Volume 5, Number 9(A), September 2011 ISSN 1881-803X

ICIC Express Letters

An International Journal of Research and Surveys

Editors-in-Chief
Yan Shi, Tokai University, Japan
Junzo Watada, Waseda University, Japan
Yunfu Huo, Dalian University, China

Indexed by *Ei Compendex (Elsevier)*
Scopus (Elsevier)
INSPEC (IET)

Published by ICIC International
http://www.ijicic.org/icicel.htm

ICIC EXPRESS LETTERS

Volume 5, Number 9(A), September 2011

CONTENTS (*Continued*)

(*Continued*)

ICIC Express Letters ICIC International ©2011 ISSN 1881-803X
Volume **5**, Number **9(A)**, September **2011** pp. 3109–3115

OSCILLATION ELIMINATION FOR MOBILE ROBOT BASED ON BEHAVIOR-MEMORIZING

Fei Liu, Shan Liang*, Xiaodong Xian and Huibo Bi

College of Automation
Chongqing University
No. 174, Shazheng Street, Shapingba District, Chongqing 400030, P. R. China
*Corresponding author: lightsun@cqu.edu.cn

Received January 2011; accepted March 2011

Abstract. *The problem of dynamic obstacle avoidance must be solved in the motion plan for a mobile robot. The oscillation phenomena will occur when a robot moves into some special circumstances, such as rotating around, moving from side to side and switching between going ahead and back. In this paper, the strategy behavior-memorizing that takes reasonable action by considering both the actions taken a period of time ago and the current situation of obstacles simultaneously is presented and applied to eliminate the oscillation phenomena in the process of avoiding obstacles. At last, the effectiveness of this strategy is proved by simulations.*

Keywords: Mobile robot, Obstacle avoidance, Oscillation, Canyon, Behavior-memorizing, Leaky-integrator

1. Introduction.

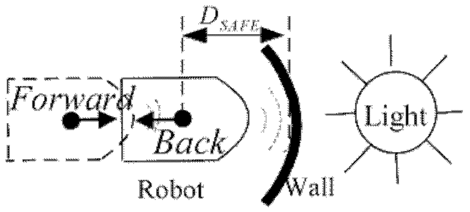
Dynamic obstacle avoidance is a key problem in the motion plan for a mobile robot [1, 2, 3]. The robot may face some particular situations in an unknown environment and go into a state of oscillation including rotating in a circle, turning from side to side and moving forward and backward repeatedly, etc. [4]. The movements referred above are not good for motion plan and harmful to the robot's motors.

In a case that a light guides a robot to go toward it, wherein a low wall is between the start point and the goal, the robot will be attracted to move forward and it may be caused to return back by the wall. If this specialty of the environment is not sensed right now, the robot will then fall into the switch of moving forward and backward, which is a typical example of oscillation. When the artificial potential field (or simply APF) method is utilized for path planning, the robot is incident to be trapped in a box canyon because the attractive potential from the goal is insufficient to overcome the repulsive potentials from the walls, which is called a local minima problem [5]. Some attempts have been made to solve this problem by modifying the APF function [6, 7]. However, these algorithms are more complex, which may weaken the timeliness, and their applications are limited by diverse circumstances. Another method called wall-following is also taken to escape such obstacles [8]. However, the robot may turn left according to the pre-designed strategy persistently and cannot get out of the trap, whereas it can avoid the obstacle by turning right with just a small angel. In addition, some researchers are willing to create several behaviors and define which is suitable for a special avoidance scenario that the robot may get exposed to, one behavior for corridors and another one for a canyon, for instance. Then, the programmer chooses a certain behavior after identifying the real situation encountered [9]. The disadvantage is that the terrain characteristics need to be known first. Besides, the way "avoiding the past" is ever utilized to avoid obstacles, but it takes much time to detect the spatial memory [10].

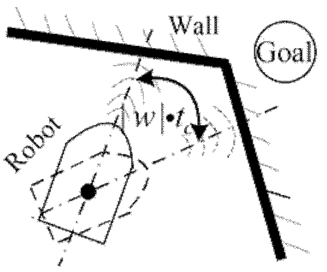
In this work, several cases that will lead the mobile robot to oscillation are described firstly. Then, the strategy behavior-memorizing is detailed as well as its central theory "leaky-integrator", and based on which, the concrete process of eliminating oscillation is stated. At last, simulations are carried out on the platform *MobileSim* and the results show the effectiveness of this strategy.

2. Typical Cases of Oscillation.

Take the situation into consideration showed in Figure 1(a). Behavior *Forward* drives the robot moving toward the light while *Back* is used to drag it back when an obstacle is in the front. Obviously, *Back* has a higher priority than *Forward*. The distance D_{SAFE} is set to keep the robot from colliding with obstacles. Now, the behavior *Back* will trigger when the robot want to go on moving where the distance to the wall is D_{SAFE}. After moving back for a while, it will not feel the wall and move toward the wall again. If so, the robot might repeat those actions without getting to the goal eventually.



(a) A robot moves toward the light



(b) A robot rotates from side to side

FIGURE 1. Two examples of oscillation

The robot might rotate from side to side in another situation showed in Figure 1(b). Assume that rotational velocity is $|w|$ and the period of obstacle detection is t_c. When an obstacle is found on the left, the robot rotates toward right at a velocity of $|w|$ for time t_c. Then, the obstacle on the right is detected which will excite the robot to turn left with velocity $|w|$ for time t_c, too. Thus, the robot will be trapped in oscillating that shows as turning from side to side.

A much more serious case is showed in Figure 2. While in a box canyon, the robot will be stuck since the two phenomena occur above may happen simultaneously.

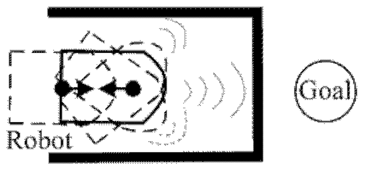


FIGURE 2. A robot is stuck in a box canyon

3. The Strategy Behavior-Memorizing.

The reason for the occurrences of those oscillation phenomena lies in that the behavior executed is decided only by the surrounding conditions detected at the current instance. Actually, the effect exert to the robot by actions carried out in a certain period of time ago is also related. Therefore, to eliminate oscillation, the robot is requested to remember what it has done just now. The basic technique of strategy behavior-memorizing is that the action executed this moment is determined by both the result generated by motions executed just now and the environment situation checked presently.

3.1. **The implementation of strategy behavior-memorizing.** The central theory of this strategy can be depicted as the robot should memorize the action taken just now so that it will not do the contrary operation for some time. Hence, the robot can keep this motion during this fixed period of time instead of chattering repeatedly. The implementation of this strategy is showed in the form of flowchart in Figure 3.

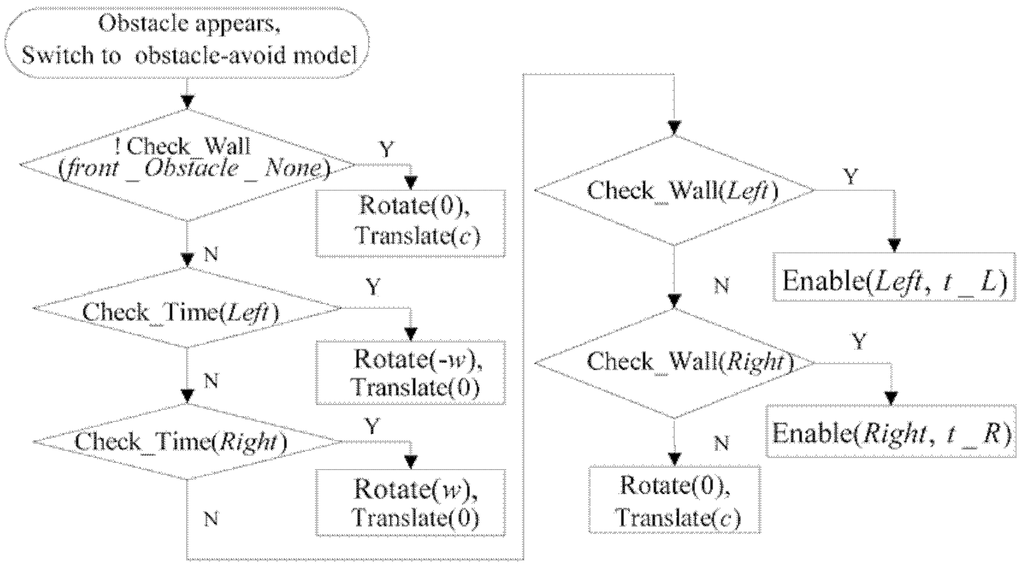


FIGURE 3. The implementation of strategy behavior-memorizing

In the flowchart showed in Figure 3, w is the rotational velocity (being positive if rotating anticlockwise). t_L and t_R are the time that rotating left (clockwise) and right (anticlockwise) respectively, and c (c>0) is the translational velocity. Function $Check\_Wall(dir)$ is used to detect whether any obstacle is existing in direction dir, and Behaviors $Rotate(v_r)$ and $Translate(v_t)$ can activate the robot to rotate at a speed of v_r and translate v_t. $Enable(dir, T)$ will start the timer that is used to count the time for rotating toward direction dir, and T is the total time to run. The return value of function $Check\_Time(dir)$ will be *true* only before time counts up to T after the related timer starts, or $false$ in other cases.

The robot switches to obstacle-avoidance model as soon as any obstacle is detected. According to strategy behavior-memorizing, it will try moving toward the opposite direction to avoid the obstacle when detecting that an obstacle is in direction dir, then if there still exists obstacles on the other side, it will not stop immediately and change the rotating direction, and however, go on rotating as before till the timer stops. This tactics can prevent the robot from oscillating rapidly. If t_L and t_R are set to be constants, we find that the robot might just oscillate with a bigger amplitude rather than get rid of this bad action after several times of rotation. So, t_L and t_R are handled as variables whose valves are determined by a leaky-integrator.

3.2. ***t_L*, *t_R* and leaky-integrator.** Leaky-integrator is a very practical technique, which can memorize what happened during a period of time in the past with finite states. Next, the physical model of a leaky-integrator is presented firstly, then we will elaborate how to apply this principle into strategy behavior-memorizing to determine the values of t_L and t_R.

3.2.1. *The physical model of a leaky-integrator.* The physical model is showed in Figure 4 [4]. A bottle with a leak is employed to imitate a leaky-integrator. The process pouring water into the bottle ($Leak\_in$) represents the action "integrating", and the water leak out

(*Leak_out*) "leaking". Such a bottle has three features "the highest water level", "actual water level" and "the lowest water level", which correspond to the three parameters of a leaky-integrator which are the max value $L\_\max$, real value L_sum and the min value $L\_\min$ respectively. Obviously, $L\_\min \le L\_sum \le L\_\max$.

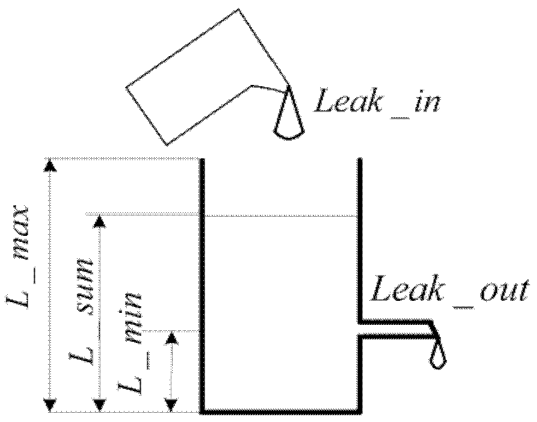


FIGURE 4. A leaky-integrator

3.2.2. *Determination of t_L and t_R.* Define *period_update* is the interval time for updating the integration of the leaky-integrator, which is equal to t_c. Δ_{out} is the leakage of *leak_out* when updating, and Δ_{in} is the value of *leak_in* when the injection *Leak_in* is not zero. The working process of a leaky-integrator is described in the form of pseudo code below.

```
$Leaky\_integrator(leak\_out, \min, \max)$
{
    $Leak\_sum = Modify(L\_sum + leak\_in - leak\_out, \min, \max);$
    $leak\_in = 0;$
}
```

In the progress above, *leak_in* will be cleaned to zero during the running process of *Leaky_integrator*, which means resetting the sampled data. The function *Time_Calculate* that calls *Leaky_integrator* is depicted by expression below:

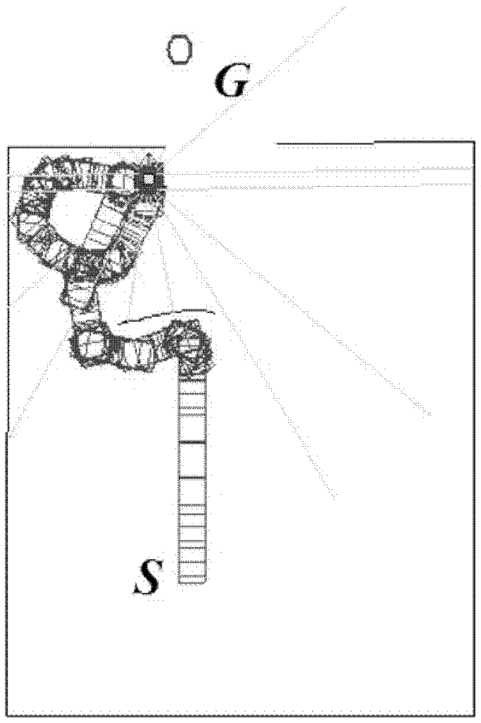
$$Time\_Calculate(period\_update, Leaky\_integrator(leak\_out, L\_\min, L\_\max)) \quad (1)$$

When called for the first time, the parameter min of function *Leaky_integrator* equals $L\_\min$ so that the robot will rotate at an angular speed w for a short period of time to avoid the obstacle. $L\_\max$ is corresponding to time $\pi/|w|$ which means that the robot rotates to the opposite direction, and any bigger value than $L\_\max$ becomes nonsense.

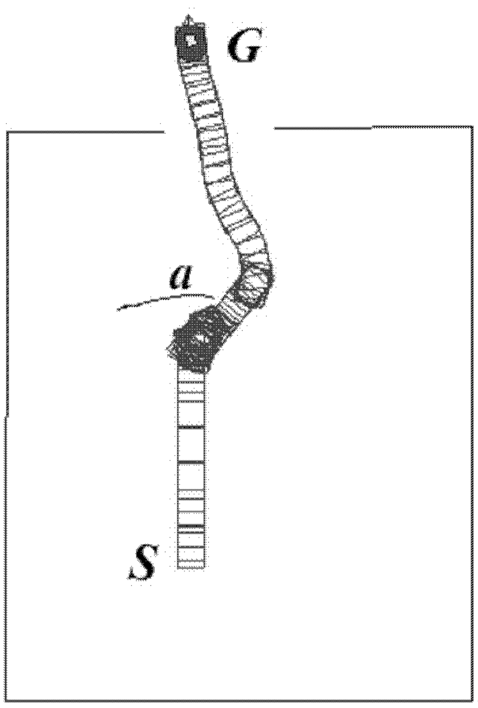
With the application that function *Time_Calculate* is used to determine *t_L* and *t_R* in strategy behavior-memorizing, the execution result is that when an obstacle is detected in one direction and the same in the other direction, *leak_in* becomes θ_{in} ($\theta_{in}>0$), then the value of *Leak_sum* gets bigger. Since *Leak_sum* is used to control the time for rotating, the robot will rotate for a bigger angle. Later, when no obstacle could be detected, *leak_in* becomes zero more frequently among the times the function *Leaky_integrator* is called, which will make *Leak_sum* decrease to $L\_\min$ till the robot exits the obstacle-avoidance procedure.

4. Simulations and Results.

The simulations are carried out on the platform *MobileSim* supported by the company *ActivMedia*, and the parameters of the mobile model are set based on the real robot *Pioneer3-AT* [11]. The strategy behavior-memorizing is realized on *Visual Studio 2003.Net*, where the *Aria* library is used to link the mobile model for driving the virtual distance sensors and motors etc. The simulation results are shown in Figure 5(a), Figure 5(b) and Figure 6.



(a) Avoid obstacle using wall-following



(b) Avoid obstacle using behavior-memorizing

FIGURE 5. Comparison of strategies behavior-memorizing and wall-following

In the environment shown in Figure 5, a wall (noted as a) is placed between the start point and the goal. The comparison of two simulations about obstacle-avoidance using strategies wall-following and behavior-memorizing respectively are showed in Figure 5(a) and Figure 5(b). In Figure 5(a), the robot turns left (given that the robot always turns left when an obstacle is found) after arriving at point a by translating linearly toward the goal, then the action wall-following guides the robot walking along the wall. After rounding along the wall, the robot runs to the goal straightly. Unfortunately, a wall (located in point b) is encountered once again, and this time the robot returns to the place b after avoiding the wall by following it. Finally, the robot falls into the circle that walking round and round. On the contrary, in Figure 5(b), when the robot reaches point a, it turns left for a small angle first, then the obstacle can still be found on the right which leads the robot turn right for a bigger angle hoping to escape. When one time after repeating these behaviors no obstacle can keep the robot from moving forward, the angel to turn will decrease down to L_min. Soon, the obstacle-avoidance model is quitted which indicates that the robot successfully avoids the wall and gets to the goal ultimately.

A simulation of dynamic obstacle-avoidance in an environment with a box canyon using strategy behavior-memorizing is illustrated in Figure 6. The robot enters a box canyon placed between the start point S and the goal G. Then, it tries to turn aiming to not colliding with the wall. Similarly it first rotates for a small angle. As the obstacle can still be checked, the time to rotate increases step by step, afterwards when it is checked that the robot can move without touching any obstacle, the time decreased down to L_min

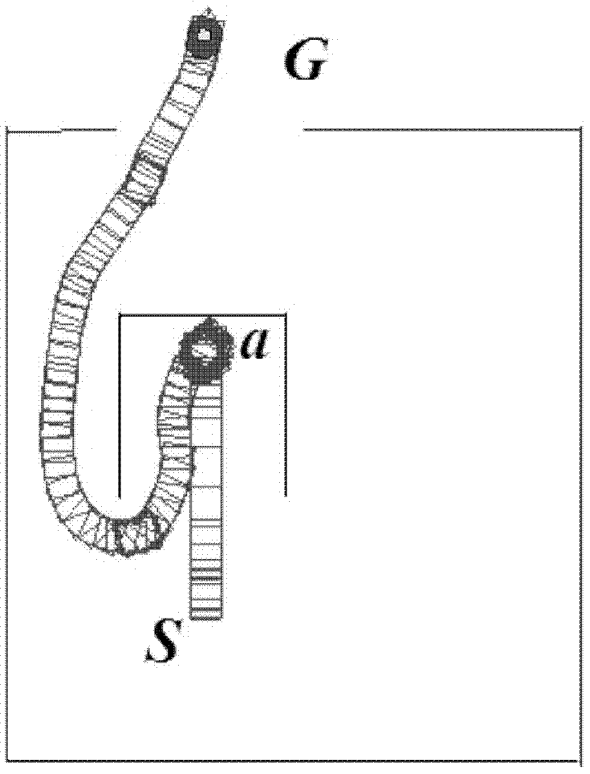


FIGURE 6. Escape the box canyon using behavior-memorizing

gradually. Eventually, the obstacle avoidance progress exits along with the result that escaping the box canyon successfully.

5. Conclusions.

One basic problem in the field of motion planning for mobile robot is dynamic obstacle avoidance. This study focuses on the solution for oscillation phenomena when avoiding obstacles by applying the strategy behavior-memorizing which holds the idea that a wiser behavior should be activated by both the actions taken in a period of time in the past and circumstance currently be considered comprehensively. The strategy is tested by simulating on platform *MobileSim* after the introduction about it. Simulations results show the oscillation problem can be overcame by using strategy behavior-memorizing approach. In the future study, we will put this strategy into use on the real robot and environment.

Acknowledgment. This research was sponsored by Project No. CDJXS11171158 Supported by the Fundamental Research Funds for the Central Universities and the Key Project of Science and Technology Committee of Chongqing (CSTC, 2009AB2139).

REFERENCES

[1] M. G. Park, J. H. Jeon and M. C. Lee, Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing, *IEEE International Symposium on Industrial Electronics*, Pusan, Korea, pp.1530-1535, 2001.
[2] L. Lapierre, R. Zapata and P. Lepinay, Simultaneous path following and obstacle avoidance control of a unicycle-type robot, *IEEE International Conference on Robotics and Automation*, Roma, Italy, pp.2617-1622, 2007.
[3] Y.-C. Chen and Y.-T. Wang, A method for multiple-obstacle avoidance of soccer robot systems, *ICIC Express Letters*, vol.4, no.1, pp.161-166, 2010.
[4] J. L. Jones and D. Roth, *Robot Programming: A Practical Guide to Behavior-based Robotics*, 1st Edition, McGraw-Hill/TAB Electronics, 2003.

[5] M. A. Goodrich, *Potential Fields Tutorial*, http://borg.cc.gatech.edu/ipr/files/goodrich_potential_fields.pdf, 2004.
[6] Y. Zhu, T. Zhang and J. Y. Song, Study on the local minima problem of path planning using potential field method in unknown environments, *Acta Automation Sinica*, vol.36, no.8, pp.1122-1130, 2010.
[7] M. K. Zhang and L. S. Li, A method for solving local minimization problem of artifical potential field, *Computer Technology and Development*, vol.17, no.5, pp.137-139, 2007.
[8] S. Fazli and L. Kleeman, Wall following and obstacle avoidance results from a multi-DSP sonar ring on a mobile robot, *International Conference on Mechatronics and Automation*, Niagara Falls, Canada, pp.432-437, 2005.
[9] A. Ranganathan and S. Koenig, A reactive robot architecture with planning on demand, *Conference on Intelligent Robots and Systems*, Las Vegas, NV, pp.1462-1468, 2003.
[10] T. Balc and R. Arkin, Avoiding the past: A simple but effective strategy for reactive navigation, *International Conference on Robotics and Automation*, Atlanta, pp.678-685, 1993.
[11] ActivMedia Corporation, *Aria Reference Manual*, http://robots.mobilerobots.com, 2003.