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).

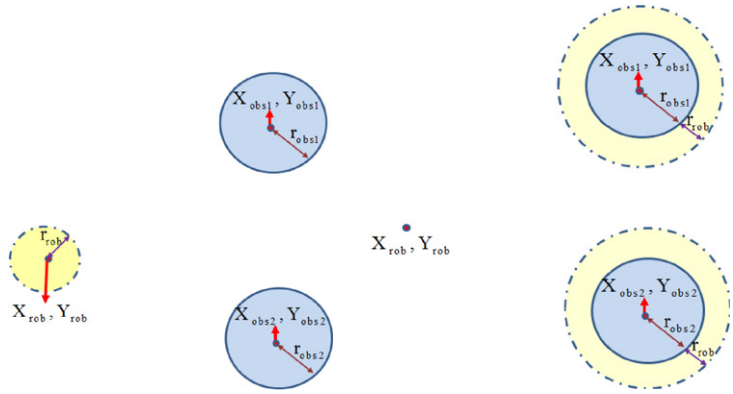
$$\begin{aligned} &d = \sqrt{(X_{obsn} - X_{rob})^2 + (Y_{obsn} - Y_{rob})^2} \\ &d_n^2 + (r_{obsn} + r_{rob})^2 = d^2 \\ &\Rightarrow d_n = \sqrt{(X_{obsn} - X_{rob})^2 + (Y_{obsn} - Y_{rob})^2 - (r_{obsn} + r_{rob})^2}. \end{aligned} \tag{1}$$

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

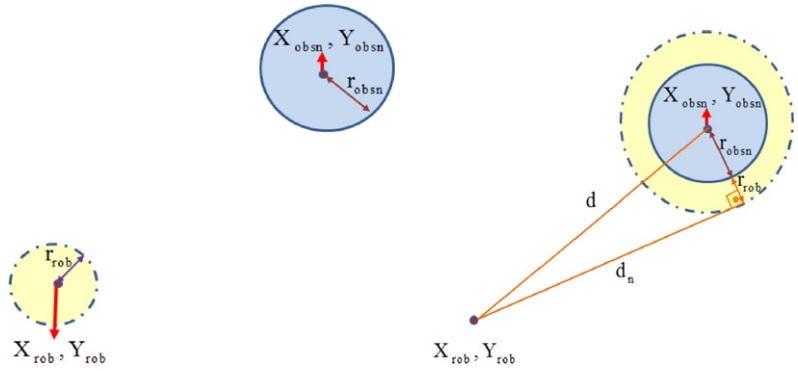
The Follow the Gap method is based on the construction of a gap array around the vehicle and calculation of the best heading angle for heading the robot into the center of the maximum gap around, while simultaneously considering the goal point. These two aims are considered simultaneously by using a 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.



(a) Circular robot with circular obstacles. (b) Point robot with enlarged obstacles.

Fig. 1. Obstacle enlargement for point robot representation.



(a) Circular robot and circular obstacle parameters.

(b) Distance to obstacle geometry.

Fig. 2. Distance to obstacle parameters and geometry.

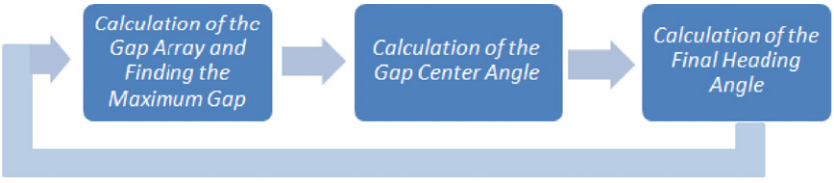


Fig. 3. Steps of the Follow the Gap method.

3.1. Calculation of the gap array and finding the maximum gap

After using the point robot approach, each obstacle around the robot is represented with two parameters. These are the border angle values of the obstacles. A sample of this representation is shown in Fig. 4. (ϕ_{obsi_l} and ϕ_{obsi_r} denote the left and right border angle values of the ith obstacle respectively.)

In order to calculate the gap array, we need two more border values in addition to the borders of obstacles. Fig. 5 shows the evaluation of the gap borders. These border values are found by using the field of view of the robot and the movement constraints. Nonholonomic constraints are very common when writing the kinematic constraints for certain kind of robots. Since our experimental platform is a nonholonomic ground vehicle here, we consider the nonholonomic constraints as a movement constraint. Roughly speaking, a mechanical system is said to be nonholonomic if the vector space of the possible motion directions in a given configuration is restricted such that the restriction can not be converted into an algebraic relation between configuration variables [22]. The physical meaning of this constraint for the vehicle, which is used in simulations and experimental studies,

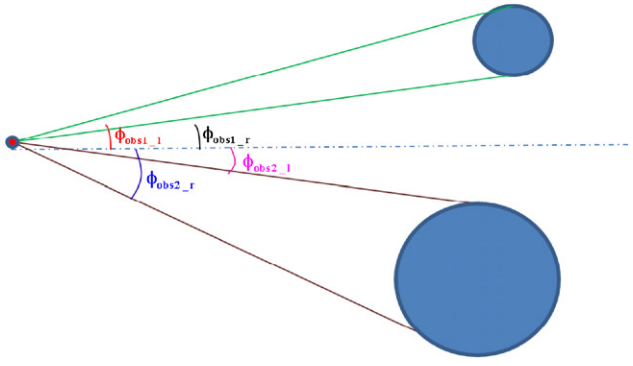
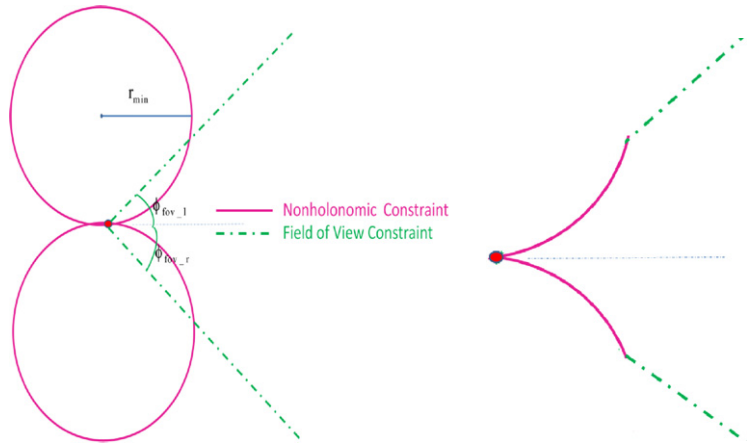


Fig. 4. Obstacle representation.

can be observed in its inability to move sideways. Instead of directly moving sideways, the vehicle has to follow an arc to arrive at a lateral coordinate. There have to be both longitudinal and lateral displacements at the same time in a car-like vehicle. Fig. 5 illustrates the field of view and the nonholonomic movement



(a) Field of view and nonholonomic movement constraint separately.

(b) Gap border.

Fig. 5. Gap border evaluation.

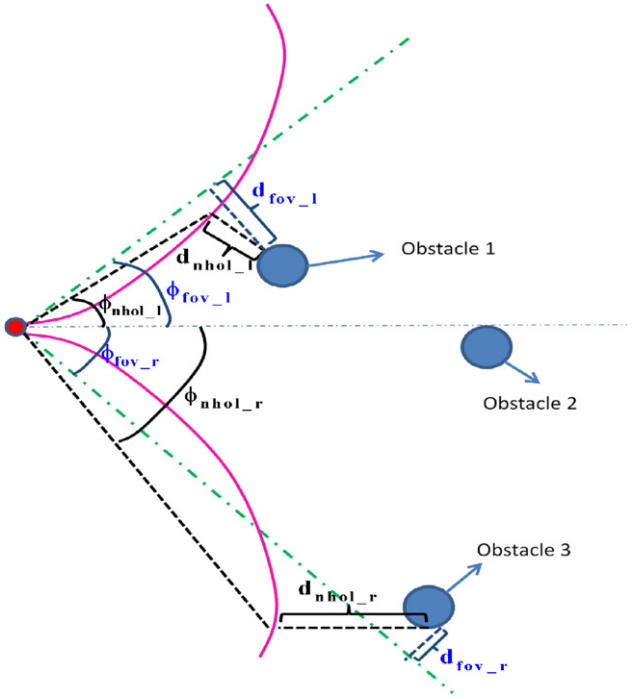


Fig. 6. Gap border parameters.

constraint which construct the border values for the vehicle. More information about nonholonomy can be found in [22,23].

Fig. 5a illustrates both the field of view and nonholonomic movement constraint separately. Circles represent the minimum radius ($r_{\min}$) turn of the vehicle coming from the nonholonomic constraint and the dashed lines are for the field of view. Fig. 5b illustrates the region in which the vehicle can view obstacles and reach any point in spite of its nonholonomic constraints. In order to understand which boundary is active for a boundary obstacle, decision rules are illustrated in Eq. (2). The parameters of interest are given in Fig. 6.

$$\begin{aligned} d_{nhol} < d_{fov} &\Rightarrow \phi_{\lim} = \phi_{nhol} \\ d_{nhol} \geq d_{fov} &\Rightarrow \phi_{\lim} = \phi_{fov} \end{aligned} \tag{2}$$

where

$\phi_{\lim}$: Gap border angle. ($\phi_{\lim\_l}$: For the left side of the vehicle. $\phi_{\lim\_r}$: For the right side of the vehicle.)

ϕ_{nhol}: Border angle coming from nonholonomic constraint. (ϕ_{nhol_l}: For the left side of the vehicle. ϕ_{nhol_r}: For the right side of the vehicle.)

ϕ_{fov}: Border angle coming from field of view. (ϕ_{fov_l}: For the left side of the vehicle. ϕ_{fov_r}: For the right side of the vehicle.)

d_{nhol}: Nearest distance between nonholonomic constraint arc and obstacle border. (d_{nhol_l}: For the left side of the vehicle. d_{nhol_r}: For the right side of the vehicle.)

d_{fov}: Nearest distance between field of view line and obstacle border. (d_{fov_l}: For the left side of the vehicle. d_{fov_r}: For the right side of the vehicle.)

In Fig. 6, d_{fov}, ϕ_{fov}, ϕ_{nhol} and d_{nhol} expressions are shown for both the left side and the right side of the vehicle. Obstacle 1 and obstacle 3 are boundary obstacles for the left and right side of the vehicle respectively. In an obstacle configuration as illustrated in Fig. 6, using Eq. (2), the gap border angle for the left side of the vehicle ($\phi_{\lim\_l}$) comes from the nonholonomic constraint (ϕ_{nhol_l}) and the gap border angle for the right side of the vehicle ($\phi_{\lim\_r}$) comes from the field of view constraint (ϕ_{fov_r}).

After the representation of the gap border and obstacle borders, the gap array can be generated. There have to be $N+1$ gaps for N obstacles. $N+1$ elements of gap array are illustrated below in terms of the previous definitions.

$$\begin{aligned} Gap[N+1] = [&(\phi_{\lim\_l} - \phi_{obs1\_l})(\phi_{obs1\_r} - \phi_{obs2\_l}) \cdots (\phi_{obs(n-1)\_r} \\ &- \phi_{obs(n)\_l})(\phi_{obs(n)\_r} - \phi_{\lim\_r})]. \end{aligned} \tag{3}$$

The maximum gap is the maximum of the gap array members and is selected for the second step of the algorithm. If more than one maximum gap with the same value exists, the algorithm selects the first calculated one.

3.2. Calculation of the gap center angle

After finding the maximum gap from the previous section, it is essential to find the gap center angle (ϕ_{gap_c}) which ensures that the robot moves through the center of the maximum gap. This is the angle of the median vector from robot to the line between the obstacles creating/causing the maximum gap. The gap center angle is shown in Fig. 7.

In order to find the gap center angle, the Cosine and Apollonius theorems are used. The details of these theorems are given in Appendix A.

It can be seen from Fig. 7 that there is a triangle with dashed lines and the aim is to find the angle of median vector of this

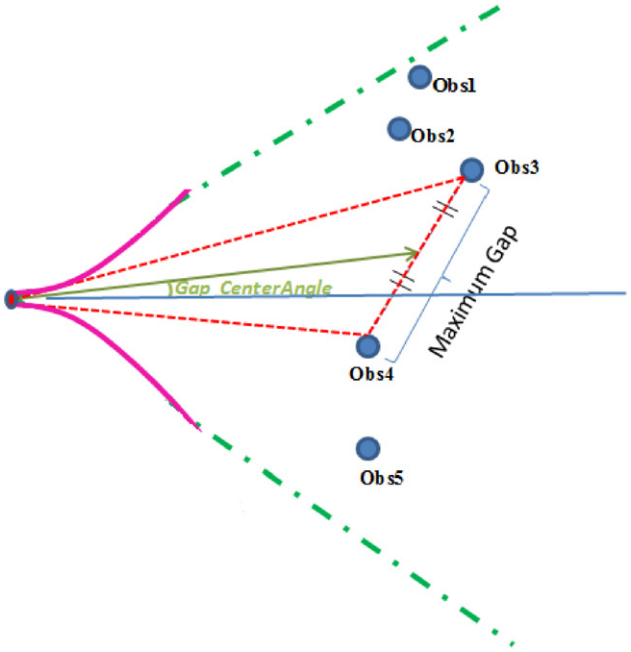


Fig. 7. Gap center angle representation.

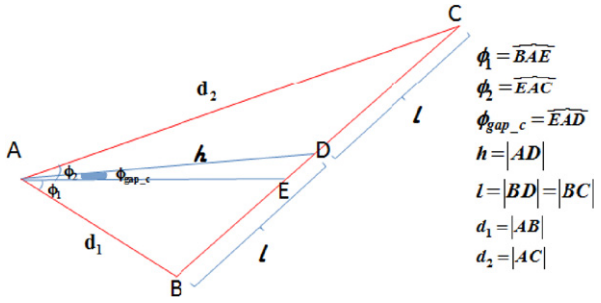


Fig. 8. Gap center angle parameterization.

triangle. The triangle and the gap center angle are shown in detail in Fig. 8.

The aim is to find the ϕ_{gap_c} in terms of the measurable d_1, d_2, ϕ_1, ϕ_2 parameters which are shown in Fig. 8. (d_1, d_2 are distances to obstacles from the maximum gap. ϕ_1, ϕ_2 are angles of obstacles from the maximum gap.)

Firstly, the Cosine Rule is applied to the ABC triangle:

$$(2l)^2 = d_1^2 + d_2^2 - 2d_1 d_2 \cos(\phi_1 + \phi_2)$$
$$l^2 = \frac{d_1^2 + d_2^2 - 2d_1 d_2 \cos(\phi_1 + \phi_2)}{4}. \quad (4)$$

After that, the Apollonius theorem is applied to the ABC triangle.

$$d_1^2 + d_2^2 = 2l^2 + 2h^2. \quad (5)$$

Replacing l^2 with Eq. (4);

$$h^2 = \frac{d_1^2 + d_2^2 + 2d_1 d_2 \cos(\phi_1 + \phi_2)}{4}. \quad (6)$$

Finally, the Cosine Rule is again applied to the ABD triangle

$$l^2 = d_1^2 + h^2 - 2d_1 h \cos(\phi_1 + \phi_{gap\_c}). \quad (7)$$

Replacing l^2 and h^2 with Eqs. (4) and (6);

$$\frac{d_1^2 + d_2^2 - 2d_1 d_2 \cos(\phi_1 + \phi_2)}{4} = d_1^2 + \frac{d_1^2 + d_2^2 + 2d_1 d_2 \cos(\phi_1 + \phi_2)}{4} - 2d_1 \frac{\sqrt{d_1^2 + d_2^2 + 2d_1 d_2 \cos(\phi_1 + \phi_2)}}{2} \cos(\phi_1 + \phi_{gap\_c}) \quad (8)$$

$$\phi_{gap\_c} = \arccos\left(\frac{d_1 + d_2 \cos(\phi_1 + \phi_2)}{\sqrt{d_1^2 + d_2^2 + 2d_1 d_2 \cos(\phi_1 + \phi_2)}}\right) - \phi_1.$$

The gap center angle (ϕ_{gap_c}) is found in terms of the measurable d_1, d_2, ϕ_1, ϕ_2 parameters in Eq. (8).

The gap center angle could be found by calculating the average of ϕ_1 and ϕ_2 angles, ignoring the d_1 and d_2 distance values:

$$\phi_{gap\_c\_basic} = \frac{\phi_1 + \phi_2}{2}. \quad (9)$$

This variant of FGM, in which $\phi_{gap\_c\_basic}$ is used instead of ϕ_{gap_c} is called FGM-basic for comparisons in the simulations section. However, FGM-basic causes unsafe trajectories in some cases. The simulation results of such an obstacle configuration are illustrated in Fig. 9.

Fig. 9a shows the result of ϕ_{gap_c} and Fig. 9b shows the result of $\phi_{gap\_c\_basic}$. There are two obstacles at the same coordinates for both of the simulations. It is shown that, ϕ_{gap_c} has resulted in a safer trajectory. This is due to the calculations for finding the angle of the median vector from the robot to the line between the obstacles, as explained previously.

Here, with the term "safe", maintaining a safe distance from the obstacle is meant. In the simulations section, a metric is defined for measuring this safety and for measuring the performances of different methods.

3.3. Calculation of the final heading angle

The last step of the algorithm is the calculation of the final heading angle. This final angle reference will be the reference heading vector of the robot for avoiding obstacles and arriving at the goal point. A sample configuration including obstacles and a goal point is illustrated in Fig. 10.

The final angle is the combination of the gap center angle and goal angle. The combination structure depends on the minimum distance to obstacles around and weight coefficients. If the obstacles are near the robot, it should consider safety first; i.e. the gap center angle, rather than the goal angle and vice versa. Fusing function below depicts the final angle calculation:

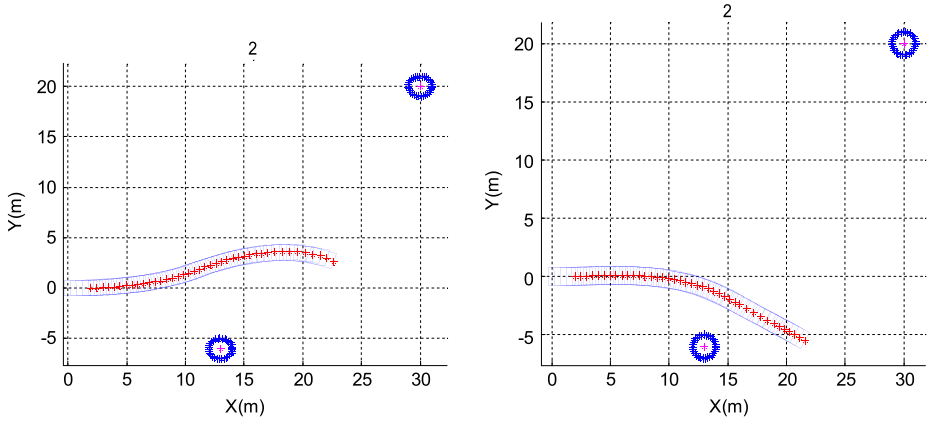
$$\phi_{final} = \frac{\frac{\alpha}{d_{\min}}\phi_{gap\_c} + \beta\phi_{goal}}{\frac{\alpha}{d_{\min}} + \beta} \quad \text{where } d_{\min} = \min_{i=1:n}(d_n). \quad (10)$$

(ϕ_{gap_c}: Gap Center Angle. ϕ_{goal}: Goal Angle. α: Weight Coefficient for gap. β: Weight Coefficient for goal. n: Number of obstacles. d_n: Distance to nth obstacle. d_{min}: Minimum of d_n distance values.)

For simplicity, β is taken as "1" and the α coefficient is used as a weight factor between the gap center angle and the goal angle. Eq. (10) can be rewritten as illustrated below:

$$\phi_{final} = \frac{\frac{\alpha}{d_{\min}}\phi_{gap\_c} + \phi_{goal}}{\frac{\alpha}{d_{\min}} + 1} \quad \text{where } d_{\min} = \min_{i=1:n}(d_n). \quad (11)$$

This weighted average function is not only dependent on tuning parameter but also the minimum distance to the obstacle.



(a) Result of ϕ_{gap_c} (FGM). (b) Result of $\phi_{gap\_c\_basic}$ (FGM-basic).

Fig. 9. Comparison of ϕ_{gap_c} and $\phi_{gap\_c\_basic}$ gap angle calculation.

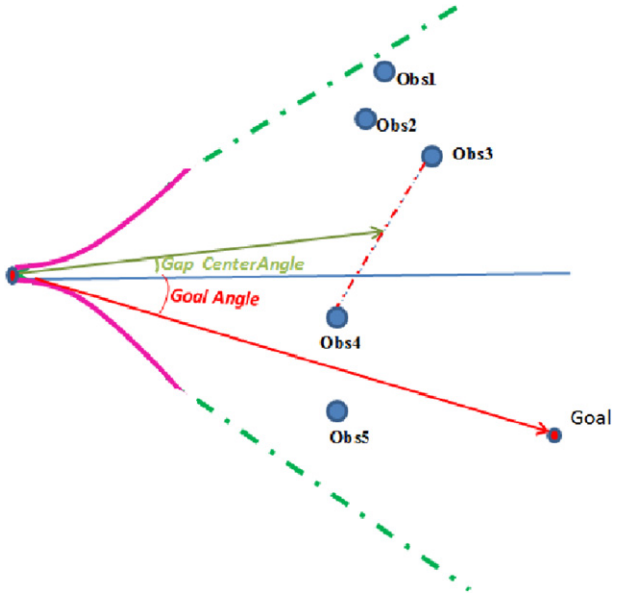


Fig. 10. Obstacles with goal point.

Theoretically, the final angle approaches the gap angle when the distance to the obstacle approaches zero. The fusing function is designed specifically to reflect this property. This can be seen from Eq. (12) by using the l'Hospital's rule.

$$\lim_{d\,\min\to 0}\phi_{final} \cong \frac{\infty}{\infty} \Rightarrow \lim_{d\,\min\to 0}\phi_{final} \cong \frac{\frac{\partial\left(\frac{\alpha}{d_{\min}}\phi_{gap\_c}+\beta\phi_{goal}\right)}{\partial d_{\min}}}{\frac{\partial\left(\frac{\alpha}{d_{\min}}+\beta\right)}{\partial d_{\min}}} = \frac{\frac{-\alpha\phi_{gap\_c}}{d_{\min}^2}}{\frac{-\alpha}{d_{\min}^2}} = \phi_{gap\_c}. \quad (12)$$

Fig. 11 shows how ϕ_{final} changes with respect to alpha and minimum obstacle distance. For this figure, $\phi_{gap\_c} = pi/4$ and $\phi_{goal} = -pi/4$ which means the gap center and the goal are in different directions.

According to Fig. 11, ϕ_{final} converges to ϕ_{gap_c} (pi/4) for decreasing values of $d_{\min}$ (an obstacle is approaching the robot) and ϕ_{final} converges to ϕ_{goal} (-pi/4) for increasing values of $d_{\min}$ (an obstacle is moving far away). The α value determines how much the robot is goal oriented or gap oriented. For $\alpha = 0$, ϕ_{final} is equal to ϕ_{goal} (-pi/4) and increasing values of alpha brings ϕ_{final} closer to ϕ_{gap_c} (pi/4) as is illustrated in Fig. 11.

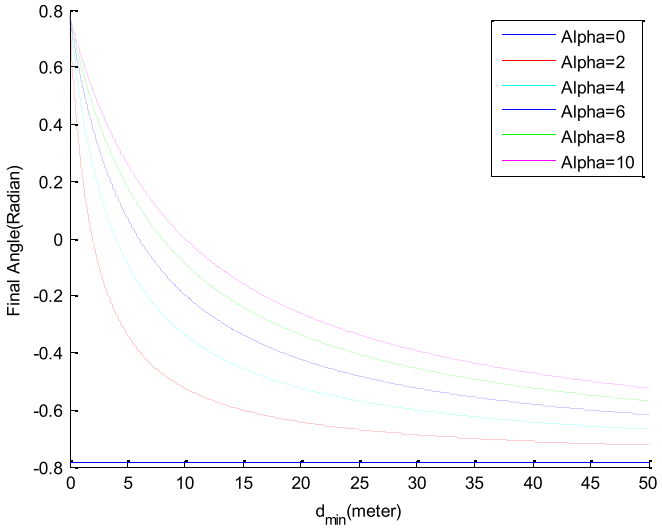


Fig. 11. Final angle with respect to minimum distance and α coefficient ($\phi_{gap\_c} = pi/4$ radian and $\phi_{goal} = -pi/4$ radian).

When compared to other obstacle avoidance methods, FGM has several advantages. FGM has only one tuning parameter, which is the alpha coefficient in Eq. (11). So tuning the algorithm is very easy. FGM does not have the local minimum problem, as in APF and VFF. Different from previous avoidance algorithms, this new approach can consider the nonholonomic constraints of the robot and only feasible trajectories are generated. Another advantage is that the field of view of the robot is taken into account and the robot is not forced to move toward unmeasured directions. Finally, FGM steers the robot to move towards the center of the maximum gap as much as possible. This results in safer trajectories compared with previous approaches, as will also be shown in simulations.

4. Simulations

The simulation and experimental platform for this novel algorithm is a nonholonomic ground vehicle. Before the real tests of the algorithm, a simulation model is prepared using the Matlab/Simulink® environment. The kinematic model parameters of the vehicle are shown in Fig. 12.

For path planning and obstacle avoidance of a low velocity and mass car-like robot, the kinematical model usually suffices [24]. Vehicle kinematic equations are illustrated below, using the

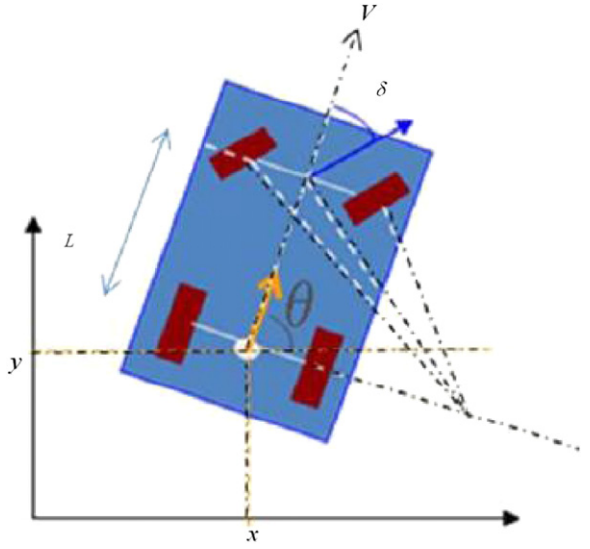


Fig. 12. Vehicle kinematics.

parameters in Fig. 12.

$$\frac{dx}{dt} = V\cos(\theta) \tag{13}$$

$$\frac{dy}{dt} = V\sin(\theta) \tag{14}$$

$$\frac{d\theta}{dt} = \frac{V}{L}\tan(\delta). \tag{15}$$

Every obstacle avoidance algorithm has tuning parameters for performance improvements. Since a fair benchmarking is a very challenging problem for different obstacle avoidance methods [25], we have done our comparison with our previously tuned APF algorithm [26], as APF is one of the most widely used obstacle avoidance algorithms in robotics area.

We perform a series of Monte Carlo simulations over random environments to compare the performance of the FGM, FGM-basic, APF and A^* methods. FGM-basic is a variant of FGM, in which $\phi_{gap\_c\_basic}$ is used instead of ϕ_{gap_c} as explained in Section 3.2.

All the algorithms are coded in S-Function using C programming language. A stereo LIDAR (Light Detection and Ranging) sensor is mounted in front of the vehicle with a total 150° field of view and 10 m range as the experimental platform used in simulations. Circular obstacles with random radius and coordinate values are used in simulations. Even though the vehicle has a rectangular shape, an equivalent minimum radius circular robot is used for the calculations. This is done by taking the diagonal of the rectangle as the diameter of the circle. Tuning parameter alpha is selected as 20. Simulation results of FGM for two different obstacle configurations are shown in Fig. 13. Goal coordinates are [70–70] for Configuration A and [80–0] for Configuration B.

FGM does not suffer from the local minimum problems except the dead-end scenarios. Fig. 14 shows the trajectories of APF and FGM in two different local minimum scenarios.

As can be seen from the simulation results, FGM can reach the goal point while avoiding obstacles but in APF method, robot gets stuck because of the local minimum where all vectors from the obstacles and goal point zero each other. FGM selects the first calculated gap value if there are equal maximum gaps as mentioned in 3.1. This provides FGM to move if at least one gap exists. However, as a result of its local characteristic, a dead-end scenario of U-shaped obstacles is a problem for FGM as it is for APF. Detecting the dead-end scenario and avoiding it needs an upper level intelligence for such a kind of local planners and can be solved with several ad-hoc approaches which have been studied in the literature extensively. In [17] virtual obstacles and in [18] virtual goal points are added to the local map for trap conditions. Beside these, [15] uses multiple goal points and [19] uses the wall following technique with the same aim. One of these solutions can be used directly in FGM for dead-end scenarios.

The following metric for obstacle avoidance safety is defined for comparison. This collision avoidance metric is based on the inverse of the distance-to-obstacle function. The same metric was also used in [27].

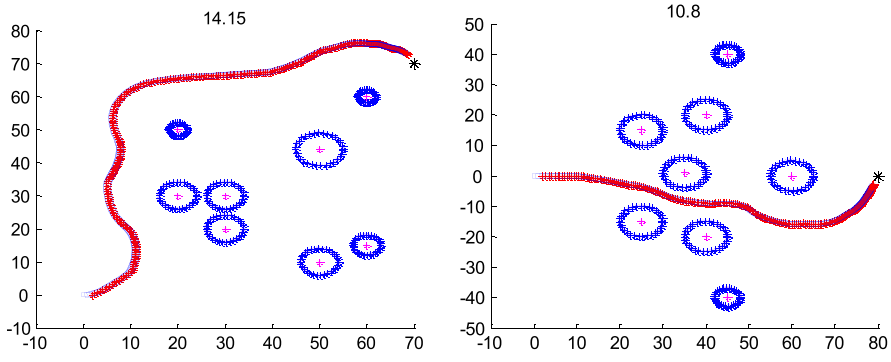
$$f(t) = \begin{cases} \dfrac{1}{d_{\min}} - \dfrac{1}{d_0} & \text{for } d_{\min} < d_0 \\ 0 & \text{for } d_{\min} \geq d_0 \end{cases} \tag{16}$$

where $d_{\min}$ is the closest distance between the vehicle and obstacles and the given scalar d_0 denotes the distance to an obstacle that poses no danger for collision during execution.

The pth norm of a function is defined as

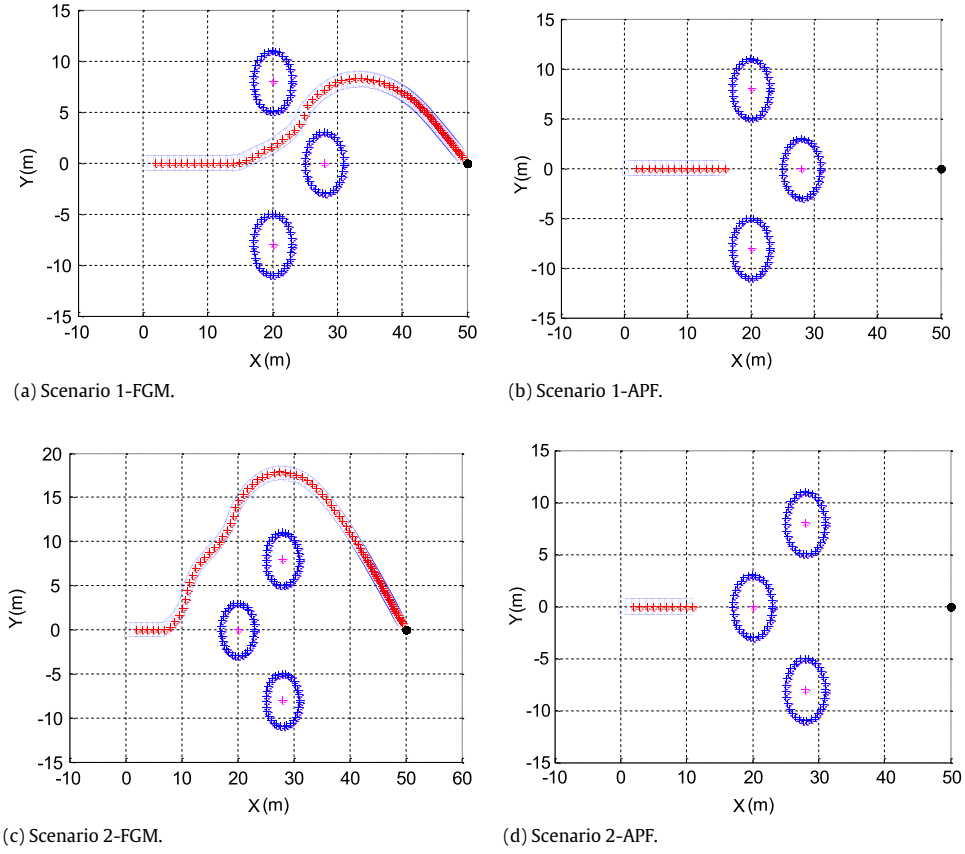
$$\|f\|_p = \left(\int |f(t)|^p \, dt\right)^{1/p}. \tag{17}$$

In this paper, the first norm ($p = 1$) of the collision avoidance metric is calculated and taken as a performance criterion. This performance criterion measures the safety of the trajectory or in another words how far away from obstacles the trajectory is. The aim of FGM is obtaining the maximum distance to obstacles as far as possible reactively. This brings an additional path length to the shortest path. Fig. 15 illustrates the trajectories of FGM, APF and A^* shortest path algorithm for three different obstacle scenarios. A^* uses a best-first search and finds the shortest path between starting and goal points [28]. A^* is a global planning algorithm which means all the information about the map is given before



(a) Configuration A. (b) Configuration B.

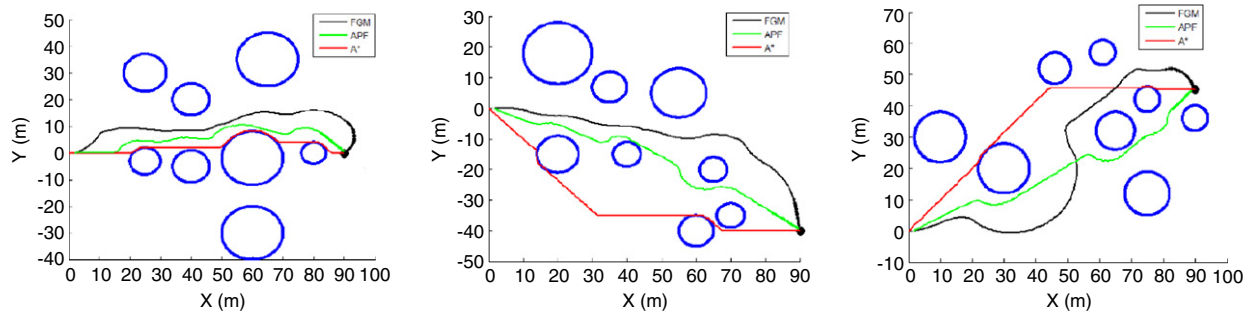
Fig. 13. Simulation results of FGM for different obstacle configurations.



(a) Scenario 1-FGM. (b) Scenario 1-APF.

(c) Scenario 2-FGM. (d) Scenario 2-APF.

Fig. 14. Trajectories of APF and FGM in local minimum scenarios.



(a) Scenario-A. (b) Scenario-B. (c) Scenario-C.

Fig. 15. Comparison of FGM, APF and A^* shortest path trajectories.

it starts, in contrast to reactive algorithms like FGM and APF. In simulations, the A^* algorithm uses 0.5 m grid size and does not consider any measurement or kinematic constraint.

Forty Monte Carlo simulations are performed for FGM, FGM-basic, APF and A^* search methods and the results are illustrated in Fig. 16. The collision avoidance norm value of the A^* algorithm is not added in Fig. 16 because the collision avoidance norm approaches infinity as the distance to obstacle value approaches zero. This can be seen in Eq. (16).

According to Fig. 16, the collision avoidance norm of FGM is lower than both the APF and FGM-basic methods which means the trajectories are safer. Despite this, total traveled distance values are almost the same as illustrated in Fig. 16. Average values of collision norm and distance traveled for this 40 Monte Carlo simulations are illustrated in Table 1.

Table 1
Average of simulation results ($d_0 = 25$ m).

	Average collision norm	Average distance traveled (m)
FGM	0.186	65.82
FGM-basic	0.243	64.94
APF	0.31	63.57
A^*	$\approx\infty$	58.81

According to Table 1, FGM is 23% safer than the FGM-basic and 40% safer than the APF in terms of the norm of the defined metric while the total distance traveled values are almost the same.

5. Experimental platform

After designing the algorithm and successful simulation results, the Follow the Gap method is implemented on the real

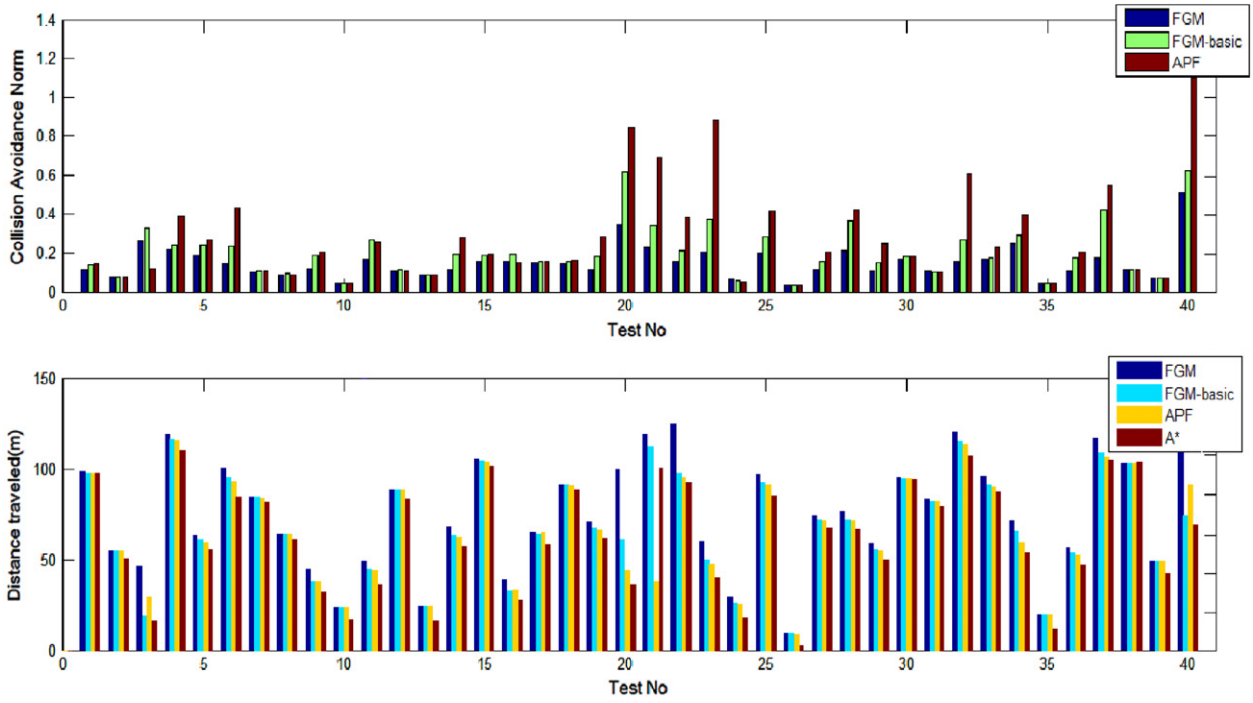
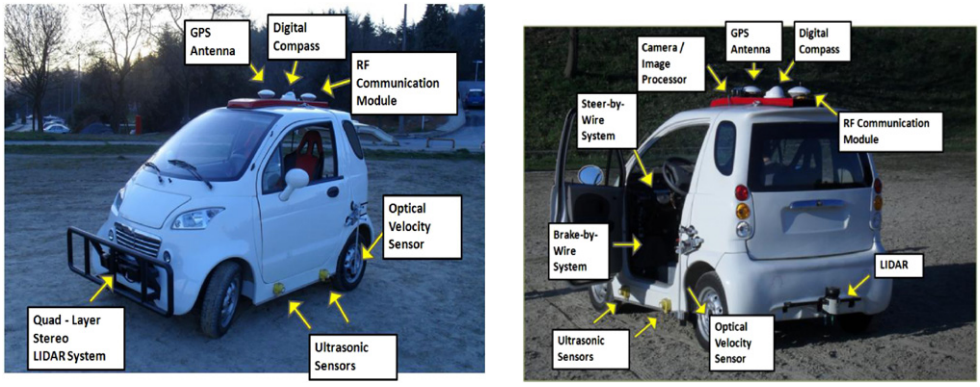


Fig. 16. Collision avoidance norm and traveled distance values for Follow the Gap, Follow the Gap—basic and artificial potential fields methods.



(a) Front view. (b) Rear view.

Fig. 17. Experimental set-up: fully autonomous ground vehicle with its sensors.

autonomous vehicle, 'Otonobil'. Otonobil is a fully autonomous ground vehicle which includes sensors for localization and mapping, real time computers and drive by wire capability. Fig. 17 shows the sensors mounted on Otonobil.

Otonobil has three main process systems which work in parallel. Each computer system works with different aims and communicates the others. A list of these computational units with their duties is shown in Table 2.

All the computer and sensor systems and their communication protocols are illustrated in Fig. 18. Since sensors have different communication protocols, in-vehicle communication is performed with CAN, Ethernet, RS232 and analog communication. More information about the hardware and software structure of the experimental platform can be found in [26,29,30].

6. Experimental results

The obstacle avoidance algorithm based on the Follow the Gap Method is coded using C programming language. The real-time code is run in Microautobox hardware, shown in Table 2 and Fig. 18. The only tuning parameter, alpha, is selected as 20 in experimental tests as in the simulations. The field of view of the LIDAR is 150° and its measurement range is restricted to 10 m. The first test configuration is composed of seven static obstacles with a goal point. The test field and the obstacle configuration are illustrated in Fig. 19.

Results of this algorithm are shown in Fig. 20. Fig. 20a shows the trajectory of the vehicle with black dots. Blue dots are obstacle measurement results from the LIDAR sensor. The goal point coordinates are [−35, 50] which is shown with a red dot and the starting point coordinates are [−3, −3]. As can be seen in Fig. 20a, static perceived obstacles and even the goal point are a cluster of scattered points. The reason for the scatter is that the LIDAR and GPS receivers have some measurement noise and these two error sources affect the calculations. Fig. 20b shows the steering wheel reference angle and steering wheel real angle values during its journey.

The results of the second test are illustrated in Fig. 21. This time the obstacle configuration is different with the starting

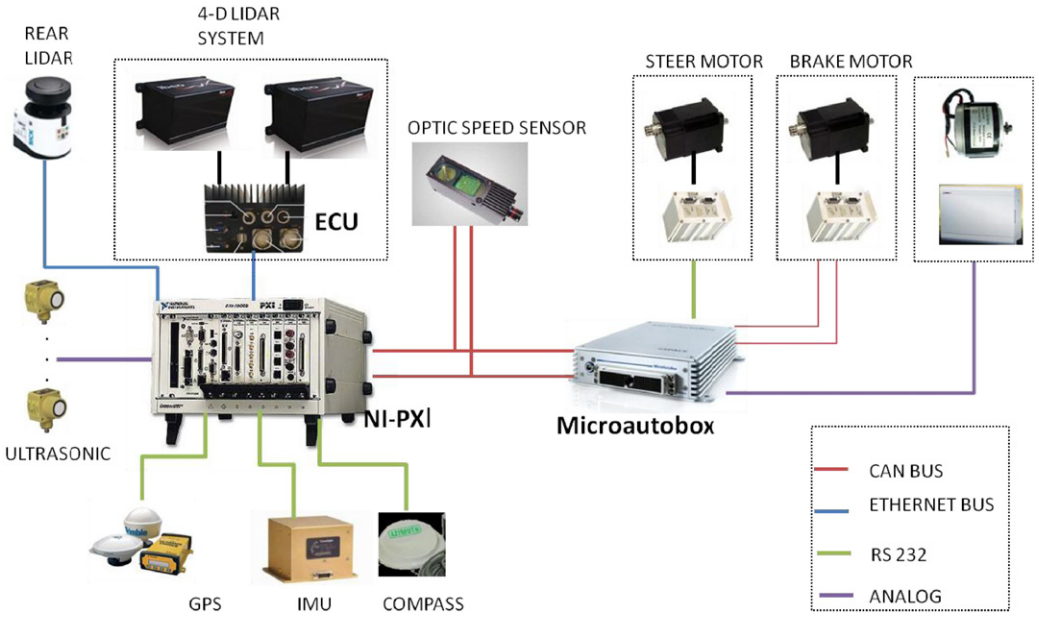


Fig. 18. Sensor and computer system and their communication in Otonobil.

Table 2
Computational power of otonobil.

Computer	Description	Duty
NI PXI-8110RT processor PXI-7954R FPGA module on PXI1000B chassis	It has 2.26 GHz quad-core processor and a powerful FPGA module with several I/O cards.	Localization, mapping and global path planning.
DSpace Microautobox 1401/1501/1507	It has 800 MHz processor and several I/O interfaces	**Obstacle avoidance**, path tracking, low level control (throttle, steering, brake), wireless communication.
IBEO ECU	It can give both object and raw data. It has tracking algorithms inside and can classify the objects around it	LIDAR processing, mapping.

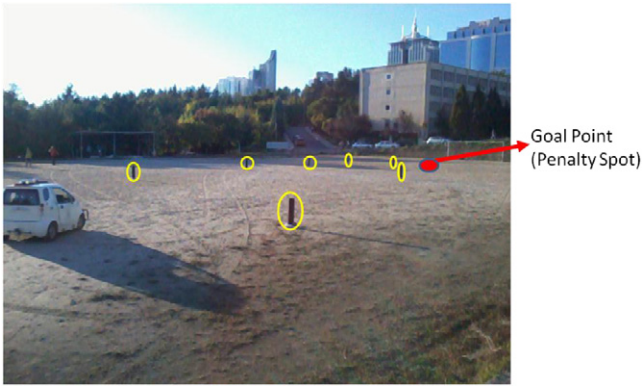


Fig. 19. Test field and first obstacle configuration.

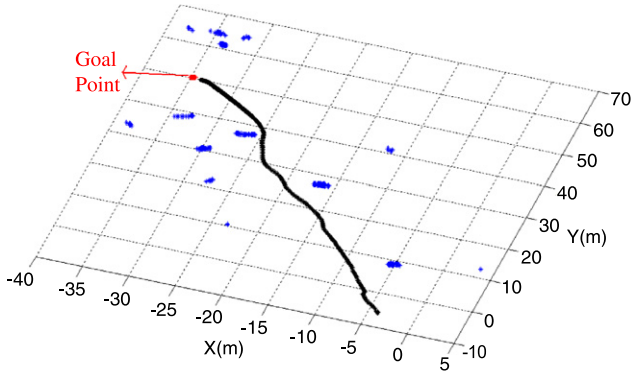
point coordinates [−57, 58] and goal point coordinates [−20, 5]. According to Fig. 21, the vehicle heads itself towards the goal point until it measures an obstacle which means the final angle is equal to the goal angle. When it detects an obstacle, the appropriate heading vector is calculated by the Follow the Gap Method and the vehicle avoids obstacles and arrives at the goal point.

Finally, the Follow the Gap Method is tested in a dynamic obstacle scenario. Fig. 22 shows the trajectory of the system in a dynamic obstacle case. In Fig. 22a, for better analysis, the dynamic obstacle trajectory is illustrated in a 3D plot where the third axis is for time. In this 3D illustration, collision means the intersection not only for X–Y axis but also for X–Y and time axes. According to Fig. 22, there is no collision and the vehicle avoids dynamic obstacles successfully. Dynamic obstacles are moving obstacles with unknown motion patterns, which are circled in Fig. 22a. These moving obstacles are people who are running in front of the vehicle in this scenario. A video demonstrating the vehicle performance within this scenario can be found at http://www.youtube.com/watch?v=TohW9xokbaM.

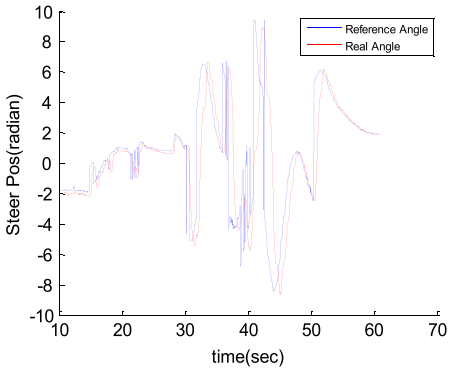
7. Conclusion

A new approach called the "Follow the Gap Method" is presented in this paper. The proposed algorithm is proven to be successful in driving the nonholonomic autonomous ground vehicle from an arbitrary initial location to a goal location while avoiding collision with static and dynamic obstacles. The major contributions can be listed as follows.

- Gap center angle formulation enables the robot to head to the center of the maximum gap around which results in safe trajectories. Fig. 16 shows this type of comparison between the Follow the Gap Method, Follow the Gap Method—basic and Artificial Potential Fields Method using Monte Carlo simulations.
- There is no local minimum problem. The APF method and VFF method have this serious problem as explained in the introduction.
- Nonholonomic constraints of the robot are taken into account and feasible trajectories are generated while the other methods do not have this property.
- The field of view of the robot is taken into account and the robot is not forced to move towards unperceived directions.
- Follow the Gap algorithm is easy to tune with only one tuning parameter, "alpha".

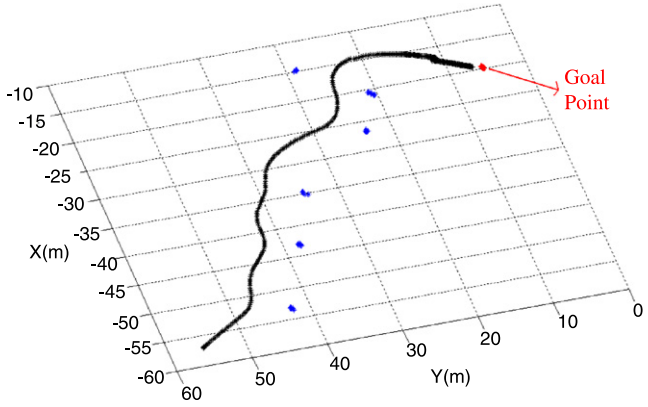


(a) Vehicle trajectory with obstacle measurement.

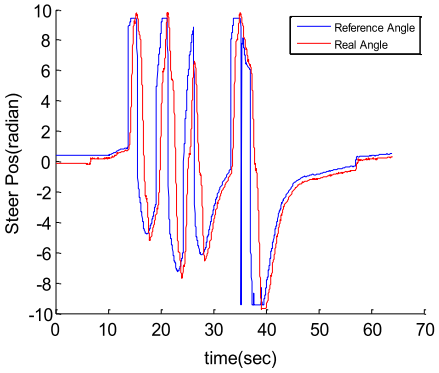


(b) Steering wheel reference and real angle.

Fig. 20. Experimental results of test1.

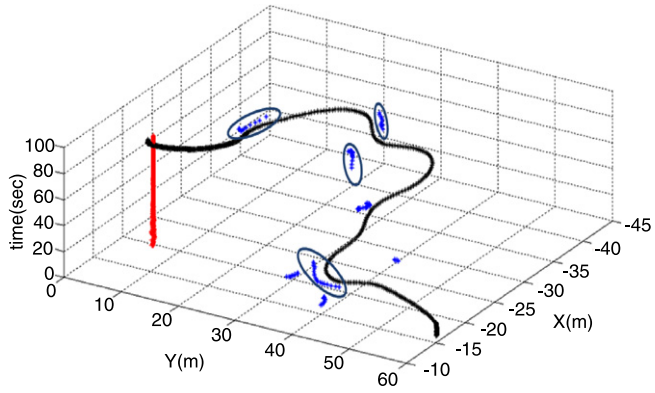


(a) Vehicle trajectory with obstacle measurement.

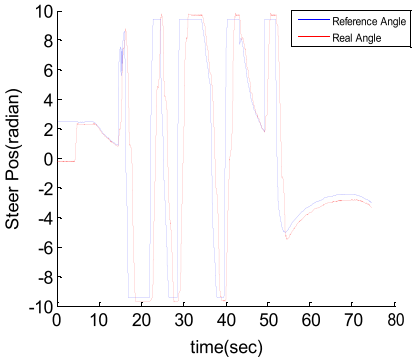


(b) Steering wheel reference and real angle.

Fig. 21. Experimental results of test2.



(a) Vehicle trajectory with obstacle measurement.



(b) Steering wheel reference and real angle.

Fig. 22. Experimental results of test3 (dynamic obstacle scenario).

In spite of successful results in both simulations and experimental tests, this new algorithm can be further extended to the dynamic model of the robot. This paper illustrates the concept of the algorithm and it is seen that the kinematical vehicle model is sufficient for the proof of concept. However, at high speed values, a dynamic model and additional constraints from the dynamic vehicle model should be taken into account. Even though the algorithm can overcome the dynamic obstacle scenarios because of its reactive nature, direction and speed values of the obstacles can be added in gap angle calculations for future studies. Finally, measurement noise and uncertainties of the system can be taken into account in order to obtain a more extensive and robust algorithm.

Appendix A

Cosine Theorem (Law of Cosines):

Fig. 23 illustrates the notation for the Cosine Theorem. According to Fig. 23, the law of cosines says:

$$\begin{aligned} a^2 &= b^2 + c^2 - 2bc\cos(\phi_1) \\ b^2 &= a^2 + c^2 - 2ac\cos(\phi_2) \\ c^2 &= a^2 + b^2 - 2ab\cos(\phi_3). \end{aligned} \tag{18}$$

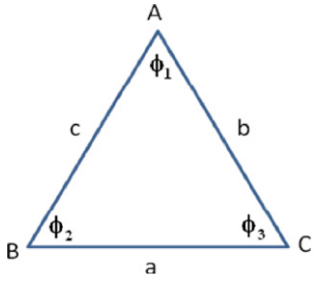


Fig. 23. Cosine theorem notation.

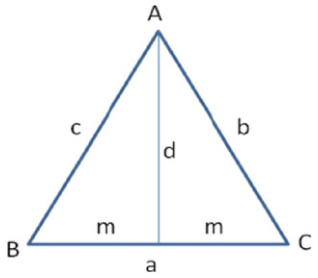


Fig. 24. Apollonius theorem notation.

Apollonius Theorem:

Fig. 24 illustrates the notation for the Apollonius Theorem. According to Fig. 24, the Apollonius Theorem says:

$$b^2 + c^2 = 2m^2 + 2d^2. \tag{19}$$

Appendix B. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2012.05.021.

References

[1] K. Fujimura, Motion Planning in Dynamic Environments, Springer-Verlag, Tokyo, Japan, 1991.
[2] A. Chakravarthy, D. Ghose, Obstacle avoidance in a dynamic environment: a collision cone approach, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 28 (5) (1998) 562–574.
[3] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (5) (1996) 566–580.
[4] S.M. Lavalle, Rapidly-exploring random trees: a new tool for path planning, Tech. Rep. (1998).
[5] S. Chakravorty, S. Kumar, Generalized sampling-based motion planners, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (3) (2011) 855–866.
[6] T. Lozano-Pecz, An algorithm for planning collision free paths among polyhedral obstacles, Communication of the ACM 22 (10) (1979) 560–570.
[7] F. Aurenhammer, Voronoi diagrams–a survey of a fundamental geometric data structure, ACM Computing Surveys 23 (3) (1991) 345–405.
[8] R. Siegwart, I.R. Nourbakhsh, Introduction to Autonomous Mobile Robots, MIT Press, 2004.
[9] J.A. Reeds, R.A. Shepp, Optimal paths for a car that goes both forward and backward, Pacific Journal of Mathematics 145 (2) (1990) 367–393.
[10] L. Tzu-Chen, L. Jing-Sin, H. Gau-Tin, C. Yau-Zen, Practical and flexible path planning for car-like mobile robot using naximal-curvature cubic spiral, Robotics and Autonomous Systems 52 (2005) 312–335.
[11] V.J. Lumelsky, A.A. Stepanov, Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape, Algorithmica 2 (1–4) (1987) 403–430.
[12] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, International Journal of Robotic Research 51 (1986) 90–98.
[13] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, IEEE Conference on Robotics and Automation (1991) 1398–1404.
[14] J.-O. Kim, P.K. Khosla, Real-time obstacle avoidance using harmonic potential functions, IEEE Transactions on Robotics and Automation 83 (1992) 338–349.
[15] M.A.P. Castaneda, J. Savage, A. Hernandez, F.A. Coso, Local autonomous robot navigation using potential fields, InTech (2008).
[16] S. Shimoda, Y. Kuroda, K. Iagnemma, Potential field navigation of high speed unmanned ground vehicles on uneven terrain, Proceedings of the 2005 IEEE International Conference on Robotics and Automation (2005) 2828–2833.
[17] M.H. Ang, H. Krishnan, Virtual obstacle concept for local- minimum-recovery in potential-field based navigation, IEEE International Conference on Robotics and Automation (ICRA) (2000) 983–988.
[18] Z. Xi-yong, Z. Jing, Virtual local target method for avoiding local minimum in potential field based robot navigation, Journal of Zhejiang University Science 43 (2003) 264–269.
[19] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Transactions on Systems, Man and Cybernetics 19 (5) (1989) 1179–1187.
[20] J. Borenstein, Y. Koren, The vector field histogram - fast obstacle-avoidance for mobile robots, IEEE Journal of Robotics and Automation 7 (3) (1991) 278–288.
[21] F. Xu, H.V. Brussel, M. Nuttin, R. Moreas, Concepts for dynamicobstacleavoidance and their extended application in underground navigation, 42 (1) (2003) 1–15.
[22] A.M. Bloch, J. Baillieul, P. Crouch, J. Marsden, Nonholonomic Mechanics and Control, Springer, 2003.
[23] I. Kolmanovsky, N.H. McClamroch, Developments in nonholonomic control problems, IEEE Control Systems Magazine 5 (10) (1995) 20–36.
[24] N. Ghita, M. Kloetzer, Trajectory planning for a car-like robot by environment abstraction, Robotics and Autonomous Systems (2012) http://dx.doi.org/10.1016/j.robot.2011.12.004.
[25] W. Nowak, A. Zakharov, S. Blumenthal, E. Prassler, Benchmarks for mobile manipulation and robust obstacle avoidance and navigation, BRICS Deliverable D31 (2010).
[26] V. Sezer, C. Dikilitas, Z. Ercan, H. Heceoglu, P. Boyraz, M. Gökaşan, Hardware and Software Structure of Unmanned Ground Vehicle Otonobil, 5th Biennial Workshop on DSP for In-Vehicle Systems, 2011.
[27] S. Nair, M. Kobilarov, Collision avoidance norms in trajectory planning, American Control Conference (ACC) (2011) 4667–4672.
[28] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.
[29] V. Sezer, C. Dikilitas, Z. Ercan, H. Heceoglu, A. Oner, A. Apak, M. Gokasan, A. Mugan, Conversion of a conventional electric automobile into an unmanned ground vehicle (UGV), IEEE International Conference on Mechatronics (2011) 564–569.
[30] Z. Ercan, V. Sezer, H. Heceoglu, C. Dikilitas, M. Gokasan, A. Mugan, S. Bogosyan, Multi-sensor data fusion of DCM based orientation estimation for land vehicles, IEEE International Conference on Mechatronics (2011) 672–677.

Volkan Sezer received his B.Sc. degree in electronics and telecommunication engineering from Yildiz Technical University, Istanbul, Turkey in 2005, and the M.Sc. degree in mechatronics engineering from Istanbul Technical University, Istanbul, Turkey in 2008. He is currently a Ph.D. candidate in control and automation engineering at Istanbul Technical University and leader of the Autonomous Ground Vehicle Project. He has been working in the Mechatronics Education and Research Center since 2009. His fields of interest are autonomous ground robots, trajectory planning, obstacle/collision avoidance, semi autonomous vehicles, active safety in road vehicles, control of hybrid electric vehicles, energy efficiency, and real-time programming.

Metin Gokasan received his B.Sc., M.Sc. and Ph.D. from Istanbul Technical University, Turkey in electrical and control engineering in 1980, 1982 and 1990, respectively. He is currently Professor at Electrical and Electronics Engineering Faculty and acting Department Chair of Control Engineering in Istanbul Technical University. Between 2003 and 2006, he conducted his research at the University of Alaska Fairbanks as a visiting scholar. His research interests are control of electrical machinery, power electronics and electrical drives, control of hybrid vehicles, autonomous robots and mechatronics systems. He has authored two books and over 80 journal and conference publications.