Robotics and Autonomous Systems 60 (2012) 1123–1134

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

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

A novel obstacle avoidance algorithm: "Follow the Gap Method"

Volkan Sezer*, Metin Gokasan

Faculty of Electrical & Electronics, Istanbul Technical University, Ayazaga Kampusu 34469, Maslak-Istanbul, Turkey

ARTICLE INFO

Article history:
Received 11 January 2012
Received in revised form
10 May 2012
Accepted 25 May 2012
Available online 2 June 2012

Keywords:
Obstacle avoidance
Autonomous robots
Path planning
Dynamic obstacle
Nonholonomic constraints
Vehicle kinematics

ABSTRACT

In this paper, a novel obstacle avoidance method is designed and applied to an experimental autonomous ground vehicle system. The proposed method brings a new solution to the problem and has several advantages compared to previous methods. This novel algorithm is easy to tune and it takes into consideration the field of view and the nonholonomic constraints of the robot. Moreover the method does not have a local minimum problem and results in safer trajectories because of its inherent properties in the definition of the algorithm. The proposed algorithm is tested in simulations and after the observation of successful results, experimental tests are performed using static and dynamic obstacle scenarios. The experimental test platform is an autonomous ground vehicle with Ackermann steering geometry which brings nonholonomic constraints to the vehicle. Experimental results show that the task of obstacle avoidance can be achieved using the algorithm on the autonomous vehicle platform. The algorithm is very promising for application in mobile and industrial robotics where obstacle avoidance is a feature of the robotic system.

1. Introduction

Robot navigation refers to the robot's ability to safely move towards the goal using its knowledge and the sensorial information of the surrounding environment. Given a map and a goal location, path planning involves finding a geometric path from the actual location of the robot to the goal/target. This type of planning is referred to as static path planning due to the fact that the map used in the algorithm is static, and not updated dynamically based on new information [1,2]. There are many studies involved with static planning, such as, Probabilistic Roadmaps (PRMs) [3], Rapidly Exploring Random Trees (RRT) [4], Generalized-Sampling Based Methods [5], Visibility Graphs [6], Voronoi Diagrams [7] and cell decomposition methods [8]. Beside these, studies on finding an optimal solution for nonholonomic robots, can be found in [9,10]. The common ground of all these path planning methods is the necessity for a map of the whole workspace.

Obstacle avoidance is different from static path planning with its aim of avoiding unexpected obstacles along the robot's trajectory. In other terms, it shapes up the trajectory of the path planners as a dynamic path planning approach. Considering the needs of autonomous robot control, it is obvious that detecting and avoiding obstacles in real time is crucial for the performance. For this reason, many researchers have turned their attention to the obstacle avoidance problem developing interesting real-time approaches for avoiding unexpected static and dynamic obstacles.

Several methods have been proposed for obstacle avoidance starting from the 1980s till now. The earliest versions of these obstacle avoidance methods are bug algorithms [11]. These algorithms follow the easiest common sense approach of moving directly towards the goal, until an obstacle is found, in which case the obstacle is contoured until moving towards the goal is possible again. The trajectories of bug algorithms are sometimes very long and the robot is prone to move too close to obstacles.

Another common approach is the artificial potential field (APF) method [12]. In the APF approach, the obstacles to be avoided are represented by a repulsive artificial potential and the goal is represented by an attractive potential, so that a robot reaches the goal without colliding with obstacles. Main drawbacks of the APF method are summarized in [13] and local minima are the most dangerous problem of APF. This happens when all the vectors from obstacles and the goal point cancel each other out and make it impossible for the robot to reach the goal. There are a great number of studies focusing on avoiding local minima. The first solution method comes from definition of the potential function by specifying a function with no local minima like the harmonic potential field approach [14]. However in this approach, the robot must know the map of the whole environment and this contradicts reactiveness and local planning properties of obstacle avoidance. Other approaches for local minimum avoidance involve some practical ad-hoc solutions, such as those proposed in [15–18], but none of these approaches can offer an ultimate guarantee to avoid local minima.

The Virtual Force Field method (VFF) [19] uses a two-dimensional Cartesian histogram grid for obstacle representation. Each cell in the histogram grid holds a certainty value that

* Corresponding author.
E-mail address: sezervolkan@gmail.com (V. Sezer).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.05.021

represents the confidence of the algorithm in the existence of an obstacle at that location. After that, APF is applied to the histogram grid, therefore the problems of the APF method still exist in the VFF method.

The Vector Field Histogram (VFH) [20] uses a two-dimensional Cartesian histogram grid like in VFF. After that, the histogram grid is reduced to a one dimensional polar histogram that is constructed around the robot's momentary location. In the second stage, the algorithm selects the most suitable sector from among all polar histogram sectors with a low polar obstacle density, and the steering of the robot is aligned with that direction. This method is very much goal oriented since it always selects the sector which is in the same direction as the goal, but the selected sector can be the wrong one in some cases. This method also does not consider nonholonomic constraints of robots like the other methods mentioned above. More information about the concepts for dynamic obstacle avoidance can be found in [21].

In this paper, a novel approach called the "Follow the Gap Method" (FGM) is presented as an obstacle avoidance algorithm. FGM ensures safety by directing the robot into the center of the maximum gap as much as possible while providing the reach of the goal point. FGM calculates a gap array around the robot, selects the appropriate gap, calculates the best heading vector through the gap, using specific geometric theorems and finally calculates the final angle considering the goal point. An important advantage of FGM over other methods is that it results in safer trajectories which will be shown in simulation results. Moreover FGM accounts for the nonholonomic and the field of view constraints of the robot. Another important advantage is that it does not have a local minimum problem and finally it is easy to tune with only one tuning parameter. Simulations and real tests are performed using the Ackerman steering ground vehicle platform. Successful results are achieved in simulations and experimental tests which will be shown in Sections 4 and 6.

The paper is organized as follows. Section 2 gives the definition of the problem. Section 3 introduces the new obstacle avoidance methodology, "Follow the Gap Method" in three subsections. Section 3.1 illustrates the calculation of the gap array and finding the maximum gap, Section 3.2 shows calculation of the gap center angle and Section 3.3 gives the calculation of the final heading reference angle for obstacle avoidance. Section 4 shows the simulation results. Section 5 gives information about the experimental set-up which is an autonomous ground vehicle. Section 6 shows the experimental results with this novel algorithm. Conclusions are given in Section 7 pointing to future directions of research.

2. Problem definition

Suppose that independent of the geometry of the robot and obstacles they are considered to be circular objects with minimum radius to include all physical boundaries. Cartesian coordinate space is used for calculations. The location of the robot and its radius values are given by the tuple $(X_{rob}, Y_{rob}, r_{rob})$ and similarly the center location and radius of the obstacles are given by $(X_{obsn}, Y_{obsn}, r_{obsn})$ for the nth obstacle. The following assumptions are made to define the problem.

(i) The robot field of view is constrained by two rays with left ϕ_{fov_l} and right ϕ_{fov_r} angles and a distance constraint with d_{fov}. The robot does not have prior information about obstacles.
(ii) The robot has a nonholonomic constraint which is represented in a summarized form as a minimum turn radius $r_{\min}$.
(iii) All the coordinates/locations and object boundaries are measurable and constraint values ϕ_{fov_l}, ϕ_{fov_r}, d_{fov}, $r_{\min}$ are previously calculated according to the sensor arrangement and the geometry of the vehicle.

Using these assumptions, the aim of the obstacle avoidance algorithm is to find a purely reactive heading reference, in order to achieve the goal coordinates while avoiding obstacles with as large distance as possible, considering the measurement and nonholonomic constraints.

Obstacle avoidance algorithms should work cooperatively with global planners. The obstacle avoidance algorithm is activated and global planner commands are disabled when an unexpected obstacle scenario is met. The goal point of the obstacle avoidance algorithm is given by the global planner. This paper's theme is obstacle avoidance. Since no prior information is available to the robot, the nature of the obstacle algorithm should be reactive because, coordinates of any obstacle may change at any time and it can not be known previously. The algorithm must compute just the next action in every instant, based on the current context. In other words, any classical optimization algorithm like dynamic programming which calculates the reference heading values from goal to initial coordinates is not possible for real time applications due to the unknown sequencing of the obstacles during the journey. We have tried to solve this problem using a new heuristic and fusing function between maximum gap and goal point, in order to calculate the reference heading angle in each sampling time reactively.

2.1. Point robot approach

As it was given in the problem definition, it is assumed that the robot and obstacles are circular objects. In order to simplify the problem given previously, the radius of the robot is added to the obstacle radius as illustrated in Fig. 1.

The obstacle avoidance problem for a circular robot is now equivalent to obstacle avoidance for a point robot in Cartesian space. This guarantees a trajectory without collision if any gap is calculated. Otherwise, a collision risk exists because of the physical dimensions of the robot.

2.2. Calculation of "distance to obstacle"

Distance to obstacle boundary value will be used for heading angle calculations during the algorithm. For this reason, the formal definition of distance between the robot and the nth obstacle border (d_n) is given here. Fig. 2 illustrates the parameters of the circular robot and the nth obstacle.

Using the distance to obstacle geometry in Fig. 2b, d_n is found by using the Pythagorean theorem as illustrated in Eq. (1).

$$\begin{aligned} &d = \sqrt{(X_{obsn} - X_{rob})^2 + (Y_{obsn} - Y_{rob})^2} \\ &d_n^2 + (r_{obsn} + r_{rob})^2 = d^2 \\ &\Rightarrow d_n = \sqrt{(X_{obsn} - X_{rob})^2 + (Y_{obsn} - Y_{rob})^2 - (r_{obsn} + r_{rob})^2}. \end{aligned} \tag{1}$$

In the rest of the paper, the circular robot is shown as a point robot whereas each obstacle is enlarged with the robot's radius.

3. Follow the gap method

The Follow the Gap method is based on the construction of a gap array around the vehicle and calculation of the best heading angle for heading the robot into the center of the maximum gap around, while simultaneously considering the goal point. These two aims are considered simultaneously by using a fusing function. The algorithm can be divided into three main parts as illustrated in Fig. 3.

The steps shown in Fig. 3 are explained in further detail in the rest of the paper.