Advances in Engineering Software 41 (2010) 1179–1186

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Navigation of mobile robots in the presence of obstacles

R. Abiyev, D. Ibrahim *, B. Erin

Near East University, Department of Computer Engineering, Mersin 10, Turkey

ARTICLE INFO

Article history:
Received 20 May 2010
Received in revised form 19 July 2010
Accepted 1 August 2010

Keywords:
Robot navigation
Obstacle avoidance
Robot navigation software
Fuzzy control
Mobile robotics
Robot simulation

ABSTRACT

Robot navigation is one of the basic problems in robotics. In general, the robot navigation algorithms are classified as global or local, depending on surrounding environment. In global navigation, the environment surrounding the robot is known and the path which avoids the obstacle is selected. In local navigation, the environment surrounding the robot is unknown, and sensors are used to detect the obstacles and avoid collision. In the past, a number of algorithms have been designed by many researchers for robot navigation problems. This paper presents software simulation of navigation problems of a mobile robot avoiding obstacles in a static environment using both classical and fuzzy based algorithms. The simulation environment is a menu-driven one where one can draw obstacles of standard shapes and sizes and assign the starting and ending points of the mobile robot. The robot will then navigate among these obstacles without hitting them and reach the specified goal point.

1. Introduction

The task of the navigation system is to guide the robot towards the goal point without a collision with obstacles. Design of a fast and efficient procedure for navigation of mobile robots in the presence of obstacles is one of the important problems in robotics. Given the initial and final configurations of a mobile robot, the navigation algorithm must be able to determine whether there is a continuous motion from one configuration to the other, and find such a motion if one exists.

Obstacle avoidance is the primary requirement for any autonomous robot. Designing a robot requires the integration and coordination of many sensors and actuators. In general, the robot acquires information about its surrounding through various sensors mounted on the robot. Usually, multiple sensors, such as infrared sensor, ultrasonic sensor, laser range finder, touch sensor and camera can be used to detect the presence of obstacles.

Many researchers developed various obstacle avoidance algorithms in the past. In general, the research in this field can be classified into two major areas: the *global path planning* [1,2] and the *local motion planning* [3,4]. In global path planning, the environment surrounding the robot and the position of obstacles are well known in advance, and the robot is required to navigate to its destination by avoiding the obstacles. The complete robot path in such applications can be calculated from the prior knowledge of the coordinates of the starting point, the destination point, and the obstacles. On the other hand, the local motion planning methods dynamically guide the robot according to the locally sensed obstacles, which requires less prior knowledge about the environment. Therefore, the local motion planning methods are more suitable and practical for mobile robots, since the environment is too complicated to be known precisely and may also be time-varying.

Recently there have been many interesting research work in the literature for navigation of mobile robots [1–14]. The earliest algorithms for robot navigation were developed in a completely known environment, filled with stationary obstacles whose positions were also known. In these algorithms, widely utilized methods are the *artificial potential field* (APF) approach [4–6], the *distance function* (DF) approach [7], *vector field histogram* (VFH) [8] and VFH+ [9] techniques, *dynamic window approach* (DWA) [10], and *rule based* (RB) approach [11,12].

The APF approach [6] involves modelling the robot as a particle in space, acted on by some combination of attractive and repulsive fields. In this technique obstacles and the goal are modelled as charged surfaces, and the net potential generates a force on the robot. These forces push the robot away from the obstacles, while pulling it towards the goal. The robot moves in the direction of greatest negative gradient in the potential. Despite its simplicity this method can be effectively used in many simple environments.

The VFH [8] uses local maps consisting of occupancy grids. These grids are transformed into maps where the occupied cells contain peaks, and open spaces contain troughs. The robot is then drawn towards the troughs. Extended VFH+ method includes the robots kinematic contstraints when generating the troughs and

* Corresponding author.
E-mail addresses: rahib@neu.edu.tr (R. Abiyev), dogan@neu.edu.tr (D. Ibrahim), besime@neu.edu.tr (B. Erin).

0965-9978/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2010.08.001

peaks. Only open spaces which can be entered by the robot while travelling at its current speed are made into troughs. If no open spaces are available, the robot's speed is reduced until an opening shows up. If no opening is found, the robot stops.

The DWA [10] is a function of the robot's velocity space. By considering all the possible velocities attainable by the robot and applying those that lead away from any impending collisions, the robot can navigate at fast speeds through its environment [10].

The rule based approch commonly uses simple if–then rules [11]. But simple rules are very limited to the environments in which they are built. Fuzzy sytems have been proven to produce better performance than simple rules in unknown environments. Many researchers have studied the robot navigation problems in unknown environments [12–20]. Different infrared and ultrasonics sensors, and computer vision have been used in these algorithms, and each of these algorithms show their feasibilities in different application areas. For a mobile robot to move in a completely unknown environment, the robot must have the capability of obstacle detection, self-positioning, and environment recognition. In mobile robot algorithms the navigation procedure is applied iteratively until the robot reaches its final destination. Robots do have dynamic characteristics and such characteristic must be taken into account when using a path planning algorithm. In this paper the positions of the mobile robot and the goal that the robot must reach are assumed to be known in advance, but the location of obstacles are unknown. The goal point is specified by its coordinates. During the actual move, the robot uses the information taken from its sensors in order to avoid the obstacles. To reach the goal, robot uses the distance to an obstacle and direction angle as the main parameters. The direction angle of the robot's movement is determined by the direction angle of the goal and sensor signals. The fuzzy knowledge base is developed for navigation of the robot. In this rule base the task is to achieve the goal, while avoiding the obstacles. The conditions for obstacle avoidance are described inside the knowledge base. The advantages of the proposed navigation scheme are that less local information is required than in most other techniques. The knowledge base includes the fuzzy if–then rules. The construction of these rules are simple and flexible.

In this paper, the simulation of potential field method, vector field histogram, local navigation, and fuzzy based navigation have been carried out for robot navigation problems. The development of such a system will allow better understanding of the problems of obstacle avoidance and their solution mechanisms, and also help us make comparison of existing navigation algorithms. We have designed a simulator for robot path planning which basically draws a path for a mobile robot in the presence of obstacles. This path may be modified with respect to changes to various parameters for each of the different algorithms implemented by the software. The start and goal positions may also be modified and the simulation can run as many times as desired.

The paper is organised as follows: In Section 2, potential field, vector field histogram plus, local navigation and fuzzy navigation algorithms are described briefly. Section 3 describes features of the simulation software. Section 4 includes simulation results and analysis of obtained results. Finally, Section 5 discusses conclusions and suggestions for future work.

2. Mobile robot navigation algorithms

In this paper mobile robot navigation using potential field method, vector field histogram, local and fuzzy navigation algorithms are modelled to investigate the movement of a mobile robot to its destination, while avoiding any obstacles on its way. Path planning methods for mobile robots are based on the idea of finding an optimal and smooth path consisting of many points close enough to each other, thus avoiding obstacles. It is possible to have a path consisting of spline curve segments. The software which is the subject of this paper utilizes four methods as described below:

2.1. Potential field method

Robot dynamic characteristics and navigation law are important in path planning [21–23], where information about location of obstacles is used to determine the desirable path. One of the path determination methods used in the simulator software which is the main topic of this paper is the *potential field* method (PFM) [27]. The philosophy of the potential field approach is that the mobile robot moves in a field of forces. The goal position to be reached is an attractive potential while each obstacle generates a repulsive potential. A potential field can be viewed as an energy field and so its gradient at each position is a force. Potential fields can be applied locally while path determination, trajectory planning and control steps are achieved in one step in real time. The idea of obstacles exerting virtual repelling forces towards a robot, while the target generates a virtual attractive force uses a similar concept that takes into consideration the robot's velocity in the vicinity of obstacles. In one example called the Brooks implementation [27], if the magnitude of the sum of repulsive forces exceeds a certain threshold, the robot stops, turns into the direction of the resultant force vector, and moves on. This method also requires the robot to stop and thus is not suited for teleautonomous operation. The theory of the potential field method is given below briefly.

A potential, $\phi(r)$, is defined by the Laplace equation $\nabla\phi = 0$ in a closed region, Ω, of continuous, equal connectivity. The boundary of Ω, $\partial\Omega$, does not have to be connected. It includes the surfaces of all obstacles and the goal point. $\phi(r) = \phi_1$ at the surfaces of obstacles and $\phi(r) = \phi_0$ at the goal point. There are no local minimas on $\phi(r)$. Nevertheless, the exponential decay of the field from any point leads to areas where the magnitude of the gradient on ϕ, $|\nabla\phi|$, is very small while the range of $|\nabla\phi|$ may be very large. The field decays rapidly near the goal, and far from the goal there is only a slight change in the field.

In this algorithm, the potential value is calculated for each point on the grid by using the Laplaces's equation where the value of the field at the goal point is set to the value of -2^{127} (the smallest negative number that can be represented by the compiler) and boundary points are to zero.

The Laplace equation in two dimensions is represented on equally spaced and connected grid by the following partial differential equation:

$$\phi_{(i,j)} = \frac{\phi_{(i+1,j)} + \phi_{(i-1,j)} + \phi_{(i,j+1)} + \phi_{(i,j-1)}}{4} \quad (1)$$

where i is position on the grid in the x direction, j is position on the grid in the y direction

The potential value given above for each grid point is referred to as the gridValue, a two-dimensional array. gridSize is the parameter that determines the length and width of the grid cell.

Field values are calculated for any point in the workspace by using linear interpolation.

Passing two parameters a and b into the function field, we use the following sets of equations to determine the field value at any given point that is anywhere on the canvas (i.e. can be anywhere in-between the grid). In this algorithm, *gridSize* is the length and width of each grid cell. *numberofGridLinesX* and *numberofGridLinesY* are the number of grid cells along the X and the Y axes. dx and dy are the differential values for x and y values. sx and sy are simply approximations. *bot*1 and *bot*2 are calculated values which eventually are used to calculate the *potential field* value at any