A Bug-Based Local Path Planning Method for Static and Dynamic Environments*

Suat Karakaya, Hasan Ocak and Gürkan Küçükyıldız
Faculty of Engineering, Department of Mechatronics Engineering, Kocaeli University, Turkey

Abstract

In this study, a local path planning method was proposed for both static and dynamic environments. In obstacle-free cases, the mobile robot was forced to basic motion-to-goal movement. In case where direct movement towards the global target is not possible, the algorithm searches for possible gaps which satisfy certain clearance criteria. The gaps were detected by taking the gradient of one dimensional distance vector acquired from the SICK LMS100 Light Detection and Ranging (LIDAR) sensor. The detected gaps were filtered by various methods which finally led to the optimal gap. Points on the line passing through the optimal gap were evaluated through a cost function and the point having the minimum cost was assumed to be the current local target. The points which were close to the two opposite corners of the gap less than a certain threshold were discarded to avoid collision. The threshold was determined based on the robot size and the kinematic model. Proportional and integral (PI) speed controller for left and right steered wheels was adapted to the proposed method. A graphical user interface (GUI) was developed to visualize the outputs of the method. On the GUI, offline LMS100 vectors and location data were visualized considering differential drive kinematic constraints for the mobile robot. The algorithm was developed at MATLAB environment.

Key words: Mobile Robot, Path Planning, Bug Algorithm

1. Introduction

Path planning is one of the most attractive topics in autonomous mobile robotics. Path planning can be defined as generating local target points from a start position to a goal position. A continuous path can be generated by combining all the local target points. Based on available knowledge about the environment, mobile robot path planning can be classified into two categories as local and global path planning. In global path planning, mobile robot utilizes information about the environment. Therefore, configuration of obstacles, gaps and the other environmental factors should be initially available in mobile robot's memory. Using the information about robot's workspace, various methods can be applied to optimal path planning problem. Artificial potential fields were used for global path planning in [1]. Genetic algorithms can also be applied in global path planning applications [2]. In addition, hybrid approaches can be developed for global path planning by combining different methods [3]. In contrast to global path planning, there is only limited information about the environment in local approaches [4]. The

*This study was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under the grant TUBITAK-113E777

Corresponding author: Address: Faculty of Engineering, Department of Mechatronics Engineering, Kocaeli University, Turkey

robot makes use of only start and goal positions, distance measurement sensors and imperfect information about the workspace. The robot needs to sense its environment before planning a move towards to goal and to avoid possible collisions. Potential fields approach is one of the widely-used local path planning methods [4]. In this method, the goal position is assumed to be an attractive electromagnetic particle while obstacles are all repulsive. The mobile robot is considered as a particle moving under the potential field. Bug algorithms are also well known local path planning techniques. Several sub-types of the method exist in the literature such as dist-bug, vis-bug, tangent-bug, bug-0, bug-1, bug-2, etc. All these methods require sensor readings within a wide angle of view. While the robot is moving towards the goal, the algorithm continuously checks the distance between the robot and the goal point. [5-6]. Tangent bug algorithm gives satisfying local targets in most dynamic environments by decreasing wall tracking requirement [6-9]. The tangent bug algorithm calculates heuristic and deterministic distances between the robot and the global target points without considering size of the the mobile robot. In this study, a bug based local path planning algorithm was developed considering not only the theory but also real time conditions such as collision avoidance, mobile robot mechanical constraints and the specs of a real Sick LMS100 LIDAR sensor. A bare tangent bug and the developed method were compared based on collision avoidance.

2. LIDAR Measurement Data

In this study, Sick LMS100 sensor was used as LIDAR. It has a 270^o angle of scanning view and a scanning frequency of either 25 Hz or 50 Hz. Angular resolutions of the measurements can be set to 0.5^o or 1^o. Depending on the angular scanning resolution, length of the LIDAR measurement data can be in size of 1x271 or 1x541. Each element of the measurement data (or array) corresponds to the distance (in mm) of related index.

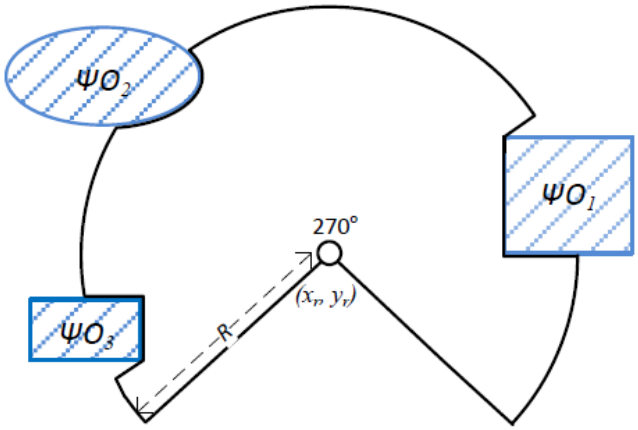

Figure 1. LIDAR data distance function

Let R be maximum measurement range of the LIDAR, M be the number of the obstacles, (x_r, y_r) be the position of the mobile robot and $\psi O_{j=1:M}$ be the obstacles as illustrated in Figure 1. The mathematical model of the LIDAR data is given in Equation 1, where $\rho(x,\theta)$ is the LIDAR measurement data at corresponding angle (θ) and d is the Euclidian distance function.

$$\rho(x,\theta) = \min_{\lambda \in [0,R)} d(x, x + \lambda[\sin\theta, \cos\theta]) \text{, where } x + \lambda[\sin\theta, \cos\theta] \in \bigcup_j \psi O_j \tag{1}$$

$$\rho(x,\theta) = \begin{cases} \rho(x,\theta), & if \quad \rho(x,\theta) < R \\ R, & else. \end{cases}$$

3. Gap Detection and Optimization

LIDAR distance data are processed to obtain existing gaps around the mobile robot by taking the gradient of the distance array. Discontinuity points are searched in the distance array by considering inconstant increment/decrement or changes bigger than a pre-determined threshold. Points of discontinuity are assumed to be the obstacle corners.

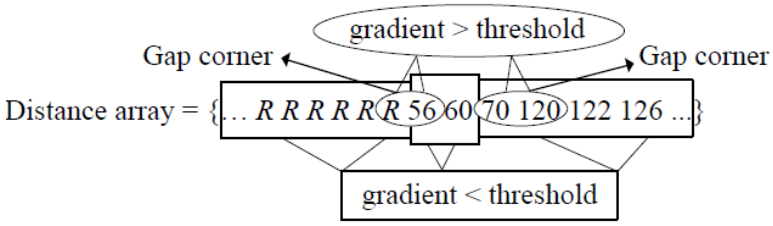

Figure 2. Gap detection process

A symbolic description of the gap detection procedure is given in Figure 2, where the maximum range of LIDAR is given as *R*. The detected gaps are represented by a *N*x2 matrix referred to as gap pair matrix $\mathbf{M_{gp}}$, where *N* is the number of the gaps. $\mathbf{M_{gp}}$ is given as follows.

$$\mathbf{M}_{gp} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \rightarrow \begin{pmatrix} \mathbf{g_1} \\ \mathbf{g_2} \end{pmatrix} \tag{2}$$

The *i-th* row of the gap matrix consists of the LIDAR measurement array indices of the right and the left corners of the *i-th* gap.

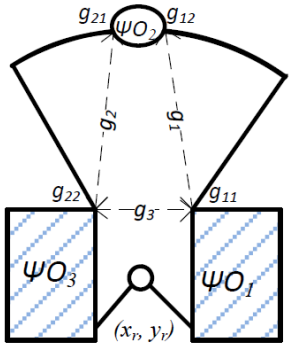

Figure 3. Gap configuration and optimization

In the scenario depicted in Figure 3, randomly placed three obstacles are in mobile robot's field of view. Let g_1 and g_2 be detected gaps; $\psi O1$, ψO_2, ψO_3 be the obstacles; (x_r, x_y) be the position of the mobile robot; $g_{11}, g_{12}, g_{21}, g_{22}$ be the LIDAR indices of the opposite corners of the detected gaps. In order for the mobile robot to move towards either g_1 or g_2, it has to pass through g_3 first. However, g_3 is not one of the gaps detected by the algorithm since the search is performed sequentially in a counter clockwise manner. Therefore, the detected gaps need to be updated. The update procedure is as follows: First, for each gap a counter clockwise search is performed starting from the left corner of the gap to check whether any other gap corner is closer to the right corner of the gap of interest or not. If so, the left corner of the gap is replaced with the closest gap corner. Similarly, a counter clockwise search is performed starting from the right corner of the gap. A pseudo code describing the gap update procedure is given below:

For each g_{i1}
  For each g_{j2}
    if $d(p_{i1},p_{j2}) < d(p_{i1},p_{i2})$
      $g_{i2} = g_{j2}$
      $i=i+j$
    end if
  end for
end for
where, pij is the point corresponding to the related obstacle corner gij

4. Speed Control and Collision Free Maneuvering

The left and the right wheel speeds of the mobile robot were controlled using a PI controller. The angle between the mobile robot and the current target (θ_c) was calculated at each process time. The angular difference between the orientation angle of the robot and θ_c were given as input parameter to the speed controller. The left and right wheel speeds were controlled by a PI type controller by minimizing the angular error.
Points on the line passing through the detected gaps were evaluated through a cost function given in Equation 4 and the point having the minimum cost was assumed to be the current local target. The points which were close to the two opposite corners of the gap less than a certain threshold were discarded to avoid collision. The threshold was set to the diagonal length of the mobile robot.

Let θ_{rl} be the angle between the robot and local target; θ_{rc} be the angle between the robot and the nearest obstacle which is closer than a critical angle of view and range. The critical range depends on the localization accuracy and the size of the mobile robot. By trial and error, the critical angle of view was set to be 120^o. If the absolute difference between θ_{rc} and θ_{rl} is smaller than 60^o, the robot's next movement may cause a collision. The critical angles of view and range parameters depend on the mobile robot dimensions. Therefore, θ_{rl} should be updated based on the angular difference. Calculation of the collision avoided heading direction (θ^u_{rl}) is given in Equation 3.

$$\theta^{u}{}_{rl} = 2\theta_{rl} - \theta_{rc} \tag{3}$$

The collision avoidance process is demonstrated in Figure 4. The red dotted-line illustrates the projected position of the left side of the mobile robot based on its current orientation. To avoid collision, the robot updates its direction based on Equation 3. The collision avoidance process does not change whether the mobile robot is headed towards a local or the global target.

Let p_i and p_r be a point on a line passing through the two opposite corners of a gap and the current position of the mobile robot, respectively. Then, the cost function for the point is defined as,

$$C(p_i) = |p_i - p_r| + |p_{gh} - p_i| + k\Delta\theta \tag{4}$$

, where $|p_i\text{-}p_r|$, $|p_{gh}\text{-}p_i|$ and $\Delta\theta$ are the Euclidian distance between the *i-th* candidate local target and the mobile robot, the Euclidian distance between the *i-th* candidate local target point and the global target and the angular difference between the mobile robot and the candidate local target of interest, respectively. The point which minimizes the cost function given Equation 4 is assumed to be the local target. The term $k\Delta\theta$ is unique to the proposed study not existent in conventional bug algorithms. The term is added to prevent oscillatory behaviors encountered when both the mobile robot and the global target are at equidistant from the opposite corners of an obstacle. In such a case, the sum of the two terms is equal for both corners of the obstacle. Consequently, the robot presents oscillatory behavior between the two corners. By adding the third term, the mobile robot is steered towards the corner of the obstacle which requires less steering effort. The constant *k* is calculated using trial and error.

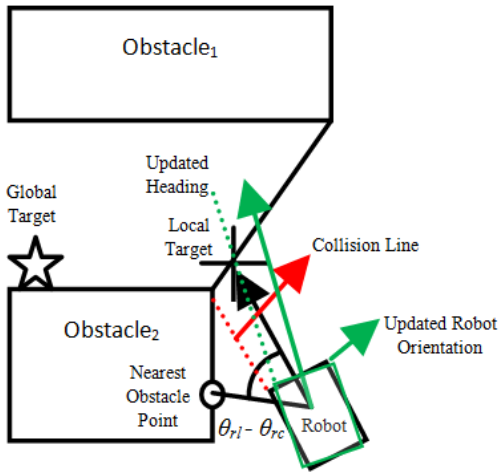

Figure 4. Collision avoidance process

A simulation output is given in Figure 5. The current and updated heading directions are illustrated with green lines. The obstacle segment which may cause collision is illustrated with pink points and collision free heading orientation is named as safe (collision avoided) heading direction. The red circular object illustrates the global target while the red colored rectangle is the mobile robot and the blue colored surfaces represents the LIDAR's view. The range of the LIDAR was simulated as 10 meters with 270^{o} angle of view.

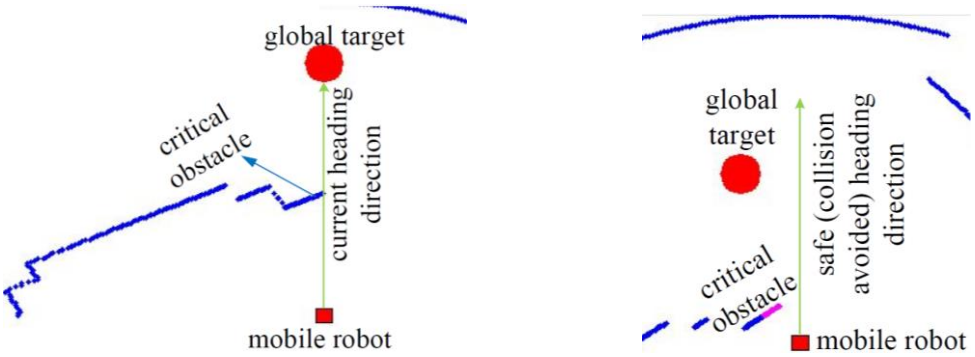

Figure 5. Collision avoidance and updated heading direction

The flow chart of the developed method is given in Figure 6. The abbreviations used in the flow chart are explained in Table 1.

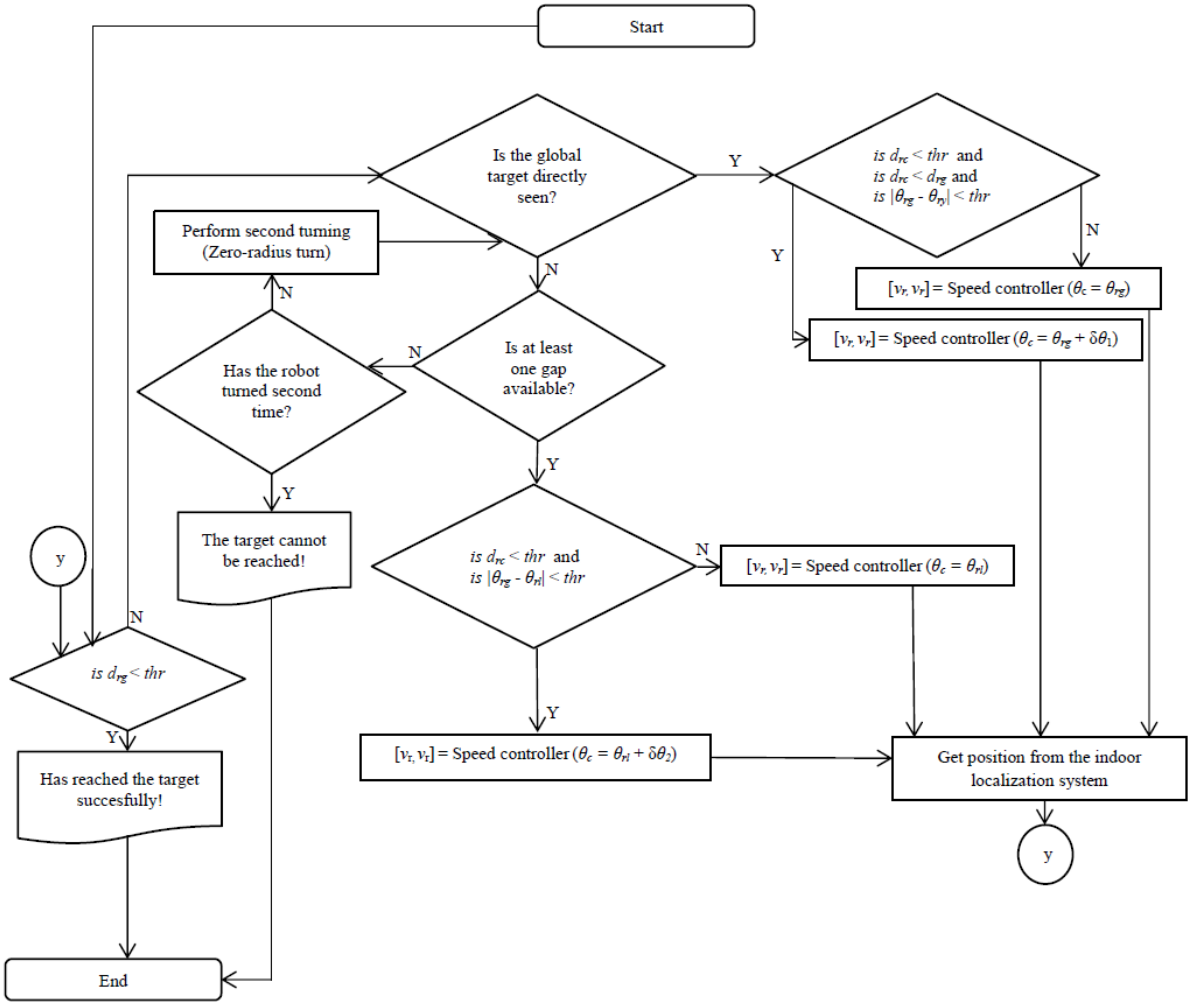

Figure 6. Flow chart of the developed scheme

Table 1. Abbreviations used in the flow chart

d_{rg}	Distance between the robot and the global target		
θ_{rg}	Angle between the robot and the global target		
d_{rl}	Distance between the robot and the local target		
θ_{rl}	Angle between the robot and the local target		
θ_c	Input angle to the speed controller		
d_{rc}	Distance between the nearest obstacle within the critical angle of view		
θ_{rg}	Angle between the robot and the nearest obstacle within the critical angle of view		
$\delta\theta_1=\lvert\theta_{rg} - \theta_{ry}\rvert$		$\delta\theta_2 = \lvert\theta_{rl} - \theta_{ry}\rvert$	
v_r	Right wheel's velocity	v_l	Left wheel's velocity
thr	Threshold (different for each criteria)		

5. Experimental Results

Experiments were executed on MATLAB environment. The collision avoidance and map-free path planning tests were performed during the experiments. The proposed scheme was applied on concave, convex and maze-type environments.

Experimental results including concave and convex maps are given in Figure 8 and 9, respectively.

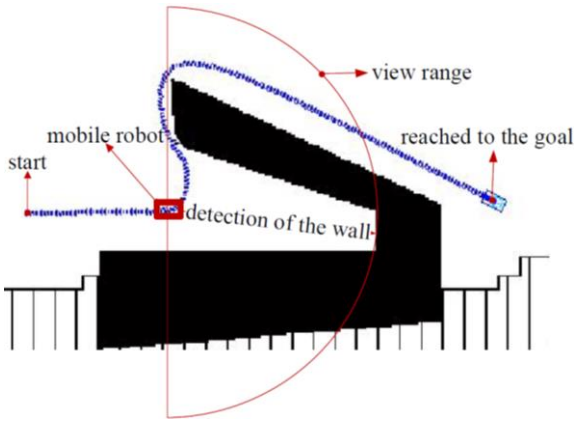

Figure 7. A concave obstacle experiment

Because the robot utilizes only sensor readings and does not have any prior knowledge about the map, the algorithm may not estimate the concave geometry till the view range of the sensor reads the inside wall of the obstacle. As soon as the inside wall is detected by the robot, an alternative gap is searched. The described scenario is illustrated in Figure 7.

The experiment executed for a convex obstacle case is given in Figure 8. Obstacle avoidance in convex obstacle setting is a relatively easy problem for the developed scheme. The obstacle geometry viewed by the robot is not fully depended on the view range of the LIDAR. Therefore, the robot reaches the global target successfully without any collision.

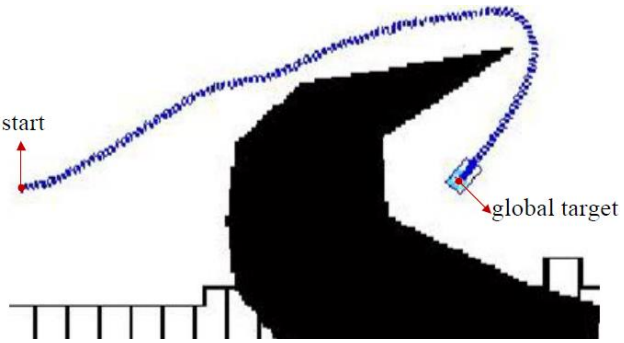

Figure 8. A convex obstacle experiment

Detailed illustration of the maze-type environment test and the LIDAR view on the selected points are given in Figure 9.

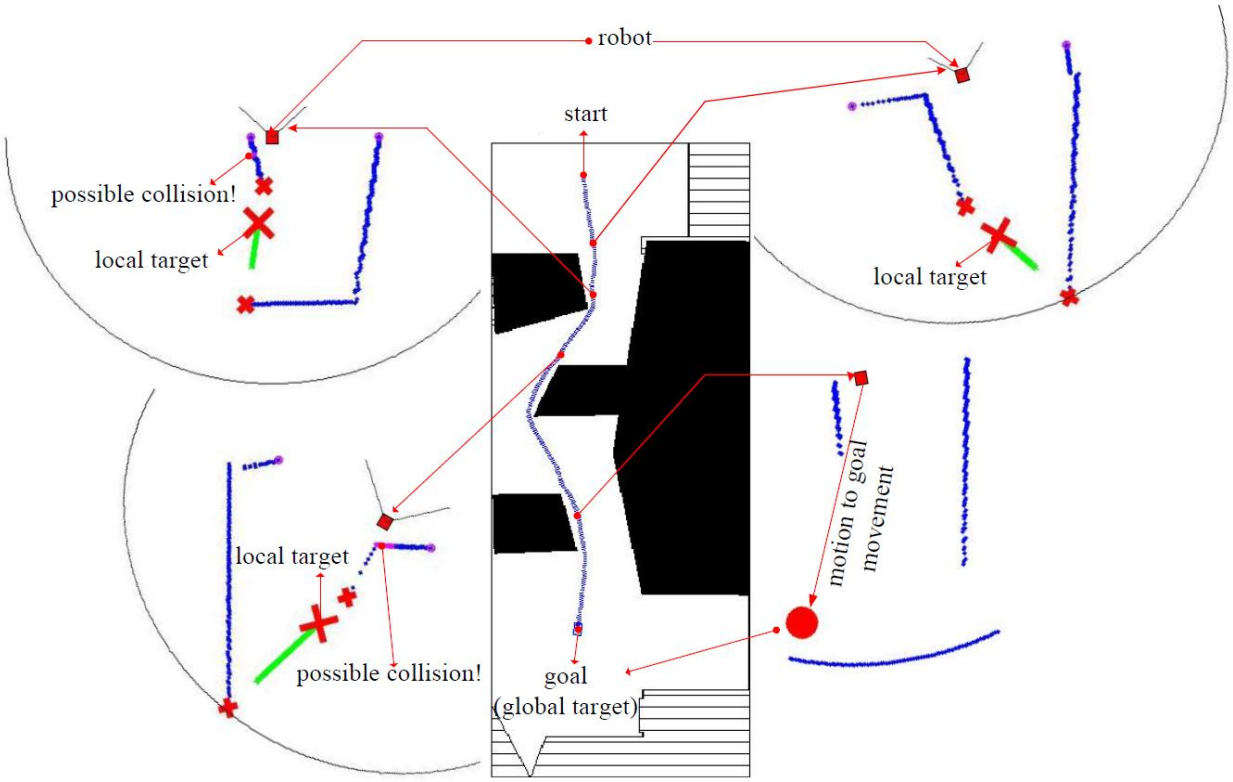

Figure 9. Maze-type map experiment path and details

The blue dotted-line between start and goal points illustrates the path which was planned and traversed by the mobile robot. The red colored dots on the path depict the selected positions of the mobile robot at certain intervals. Local target points were illustrated by a red colored cross while the smaller red crosses depict the opposite corners of each gap. The global target was given as the larger red disk. The collision avoidance process explained in Collision Free Maneuvering section was executed several times during the motion of the robot. The possible collision situations were given by pink surfaces in Figure 9.

Comparison of tangent bug and the proposed scheme in terms of collision avoidance performance under similar scenarios was given in Figures 10 and 11. The tangent bug approach calculates the optimal local target based on the pre-defined cost function without taking collision possibility into account. Therefore, the generated local target may be too close to the obstacle. In Figures 10 and 11, the bare tangent bug approach causes collision before reaching the global target. Employing the collision avoidance procedure present in the proposed scheme, the mobile robot successfully reaches the global target without hitting any obstacles on its way.

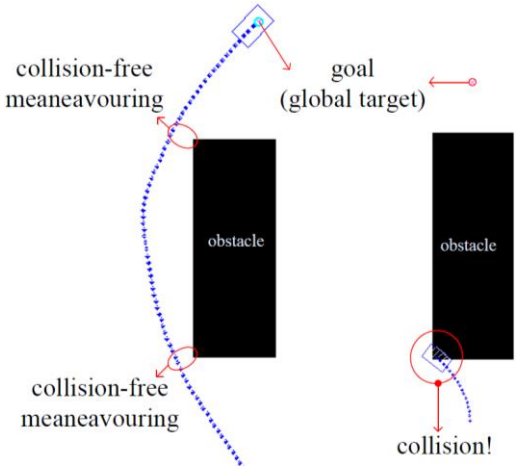

Figure 10. Comparison 1 - between tangent-bug and the proposed scheme based on collision avoidance

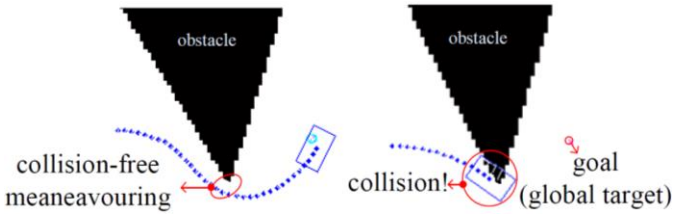

Figure 11. Comparison 2 - between tangent-bug and the proposed scheme based on collision avoidance

6. Conclusion

In this study, a local path planning scheme was developed and an obstacle avoidance approach was proposed. The proposed scheme and methods were applied on different environments. Concave shaped obstacles are challenging for the developed method as the algorithm solely relies on the sensor readings. The information about the environment was limited to the sensor's view angle and range. The motion planning performance can be improved by using a wide range LIDAR and even limited information about the map.

References

[1]. Warren, C.W., Global path planning using artificial potential fields, IEEE International Conference on Robotics and Automation, 1989; 316–321.

[2]. Samadi, M., Othman, M.F., Global Path Planning for Autonomous Mobile Robot Using Genetic Algorithm, International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2013; 726-730.

[3]. Zhengcai, C., Yingtao, Z, Qidi W. "Genetic Fuzzy + PI Path Tracking Control of a Nonholonomic Mobile Robot", Chinese Journal of Electronics, Vol. 20, No.1, 2011.

[4]. Zeng, C., Zhang, Q., Wei, X., Robotic Global Path-Planning Based Modified Genetic Algorithm and A* Algorithm, International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 2011; 167-170

[5]. Khatib, O., Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, The International Journal of Robotics Research, 1986; 5:90-98.

[6]. Schwager, M.,Bullo, F., Skelly, D., Rus, D. , Ladybug Exploration Strategy for Distributed Adaptive Coverage Control, IEEE International Conference on Robotics and Automation, 2008; 2346 – 2353

[7]. Kim, S., Russell, J., and Koo, K., Construction Robot Path-Planning for Earthwork Operations, Journal of Computing in Civil Engineering, 2003; 17; 97-104.

[8]. Hazas, M., Hopper, A., “A Novel Broadband ultrasonic location system for improved indoor positioning”, IEEE Transactions on Mobile Computing, 2006, 5, 5, 536 - 547

[9]. Mautz, R., “Overview of current indoor positioning systems”, Journal of Geodesy And Cartography, 2009, 35, 1, 18-22