International Journal of Fuzzy Logic and Intelligent Systems, vol. 9, no. 4, December 2009 pp. 315-320

Path Planning for Autonomous Mobile Robot using Potential Field

Kwang-Min Jung and Kwee-Bo Sim*

School of Electrical and Electronics Engineering, Chung-Ang University
221 Heukseok-dong Donnjak-gu, Seoul, Korea
Tel : +82-2-820-6396, Fax : +82-2-813-8258, E-mail : kbsim@cau.ac.kr

Abstract

The popularity of autonomous mobile robots have been rapidly increasing due to their new emerging application areas, from room cleaning, tourist guidance to space explorations. However, the development of a satisfactory control algorithm that will enable the autonomous mobile robots to navigate safely especially in dynamic environments is still an open research problem. In this paper, a newly proposed potential field based control method is implemented, analyzed, and improvements are suggested based on experimental results obtained from simulations. The experimental results are presented to show the effectiveness of the behavior-based control using the proposed potential field generation technique.

Key Word : Path planning, Potential field, Autonomous mobile robot, Navigation, Dynamic environment

1. Introduction

With the exponential increase in the scientific knowledge and thus technology in recent years and in particular with the advent of computers, stories in science fiction books started to become true, and robots started to appear in daily life. Robots can help human beings; they can clean the room, disarm land mines, make explorations on the Moon, help surgeons in operations, work in factory environments, run production lines doing the jobs hard, dangerous or boring for humans. One of the fundamental challenges in robotics is motion or path planning, which means to generate a continuous path for a given robot between an initial and a goal configuration of the robot. Along this path, the robot must not intersect given forbidden regions, which are usually obstacles. Different versions of the motion planning problem correspond to the cases where the obstacles are stationary or moving and where the system of objects consists of a point, polygonal object, a single polyhedral object or a set of polygonal. Practical instances of the motion-planning problem occur frequently in robot-based assembly operations where a continuous trajectory must be planned for a robot such that collision with neighboring objects and other robots in the same workspace is avoided and the structural limits of the joint actuators are not violated. Similar problems are also encountered in the automatic navigation of vehicles in constrained environments, missile guidance, VLSI circuit layout and aircraft routing. Due to its widespread application, there is a lot of interest in this problem in the academic community and various approaches to its solution have been proposed [7].

Manuscript received Nov. 19. 2009 ; revised Dec. 1. 2009.
* Corresponding author
This research was supported by the industry-university-research joint technical development program from the Seoul city and Small & Medium Business Administration(SMBA) of Korea

1.1 Deliberative Control

In deliberative control, the robot uses all of the available sensory information, and all of the internally stored knowledge, to reason about what actions to take next. The knowledge must be consistent, reliable and certain. In a dynamic world, where the objects are moving arbitrarily, it is dangerous to rely on past information that may no longer be valid. Representational world models are therefore generally constructed from both prior knowledge about environment and incoming sensor data in support of deliberation. When there is sufficient time to generate a plan and the world model is accurate, this approach allows the robot to act strategically, selecting the best course of action for a given situation. The advent of alternative architectures was driven by the need for faster yet appropriate action in response to the demands of complex and dynamically changing real-world environments.

1.2 Reactive Control

Reactive control is a technique for tightly coupling sensory inputs and actuator outputs, typically involving no intervening reasoning to allow the robot to respond very quickly to changing and unstructured environments. Reactive control is inspired by the biological notion of stimulus-response. Use of abstract representational knowledge is avoided in the generation of a response, and the robot reacts directly to the world as it is sensed. Mapping between the stimulus domain and response range is achieved by discrete encoding in the form of if-then-else; or continuous functional encoding allowing a robot to have infinite space of potential reactions to its world. Thus, rule-based methods involving a minimal amount of computation, and no internal representations or knowledge of the world are typically used. However, limitations to pure reactivity include the inability to store information or have memory or internal representations of the world, and therefore the inability to learn and improve over time. For example, reactive control is the best choice for

environments demanding immediate response, but such speed of reaction comes at the price of being myopic, not looking into the past or the future.

1.3 Hybrid Control

Hybrid control aims to combine the best aspects of reactive and deliberative control. As a result, hybrid control systems contain two different components, the reactive condition-action rules and the deliberative ones, which must interact in order to produce a coherent output. Hybrid deliberative/reactive robotic architectures have recently emerged combining aspects of traditional AI symbolic methods and their use of abstract representational knowledge, but maintaining the goal of providing the responsiveness, robustness, and flexibility of reactive system.

1.4 Behavior-based Control

Behavior-based control employs a set of distributed, interacting modules, called behaviors, that collectively achieve the desired system level behavior. These behaviors are often reactive in nature, since reactive rules can and often do form components of simple behaviors. There is a distinction between activation conditions, which allow the behavior to generate actions, and stimuli, from which actions are generated. Behavior-based systems are typically designed so the effects of the behaviors interact largely in the environment rather than internally through the system, taking advantage of the richness of the interaction dynamics by exploiting the properties of situatedness. These dynamics are sometimes called emergent behaviors because they emerge from the interactions and are not internally specified by the robot's program. For example, a robot that flocks with other robots may not have a specific flocking behavior. Instead, its interaction with the environment and other robots may result in flocking, although its only behaviors may be avoid-collision, stay-close-to-the-group, and keep-going.

For such an approach to work, a behavior-based system must resolve the issue of choosing a particular action or behavior from multiple options, a process know as action selection or the behavior coordination problem. This is one of the central design challenges of behavior-based systems[4].

2. Proposed Control Algorithm

2.1 Mathematical Model of Robot

The robot used in this work is differential drive mobile robot. It is non-holonomic, having restrictions in the way it can move, due to kinematic or dynamic constraints, such as limited turning ability or momentum at high velocities. Assuming no slip at the tires, the kinematic model of such a robot is given by

$$\begin{aligned} \dot{x} &= \nu \cdot \cos(\varphi) \quad & \nu &= (V_R + V_L)/2 \\ \dot{y} &= \nu \cdot \sin(\varphi) \quad & \omega &= (V_R - V_L)/L \end{aligned} \tag{1}$$

Where ν and ω are the translational and rotational velocities of the robot, V_R and V_L are the right and left wheel velocities, L is the inter wheel distance, and φ denotes the orientation of the robot as shown in Fig. 1.

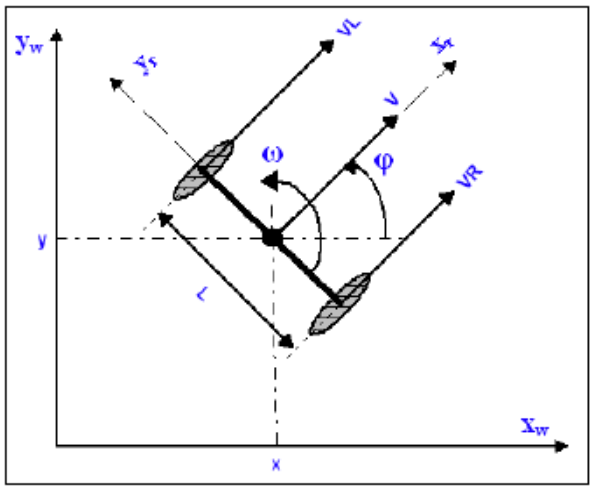

Fig. 1. Kinematic model of the robot

2.2 Potential Field Method

Probably none of the robot navigation methods has attracted so great an interest from researchers as potential fields method [3], such that so many variations of the method have been developed and used [1][2][5][6]. The methods was developed as a basis for generating smooth trajectories for both mobile and manipulator robotic systems. Separate attractive and repelling potential fields are constructed to represent the relationship between the robot and each of the objects within the object's sensory range. These fields are then combined to yield a single global field. A smooth trajectory is computed based upon the gradient within the globally computed potential field.

Drawing from the field theory concept in physics, this method models obstacles as emitting a repelling force and the goal point as emitting an attractive force on the robot. Navigation is performed by moving the robot so as to minimize the potential energy.

This approach assumes knowledge of the type of obstacles in the environment and in the planning phase polygons or spheres approximate these known obstacles. The environment in the original formulation of this idea was assumed to be static, however, there have been adaptations to use this approach for dynamic environments. Potentials are associated with the objects in the environment as and when it is encountered.

Fig. 2(a) shows an example with repelling forces from obstacles and walls, plus a superimposed general field direction from start to goal. And Fig. 2(b) exemplifies the potential field generation steps in the form of 3D surface plots.

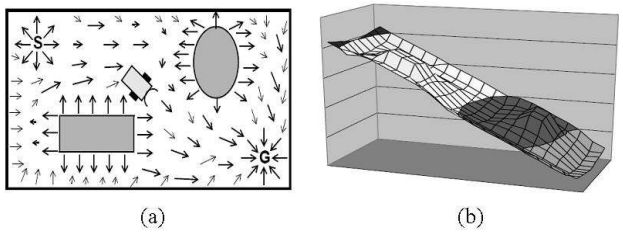

Fig. 2. (a) Potential field as 2D surfaces (b) Potential field as 3D surfaces

2.3 Formulation of Potential Field

As described in the previous chapter, the main idea of potential field method is to assume repelling forces from obstacles and attractive forces from the goal to the robot. The repelling force is inversely proportional to the square of the distance and calculated as,

$$\vec{F}_{obs} = -A \cdot \sum_{i=1}^{n} \frac{1}{d_i^2} \cdot \hat{r}_i \tag{2}$$

Where A is a constant scaling factor, n is the number of obstacles, d_i is the distance between obstacle i and the robot, $\hat{r}_i$ is the direction vector from robot to representative point of obstacle(e.g., center of obstacle, point of obstacle closest to the robot). This is the original repelling force formulation of the potential field and it requires the knowledge of all obstacles in the environment. The magnitude of the repulsive force increases significantly when the robot is close to the obstacle, which is the main idea for avoiding the obstacle.

The attractive force of the goal is in the form.

$$\vec{F}_{goal} = B \cdot d^2 \cdot \hat{r} \tag{3}$$

Where B is similarly a scaling factor, d is the distance from robot to the goal, and $\hat{r}$ is the direction vector from robot to goal.

After calculating each repelling and attractive force, the total force on the robot is calculated by the vector sum of these individual forces, and robot moves in the direction of the resultant force vector in Fig. 3.

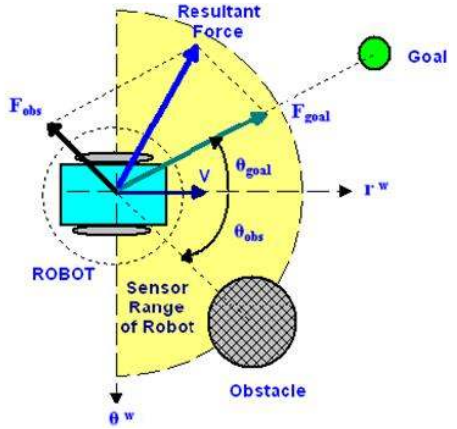

Fig. 3. Potential field forces on the robot

2.4 Design of a Layered Control System

In the proposed new approach a layered structure is adopted, such that instead of adding up the attractive and repelling forces as in classical potential field method, these forces are treated separately in parallel layers, Obstacle Avoidance (OA), Drive Toward Goal (DTG) and Drive Toward Landmark (DTL) in Fig. 4. Each layer produces its own output using potential field method and these outputs are later combined by the behavior arbitration layer to result in the reference orientation of the robot for obstacle avoidance and goal tracking.

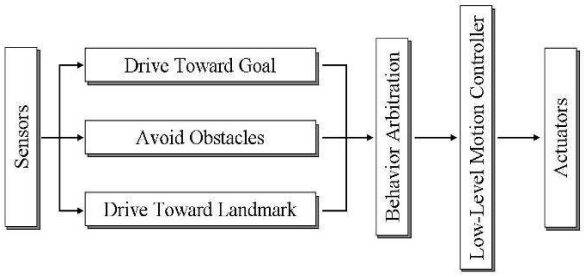

Fig. 4. Layered control structure

2.5 Obstacle Avoidance, Drive Toward Goal and Drive Toward Landmark Layer

In obstacle avoidance layer, using the outputs of the distance sensors, the net repelling force is calculated and decomposed to its components, one along the direction of motion of the robot and one perpendicular to it.

$$\begin{aligned} F_r^{ref} &= \left|\vec{F}_{obs}\right| \cdot \cos(\theta_{obs}) \\ F_\theta^{ref} &= \left|\vec{F}_{obs}\right| \cdot \sin(\theta_{obs}) \end{aligned} \tag{4}$$

For safe navigation, the robot should try to keep the force component along its direction of motion, F_r, minimum or ideally zero. This can be achieved by changing the orientation of the robot, since the force components are dependent on the orientation. To this end, a controller can be used for the optimization. The rate of change of the force components with respect to the obstacle angle is

$$\begin{aligned} \dot{F}_r &= -\left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \sin(\theta_{obs}) \\ \dot{F}_\theta &= \left|\vec{F}_{obs}\right| \cdot \dot{\theta}_{obs} \cdot \cos(\theta_{obs}) \end{aligned} \tag{5}$$

Then the controls can be chosen as,

$$u_r = \dot{F}_r,\ u_\theta = \dot{F}_\theta \tag{6}$$

and errors to be minimized are,

$$\begin{aligned} e_r &= F_r^{ref} - F_r \\ e_\theta &= F_\theta^{ref} - F_\theta \end{aligned} \tag{7}$$

Now, any suitable control method can be utilized for the control of this first order system. In [6], sliding mode controller is used with positive definite Lyapunov function

$\gamma = e^T \cdot e / 2 \geq 0$. In this work, a simple proportional controller is used, resulting in the following controls.

$$\begin{aligned} u_r &= K_{pr} \cdot e_r \\ u_\theta &= K_{p\theta} \cdot e_\theta \end{aligned} \tag{8}$$

where K_{pr} and $K_{p\theta}$ are proportional control gains.

Finally, using equations (5) and (8) together, the desired orientation of the robot for an obstacle free path can be calculated.

$$\frac{u_r}{u_\theta} = \frac{-\sin(\theta_{obs})}{\cos(\theta_{obs})} \rightarrow \theta_{OA}^{ref} = \tan^{-1}(-\frac{u_r}{u_\theta}) \tag{9}$$

Obviously due to the function arctan() , the output will be in the range -90° to $+90^\circ$.

Similar to the obstacle avoidance layer, an attractive force is calculated and decomposed to its components in the direction of motion and perpendicular to it. Contrary to obstacle avoidance layer, this time the force along the motion direction must be maximized and the perpendicular force must be minimized. As a result, we can get the desired orientation of robot for goal or landmark tracking as follows.

$$\frac{u_r}{u_\theta} = \frac{-\sin(\theta_{goal\ or\ landmark})}{\cos(\theta_{goal\ or\ landmark})} \rightarrow \theta_{DTG\ or\ DTL}^{ref} = \tan^{-1}(-\frac{u_r}{u_\theta}) \tag{10}$$

2.6 Behavior Arbitration Layer

Having calculated three reference orientations, one for obstacle avoidance and the other for goal or landmark tracking, the resultant reference orientation of robot should be calculated by the combination of these references, which will be done by the behavior arbitration layer being central to the success or failure of the algorithm. The three outputs might be conflicting. However, the behavior arbitration should combine them in such a way that both obstacle avoidance and goal tracking are partially fulfilled. Dynamic weights that are calculated from the geometric relations between the robot, goal and obstacle are assigned to the outputs of the two layers, and the overall reference orientation is calculated by the addition of them.

$$\theta^{ref} = \frac{\alpha \cdot \theta_{OA}^{ref} + \beta \cdot \theta_{DTG\ or\ DTL}^{ref}}{2} \tag{11}$$

The weight α and β in equation (11) are complementary, $\alpha + \beta = 1$, and are calculated by considering θ'_{turn} the angle between the direction of velocity vector of the robot repelling force vector on the robot in Fig. 5. This classical method might be working well in the simulations where inertia of the robot is probably not considered, time delay between successive samples is small, and there is no noise. The problem of this method occurs as follows : At the instant shown, the robot is moving parallel to the obstacle, in which case the output of OA layer has no importance since its weight is zero. However, due to the position of the goal or landmark, the DTG or DTL layer produces a large output as orientation reference, and since its weight is, $\beta = 1$, the resulting reference orientation for the robot will be large and in the direction of the goal. This will cause the robot to turn fast in the reference direction. If the robot has large inertia and the sampling time of sensor is also large, this situation will cause the robot to hit the obstacle, or at best will cause severe directional oscillations.

The new turning angle θ_{turn} is calculated as the difference between the angle to obstacle and angle to goal point, as shown in Fig. 5 for goal point 1.

$$\alpha = 1 - \beta = \begin{cases} 1 & if\ \ \theta_{turn} = 0 \\ \quad \vdots & \\ 0 & if\ \ \theta_{turn} \geq \pi / 2 \end{cases} \tag{12}$$

This formulation shows that the weight of obstacle will be large since the turning angle is small according to classical method, and the robot will continue to turn around it until it comes to a position where the goal is directly in front of it, like goal point 2.

Another problem, which is also inherent in potential field method, is that the robot can not reach the goal if it is located too close to an obstacle, due to large repelling forces in the vicinity of obstacle resulting in large reference orientation changes calculated by the OA layer. This can be relieved if the behavior arbitration layer takes into account obstacle and goal distances.

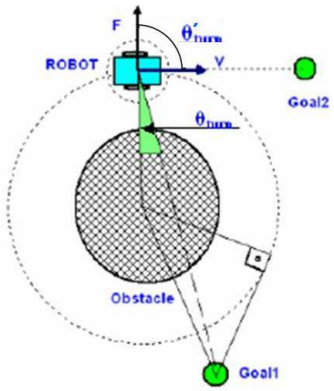

Fig. 5. Turning angle in behavior arbitration

If goal is close to an obstacle, since the robot is approaching the obstacle, it will be repelled and forced to go away even though the goal is in front of it. In such a case, if behavior arbitration layer sets the weight of OA layer to zero, the robot will be able to reach the goal. This approach only relieves the problem, since for example, if the goal were on the other side of the obstacle, the distance to the goal would not be smaller while the robot is turning around the obstacle, and it will enter an infinite loop turning around the obstacle forever and never reaching the goal.

The main idea of the design of a behavior arbitration layer is used to the 3-10-2 neural networks. This system has θ_{turn}, $\bar{d}_{obs}$ (the average distance between obstacles and robot) and $d_{goal\ or\ landmark}$ (the distance between the goal or landmark position and robot) as the input, and the output is α (the

weight parameter of θ_{OA}^{ref}) in Fig. 6. First, this system can be learning as equation (12). Also, a behavior arbitration layer can be possible continuously through reinforcement learning.

2.7 Low-level Motion Control Layer

Now that the reference orientation of robot has been calculated, it is time to drive the robot according to this reference for safe navigation. The obstacle avoidance and drive toward goal or landmark parallel layers give only the reference orientation of the robot and does not tell anything related to speed of the robot. In the original form of potential field method, the velocity is taken to be proportional to the magnitude of the resultant force. If the attractive force is chosen to be proportional to the square of the distance from robot to goal or landmark point, the velocity becomes proportional to the distance to the goal or landmark. Accordingly, robot should drive fast when it is far away, and should naturally stop when it reaches the goal. There is of course an upper limit for the velocity for safe navigation. A different approach is implemented in this work, and it will be discussed later in this chapter.

Having the reference orientation, θ^{ref}, and velocity, V^{ref}, a motion controller can be implemented according to the local coordinate system of the robot to calculate the individual wheel velocities assuming the robot is as defined in section A.

Similar to the θ^{ref} calculations, separate controls are implemented in r^{ω}, θ^{ω} components. Proportional controller is used instead of sliding mode controller. First, errors are calculated as follows.

$$e_r = r^{ref} - r, \quad e_\theta = \theta^{ref} - \theta \tag{13}$$

Where r is the distance from robot to goal or landmark point, $r^{ref} = 0$, since the desired distance is zero; θ is reference orientation generated by behavior arbitration layer, and θ^{ref} is zero. Fig. 6 illustrates the parameters.

Controls are chosen as

$$\begin{aligned} u_r &= (V_R + V_L)/2 \qquad \dot{r} = u_r \\ u_\theta &= (V_R - V_L)/L \text{'} \quad \dot{\theta} = u_\theta \end{aligned} \tag{14}$$

Where V_R and V_L are right and left wheel velocity references to be used by the controller on the robot, L is the robot's interwheel distance. Then the controls are calculated using a proportional controller.

$$\begin{aligned} u_r &= K_{pr} \cdot e_r \\ u_\theta &= K_{p\theta} \cdot e_\theta \end{aligned} \tag{15}$$

Where K_{pr} and $K_{p\theta}$ are proportional control gains. Finally, wheel velocity references can be calculated.

$$\begin{aligned} V_R^{ref} &= u_r - L \cdot \frac{u_\theta}{2} \\ V_L^{ref} &= u_r + L \cdot \frac{u_\theta}{2} \end{aligned} \tag{16}$$

3. Simulations

In this section, simulation results are shown to confirm the validity of the proposed method. The goal of the simulation is to successfully reach for each robot to the goal position. In Fig. 6, the first experiment is implemented in the environment, which is a circular obstacle between a robot and a goal. This experiment is shown the optimal path and the reduced oscillation when a robot pass a circular obstacle. In Fig. 7, the second experiment is implemented in the environment, which is a U-shaped obstacle between a robot and a goal. This experiment is shown that a robot overcomes trap situation. In Fig. 8, the third experiment is implemented in the environment, which is a combination of circular obstacles. This experiment is shown that a robot overcomes no passage between closely spaced obstacles. In Fig. 9, the forth experiment is implemented in the environment, in which another robot and combination of circular obstacles exist. This experiment is shown the performance in dynamic environment. The proposed algorithm for navigation is shown the good performance through these experiments.

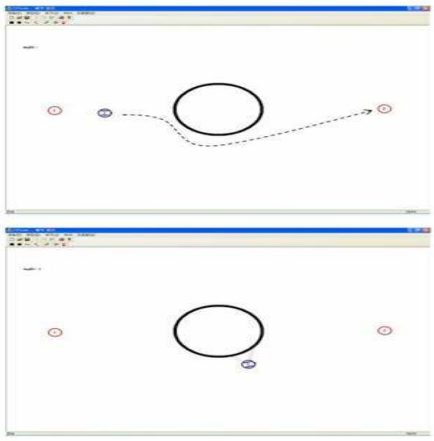

Fig. 6. 1st experiment

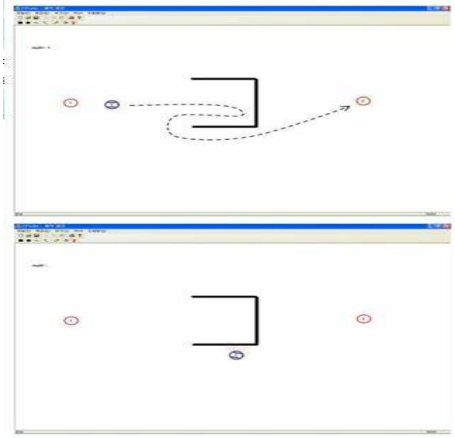

Fig. 7. 2nd experiment

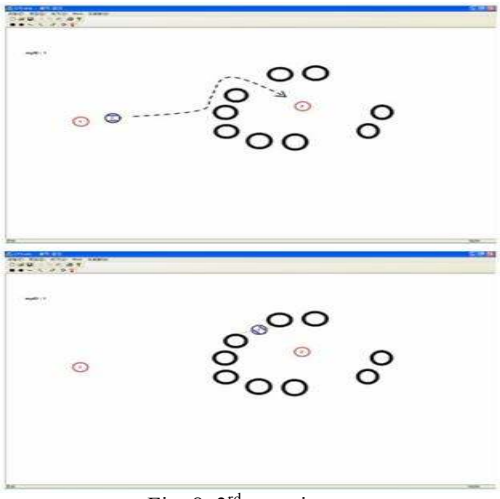

Fig. 8. 3rd experiment

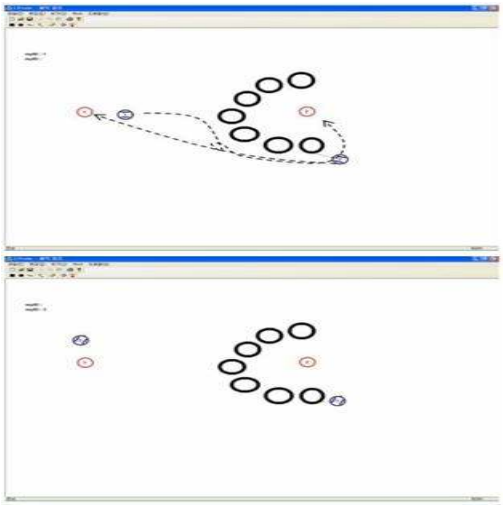

Fig. 8. 4th experiment

4. Conclusion

In this work, a potential field based autonomous mobile robot navigation algorithm is implemented on a computer simulation system. Modifications for improvements, and better performance are suggested in the light of experimental results. The results are evaluated and the shortcomings of experimental setup and algorithms used are stated. The performance of the system is satisfactory proposing solutions to the problematic cases of especially potential field based robot navigation algorithms and implementations, such as reducing oscillations, ability to pass through closely spaced obstacles, and navigation among complex shaped obstacles. In our application of the potential model for mobile robots, however, we still encounter the problem of concavities in the potential field. In future work, we will study on potential field model more suitable for mobile robot navigation tasks and apply the proposed navigation algorithm to real robot system.

References

[1] L. C. A. Pimenta and A. R. Fonseca, "Robot Navigation Based on Electrostatic Field Computation", *IEEE Transaction on Magnetics*, vol. 42, no. 4, april 2006.

[2] J. Ren, K. A. McIsaac and R. V. Patel, "Modified Newton's Method Applied to Potential Field-Based Navigation for Mobile Robots", *IEEE Transaction on Robotics*, vol. 22, no. 2, April, 2006.

[3] J. Borenstein, Y. Koren, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", *Proceeding of the IEEE International Conference on Robotics and Automation*, California, April 1991.

[4] R.C. Arkin, *Behavior Based Robotics*, MIT Pres, 1998.

[5] R. Daily and D. M. Bevly, "Harmonic Potential Field Path Planning for High Speed Vehicles", *American Control Conference 2008*, Seattle, Washington, USA, 2008.

[6] S. Yannier, A. Onat, A. sabanovic, "Basic Configuration for Mobile Robots", *International Conference on Industrial Technology, ICIT03*, Maribor, Slovenia, 2003.

[7] Thomas Braunl, *Embedded Robotics*, Springer-Verlag, Berlin, Heidelberg 2003.

Kwang-Min Jung

He received his B.S degree in the Department of Electrical and Electronics Engineering from Chung-Ang University, Seoul, Korea. He is currently Master course in the School of Electrical and Electronics Engineering from Chung-Ang University. His research interests include machine learning, multi agent robotic system, Evolutionary Computation, Evolutionary Robot, etc.

Kwee-Bo Sim

He received his B.S. and M.S. degrees in the Department of Electronic Engineering from Chung-Ang University, Korea, in 1984 and 1986 respectively, and Ph.D. degree in the Department of Electronics Engineering from The University of Tokyo, Japan, in 1990. Since 1991, he is currently a Professor. His research interests include artificial life, emotion recognition, ubiquitous intelligent robot, intelligent system, computational intelligence, intelligent home and home network, ubiquitous computing and Sense Network, adaptation and machine learning algorithms, neural network, fuzzy system, evolutionary computation, multi-agent and distributed autonomous robotic system, artificial immune system, evolvable hardware and embedded system etc. He is a member of IEEE, SICE, RSJ, KITE, KIEE, KIIS, and ICROS Fellow.