drive, Circular Trajectory, Histogram grid.**

I. INTRODUCTION

In autonomous mobile robot, obstacle avoidance is needed to navigate the robot to destination location safely. The task of obstacle avoidance is to select the most appropriate control command for the robot in order to drive it toward the preferred location without crashing any obstacles. In this work, we focus on local obstacle avoidance which concerns only obstacles in current frame.

Currently, obstacle avoidance is a well-studied topic. For instance, I. Ulrich proposed Vector Field Histogram plus (VFH+) [1]. VFH+ uses many representations to select the most proper control command. First, VFH+ considers obstacles in histogram grid representation. Second, the histogram grid is then converted to polar histogram. Finally, the optimal command is chosen using polar histogram. Other than VFH+, Dynamic Window approach (DWA) is also well-known obstacle avoidance algorithm [2]. This approach is proposed by D. Fox. DWA is to search for optimal command directly in possible command space of the robot. It concerns some constraints of the robot such as maximum acceleration, possible trajectories, etc. Hence, the output command from this algorithm is seemed well-suited in various conditions. Currently, there are many extension of Dynamic Window approach [4, 5, 6, 7, 8]. For example, O. Brock and O. Khatib developed global Dynamic Window approach to apply Dynamic Window approach in global obstacle avoidance problem [7]. C. Schoröter applied particle filter techniques with Dynamic Window Approach to optimize objective function for clothoids curves trajectories [5]. Furthermore, J. G. Li also proposed odor path estimation with Dynamic Window Approach [8]. To perform obstacle avoidance in dynamic environment, M. Seder developed the algorithms which are the combination of FD* search algorithm and Dynamic Window Approach to do obstacle avoidance [6].

Dynamic Window approach provides good results in many scenarios. However, in the case that robot sensor cannot precisely estimate the obstacles' location, the robot might scrape with some obstacles which their measures are not exactly on the robot trajectory but still near the trajectory. This scraping is likely to occur because the original objective function of Dynamic Window approach concerns only the obstacles on the robot trajectory. In this work, we develop the modification version of Dynamic Window approach which includes the effect of near-trajectory obstacles into the objective function. Please note that in this work, the robot shape is assumed to be circular shape.

This paper is organized as follow. In chapter II, we explain the movement of differential drive robot. Then, the concept of Dynamic Window approach is introduced in chapter III. Our modification of Dynamic Window approach is proposed in the following chapter. To demonstrate the performance of our proposed method, the results of the experiment is presented in chapter V. Finally, the conclusion of our work is in the last chapter.

II. DIFFERENTIAL DRIVE ROBOT

In this chapter, we introduce the motion control model of differential drive robot and explain how to estimate the trajectory of the robot if we know control command.

To estimate the trajectory of the robot, we have to understand how the robot moves. The control command of any differential drive robots consists of two values, velocity of left and right wheels as in equation (1).

978-1-4799-0545-4/13/$31.00 ©2013 IEEE

$$\vec{v} = [v_L \quad v_R]^T \tag{1}$$

, where v_L and v_R are the velocity of left and right wheels respectively.

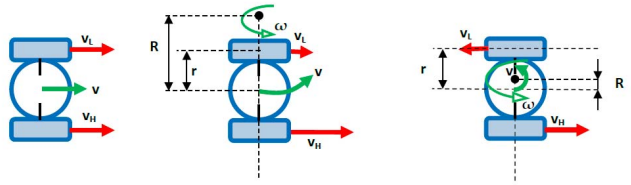

Fig. 1 shows robot trajectory for differential drive robot in three possible cases (a) both wheels have same velocity (b) both wheels have same moving direction but different magnitude (c) both wheels have opposite moving direction.

There are three possible cases for movement of differential drive robot as shown in Fig. 1. In the first case, both wheels have same velocity. In this case, the trajectory of the robot is a straight line in the same direction of the velocity (Fig. 1(a)). However, if the velocity of each wheel still in the same direction but different magnitude, it will be in the second case. The robot trajectory is a circular trajectory with origin located outside the wheels boundary in the side of lower velocity (Fig. 1(b)). The radius of its circular trajectory and angular velocity around its origin can be calculated by the formulas in equation (2) and (3) respectively.

$$R = \frac{|v_H + v_L| r}{|v_H - v_L|} \tag{2}$$

$$\omega = \frac{|v_L|}{R - r} = \frac{|v_H|}{R + r} \tag{3}$$

, where r is robot radius which is measured from the center point between both wheels to any wheel. v_L and v_H are velocities of wheels with lower and higher magnitude respectively.

In the last case, both wheels have velocity in opposite direction. As a result, robot trajectory is also circular trajectory but its origin is located inside robot's wheel boundary in the side of lower velocity (Fig. 1(c)). The radius of the trajectory and angular velocity around its origin can be calculated by the formulas in equation (4) and (5) respectively.

$$R = \frac{|v_H - v_L| r}{|v_H + v_L|} \tag{4}$$

$$\omega = \frac{|v_L|}{r - R} = \frac{|v_H|}{r + R} \tag{5}$$

III. Dynamic Window Approach

Dynamic Window approach is obstacle avoidance algorithm which is first proposed by D. Fox [2]. The concept of Dynamic Window approach is to search for an optimal control command in the command space directly. The optimal control command is the command that maximize objective function. Overall, there are two steps in Dynamic Window approach. The first step is to limit search space to only possible command space. Then, each command is tested its appropriateness with objective function in the second step. The command with highest value of objective function is chosen as optimal command. Optimization is performed by discretising search space. Here is detail of each step.

A. First Step : Search Space

In this step, each command in discretised search space is tested if it is possible and safe to control robot. The set of possible commands might be limited according to robot maximum acceleration. In addition, the safety of each command can be checked if, with this command, the robot can stop before reaching nearest obstacle on trajectory. Finally, the result search space contains only safe and possible commands.

B. Second Step : Optimization

The optimization step is to pick the most appropriate command. The appropriateness of each command can be measured by the value of objective function. In original Dynamic Window approach, objective function is defined as in equation (6)

$$G(\vec{v}) = c_1 \cdot heading(\vec{v}) + c_2 \cdot obstacle(\vec{v}) + c_3 \cdot vel(\vec{v}) \tag{6}$$

, where c_1, c_2 and c_3 are constants.

$heading(\vec{v})$ is measured the angular proximity of the trajectory toward the destination.

$obstacle(\vec{v})$ is the distance to the nearest obstacle on the trajectory of $\vec{v}$.

$vel(\vec{v})$ is the measured of speed of the robot.

Using this original Dynamic approach, the robot can perform local obstacle avoidance toward destination location successfully in many cases. Nevertheless, the objective function of this approach considers only obstacles on the command's trajectory. As a result, the command which maximizes objective function might have its trajectory near obstacles. This increases the possibility to crash with the obstacles, specifically, in the case that the measured locations of the obstacles aren't precise.

IV. Field Dynamic Window Approach

In our version of Dynamic Window approach called Field Dynamic Window approach (F-DWA), the obstacles are processed in histogram grid representation. One of this representation's advantages is the easiness to determine the

proximity of any cells to the nearest obstacles. In other words, it is easy to search for the nearest obstacle in the trajectory in $obstacle(\vec{v})$ function in equation (6). Moreover, it helps to determine the safety of the trajectory which is used in our proposed approach. The detail of F-DWA is explained in this section.

A. Modify objective function

In grid Dynamic Window approach, the objective function is modified to equation (7).

$$G(\vec{v}) = c_1 \cdot dist(\vec{v}) + c_2 \cdot heading(\vec{v}) + c_3 \cdot obstacle(\vec{v}) + c_4 \cdot vel(\vec{v}) + c_5 \cdot safety(\vec{v}) \quad (7)$$

, where c_1, c_2, c_3, c_4 and c_5 are constants.

$dist(\vec{v})$ is the proximity of destination point and its projection on the trajectory. It is zero if the destination point is on the trajectory.

The key value is $safety(\vec{v})$ which indicate the probability that robot doesn't crash with any obstacles if it applies $\vec{v}$ command. The objective function is affected by near-trajectory obstacles via $safety(\vec{v})$ value. In this work, $safety(\vec{v})$ can be calculated by the equation (8).

$$safety(\vec{v}) = 1 - crash(\vec{v}) \quad (8)$$

, where $crash(\vec{v})$ is the probability of crashing in the trajectory generated by $\vec{v}$.

B. Estimate Crashing Probability

To estimated crashing probability, histogram grid is used. The steps to estimate this probability is as follow.

1) Initializing Histogram

When the robot measures the distance to obstacles using any range sensor, detected obstacles are mapped into cells in histogram. Firstly, the initial value for each obstacle cell is n_0 as defined in equation (9) while other cells which do not belong to any obstacle have values of zero.

$$n_0 = \frac{r}{w_{cell}} + k \quad (9)$$

, where r is robot radius. w_{cell} is the width of each cell in histogram in world frame. k is constant which is non-negative integer. The more value of k indicates the more uncertainty of obstacle locations.

2) Spreading obstacle cells' values

Next, the values of obstacle cells are spread to nearby cells. This can be done by simple flood-fill algorithm. In detail, each time a cell is spreading its value c, its adjacent cells will set their values to be the maximum values of their current values and $c-1$. The histogram continues spreading cell values until there is no possible change occurred by spreading process.

The result of spreading process is that the value of each cell indicates the proximity of itself to the nearest obstacles. The more value of the cell indicates the nearer distance from it to its nearest obstacle. From equation (9), the cell which is an obstacle cell has the value in range $[k, n_0]$. The cell in this type doesn't allow the robot to be presented in its location. Another type of cell has the value in range $(0, k)$. The cell in this type is near obstacle. Lastly, the cell which is far from obstacle will have the value of zero.

3) Calculate Probability of Crashing in the Trajectory

For each trajectory, its probability of crashing will be calculated using the formula in equation (10).

$$crash(\vec{v}) = \frac{\sum_{\vec{p} \in Trajectory(\vec{v})} P_{obstacle}(\vec{p}) \cdot P_{rpbpt}(\vec{p})}{\sum_{\vec{p} \in Trajectory(\vec{v})} P_{robot}(\vec{p})} \quad (10)$$

, where $\vec{p} = [p_x \quad p_y]^T$ is a cell in $trajectory(\vec{v})$ calculated by the method described in chapter II. $P_{robot}(\vec{p})$ is the probability that robot is in cell $\vec{p}$ given normal distribution of robot location. $P_{obstacle}(\vec{p})$ relates to the probability that cell $\vec{p}$ is an obstacle. It can be calculated using the value of cell $\vec{p}$. If the cell $\vec{p}$ has value $hist(\vec{p})$, then, $P_{obstacle}(\vec{p})$ of any cells can be calculated by

$$P_{obstacle}(\vec{p}) = \begin{cases} 1 & hist(\vec{p}) \in [k, n_0] \\ \frac{N(hist(\vec{p}); \mu = k, \sigma_{obs})}{N(k; \mu = k, \sigma_{obs})} & hist(\vec{p}) \in (0, k) \\ 0 & hist(\vec{p}) = 0 \end{cases} \quad (11)$$

, where $N(x; \mu, \sigma)$ is the probability density function value x in Gaussian distribution with mean μ and standard deviation σ.

From equation (11), You'll see the cells is classified into 3 types. First, the obstacle cell, has the value of $P_{obstacle}(\vec{p}) = 1$ because of its high possibility to has an obstacle. Second, the near-obstacle cell, which has the value in range $(0, k)$, always has value of $P_{obstacle}(\vec{p})$ lower than 1. The lower value of $hist(\vec{p})$, the lower the value of $P_{obstacle}(\vec{p})$. In other words, if the cell is far from obstacle then $P_{obstacle}(\vec{p})$ is low. Lastly, the free cell, which its value is 0. It has low possibility to has an obstacle. Therefore, $P_{obstacle}(\vec{p})$ is set to zero in this case.

V. EXPERIMENT

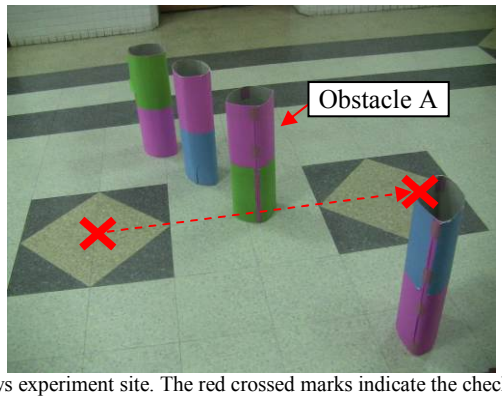

Fig. 2 shows experiment site. The red crossed marks indicate the checkpoints.

The experiment is taken place at Chulalongkorn University, Thailand (Fig. 2). Robot platform iRobot Create, which is differential drive robot, is used. The robot (Fig. 3) is equipped with Hokuyo URG-04LX-UG01 laser range finder to measure the obstacle data from the environment. The Logitech camera with fisheye lens is also installed on the robot in order to perform localization. The program is all written in C++ running on Laptop CPU Core i3 M370 2.4 GHz.

Fig. 3 Robot which is used in the experiment.

In the experiment, the robot has to travel from the left checkpoint to the right check point in Fig. 2. The checkpoints are indicated by red crossed marks. Obstacle A makes the robot cannot go to destination directly. To perform localization, we use Extend Kalman Filter to track the location of the robot using visual features from the ceiling. The feature map is created before runtime using EKF-SLAM [3].

To test local obstacle avoidance module, obstacles are placed on the floor. Then the robot has to travel to its destination without crashing with any placed obstacles. F-DWA is tested in two cases. First case is tested without safety function ($c_5 = 0$). And, second case is tested with safety function ($c_5 \neq 0$). After testing both cases, we compare the measured distance to the nearest obstacle obtained from laser range finder in both cases. The result is shown in Fig. 4.

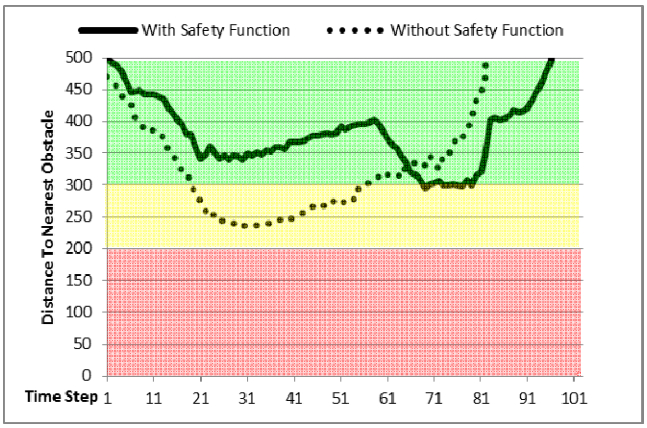

Fig. 4 shows the nearest obstacle distance obtained from laser range finder in each time step in two cases. The dotted and solid lines correspond to the case without safety function and with safety function respectively. The colored areas indicate the type of cells in which robot located. Red, yellow and green areas relate to obstacle, near-obstacle and free cell respectively.

From Fig. 4, in the case which doesn't use safety function, the distance from robot to nearest obstacle is partly in the yellow area which indicates that the robot is in near-obstacle cell. In the other hand, in the case with safety function, the robot spends almost all of the time in green area which indicates that the robot travels mostly in free cell. In other words, it is safer if safety function is applied. Please note that, when the robot is near the destination, the distance to nearest obstacle is likely to increase because laser range finder is faced in the direction that has no obstacle.

VI. CONCLUSIONS

In this work, we propose Field Dynamic Window approach (F-DWA), which is the modification of original Dynamic Window approach, to perform local obstacle avoidance. The objective function of our approach also concerns near-trajectory obstacles. This makes optimal control command to be safer than one without concerning near-trajectory obstacle.

REFERENCES

[1] I. Ulrich and J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots," IEEE International Conference on Robotics and Automation, May, 1998.

[2] D. Fox, W. Burgard and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," IEEE Robotics & Automation Magazine, Mar, 1997.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping:part i,ii," IEEE Robotics & Automation Magazine, June, 2006.

[4] P. Ögren and N. E. Leonard, "A Convergent Dynamic Window Approach to Obstacle Avoidace," IEEE Transactions on Robotics, April, 2005.

[5] C. Schröter, M. Höchemer and H. M. Gross, "A Particle Filter for the Dynamic Window Approach to Mobile Robot Control," In Proceeding 52nd International Scientific Colloquium, 2007.

[6] M. Seder and I. Petrović, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," IEEE International Conference on Robotics and Automation, April, 2007.

[7] O. Brock and O..Khatib, "High-speed Navigation Using the Global Dynamic Window Approach," IEEE International Conference on Robotics & Automation, May, 1999.

[8] J. G. Li, Q. H. Meng, F. Li and M. L. Zhang, "Mobile Robot based Odor Path Estimation via Dynamic Window Approach," IEEE Conference on Robotics, Automation and Mechatronics, 2008.