Scientific Research and Essays Vol. 7(30), pp. 2744-2753, 2 August, 2012
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE12.351
ISSN 1992-2248 ©2012 Academic Journals

Full Length Research Paper

Improved Bug algorithm for online path planning: Utilization of vision sensor

Weria Khaksar[1]*, S. H. Tang[1] and Mansoor Khaksar[2]

[1]Department of Mechanical and Manufacturing Engineering, Faculty of Engineering, University Putra Malaysia, Malaysia.
[2]Department of Industrial Engineering, Faculty of Engineering, Islamic Azad University, Sanandaj, Iran.

Accepted 16 July, 2012

The area of robot path planning and navigation has been studied by various researchers over the last decades, resulting in a large number of works. One of the most challenging fields in motion planning is dealing with unknown environment, which is known as online path planning. This paper aims to improve one of the most famous methods for online navigation, that is, Bug algorithm, which has been introduced by V. J. Lumelsky. An improved Bug algorithm was provided in our research and the simulation result for several cases shows its advantage in terms of path shortening. Also the justified Bug algorithm was compared with Bug algorithm and some of the well-known approaches in the field of robotic motion planning.

Key words: Bug algorithm, robotic motion planning, path shortening, online path planning.

INTRODUCTION

Some of the most significant challenges confronting autonomous robotics lie in the area of autonomous motion planning. The prototypical task is to find a path for a robot, whether it is a robot arm, a mobile robot, or a magically free-flying piano, from one configuration to another while avoiding collision with obstacles. From this early piano mover's problem, motion planning has evolved to address a huge number of variations on the problem, allowing application in areas such as animation of digital characters, surgical planning, automatic verification of factory layout, mapping of unexplored environments, navigation of changing environments, assembly sequencing, and drug design (Choset et al., 2005).

During the past few decades, one of the most attractive and useful fields in robot motion planning is robot navigation in unknown environment. This kind of problem is known as online, sensor-based, local, real-time, or reactive motion planning (Masehian and Amin-Naseri, 2008). In sensor-based motion planning, there are two different varieties of sensors, namely Touch and Vision. A touch sensor detects when the robot touches an obstacle. A vision sensor typically provides the information visible to the robot (Rao et al., 1993). One of the first researches in the field of online motion planning is the Bug algorithm presented by V. J. Lumelsky for a point robot to traverse from a start point to a goal point in which the robot is equipped with a touch sensor in an environment with different shaped obstacles (Lumelsky and Stepanov, 1986). A survey on preliminary researches for online motion planning is presented in Rao et al. (1993). One of the most important methods for solving the online motion planning problem is Potential Fields (PFs) introduced by Khatib (1986). This approach treats robot as a point which is influenced by an artificial potential field. This function can be defined over free space as the sum of an attractive potential which is pulling the robot toward the goal configuration, and a repulsive potential that is pushing the robot away from the obstacles. A real-time motion planner is presented by Lengyel et al. (1990), which uses standard graphics hardware to rasterize configuration space obstacles into

*Corresponding author. E-mail: GS22153@mutiara.upm.edu.my. Tel: +601 266 77160. Fax: 603 8656 7122.

a series of bitmap slices, and then uses dynamic programming to create a navigation function and to calculate paths in this rasterized space. Moreover, another method is presented in Masehian et al. (2003) for online motion planning through incremental construction of medial axis. Gabriely and Rimon (2009) introduced a notion of competiveness suitable for online mobile robot navigation in general planar environment. They described CBUG, an online navigation algorithm for a size D disc robot moving in general planar environment. Most collision avoidance methods do not consider the vehicle shape and its kinematic and dynamic constraints, assuming the robot to be point-like with no acceleration constraints. In this paper, Minguez and Montano (2009) methodology is used to consider the exact shape and kinematics, as well as the effects of dynamics in the collision avoidance layer, since the original avoidance method does not address them. In Erfanian and Kabganian (2009), an adaptive trajectory tracking controller is proposed for a single flexile-link manipulator with presence of friction in the joint and parameter uncertainty. They employed a distributed-parameter dynamic modelling approach to design the controller.

In addition to the classic motion planning methods, other optimization approaches which are known as heuristic approaches have been increasingly employed for planning and optimization of robot motions. This group of approaches do not guarantee finding a solution, but if they do, they are likely to do so much faster than the other approaches (Khaksar et al., 2012). Most important heuristic approaches in the field of motion planning are Neural Network, Genetic Algorithms, Fuzzy Logic and Tabu Search. In Ghatee and Mohades (2009), an approach for motion planning is presented in order to optimize the length and clearance, applying a Hopfield neural network. A hybrid genetic algorithm based optimum path planning approach for mobile robots is proposed by Li et al. (2006). A fuzzy logic-based approach for mobile robot path tracking is presented in Antonelli et al. (2007). The proposed model is a path following approach based on a fuzzy-logic set of rules which emulates the human driving behaviour. An algorithm for reactive navigation of mobile robots was proposed in 2008 by Motlagh et al. (2009) using fuzzy artificial potential field. A novel target switching technique was included for local minimum avoidance. In Masehian and Amin-Naseri (2008), a new online motion planner was developed, based on the Tabu search approach. Various components of the classic Tabu search have been remodelled and integrated in a single algorithm to

craft a motion planner capable of solving varieties of exploration and goal-finding problems. A real time object tracking approach is proposed by Uzer and Yilmaz (2011) based on robot vision. Their robotic system is based on a fuzzy controller and image-based controller. A neuro-genetic approach to increase the kinematics solution of robotic manipulators is presented by Koker (2011).

In this research we focus on the Bug algorithm and aim to improve the drawbacks of this approach. In the following parts of this paper, after a brief introduction about Bug algorithm, the deficiencies of this approach are mentioned. Then our improved Bug algorithm is presented and compared with classic Bug algorithms and some other approaches.

MATERIALS AND METHODS

Bug Algorithm

Perhaps the most straight forward path planning approach is to move toward the goal, unless an obstacle is encountered, in which case, circumnavigate the obstacle until motion toward the goal is once again allowable. In this point of view, the robot is equipped with some touch sensors on its perimeter which can detect the obstacle just when the robot reaches one. This is the main idea of Bug algorithm. Essentially, the Bug algorithm formulates the “common sense” idea of moving toward the goal and going around obstacles. The robot is assumed to be a point with perfect positioning with a contact sensor that can detect an obstacle boundary if the robot touches it. The robot can also measure the distance $d(a,b)$ between any two points a and b. Finally, assume that the work space is bounded. The start and goal points are labelled q_{start} and q_{goal}, respectively. Let $q_0^L = q_{start}$ and the m-line be the line segment that connects q_i^L to q_{goal}. Initially, i=0.

The Bug I algorithm exhibits two behaviours: motion-to-goal and boundary-following. During motion- to-goal, the robot moves along the m-line toward q_{goal} until it either encounters the goal or an obstacle. If the robot encounters an obstacle, let q_1^H be the point where the robot first encounters an obstacle and call it a hit point. The robot then circumnavigates the obstacle until it returns to q_1^H.“Then, the robot determines the closest point to the goal on the perimeter of the obstacle and traverses to this point. This point is called a leave point and is labelled q_1^L. From q_1^L, the robot heads straight toward the goal again. If the line that connect q_1^L and the goal intersect the current obstacles, then there is no path to the goal. Otherwise, the index “i” is incremented and this procedure is then repeated for q_i^L and q_1^H until the goal is reached or the planner determines that the robot cannot reach the goal.

Like its Bug I sibling, the Bug II algorithm exhibits two behaviours; motion-to-goal and boundary-following. During motion-to-goal, the robot moves toward the goal on the m-line; however, in Bug II the m-line connects q_{start} and q_{goal}, and thus remains fixed. The boundary following behavior in invoked if the robot encounters an obstacle, but this behavior is invoked if the robot encounters an obstacle; however this behaviour is different from that of Bug I. For Bug II, the robot circumnavigates the obstacle until it reaches a new point on the m-line closer to the goal than the initial point of contact with the obstacle. At this time, the robot

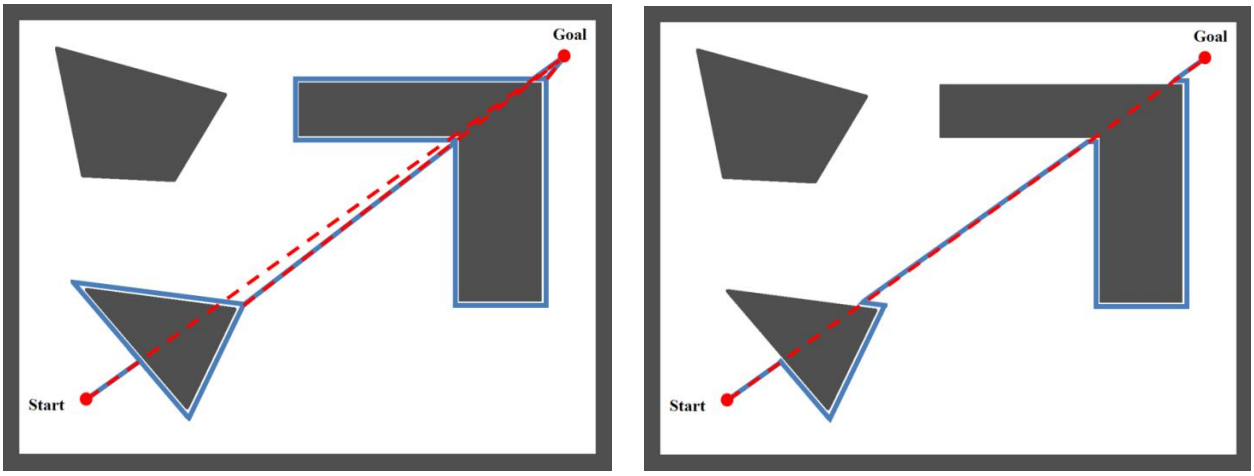


Figure 1. The performance of Bug algorithms. (a) A path by using the Bug I and (b) A path from Bug II algorithm.

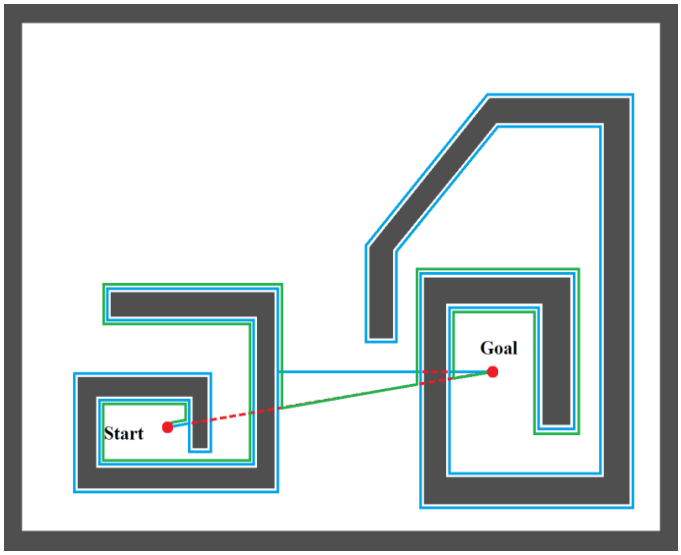


Figure 2. Bug I & II performance in a maze-like space. Blue and green lines correspond to Bug I and Bug II paths respectively.

proceeds toward the goal, repeating this process if it encounters an object. If the robot re-encounters the original departure point from the m-line, then there is no path to the goal.

At first glance, it seems that Bug II is a more effective algorithm than Bug I because the robot does not have to entirely circumnavigate the obstacles; however this is not always the case. The following formulations provide a comparison between these two algorithms (Rao et al., 1993):

(1) $L_{Bug\,I} \leq d(q_{start}, q_{goal}) + 1.5 \sum_{i=1}^{n} p_i$

(2) $L_{Bug\,II} \leq d(q_{start}, q_{goal}) + 0.5 \sum_{i=1}^{n} n_i p_i$

Where p_i is the perimeter of the ith obstacle. The line through q_{start} and q_{goal} intersects the ith obstacle n_i times. Naturally (2) is an upper-bound because the summation is over all of the obstacles as opposed to over the set of obstacles that are encountered by the robot.

Figure 1 present two paths that are obtained from Bug I and Bug II algorithms. As can be observed in Figure 1, the Bugs Algorithms does not try to reduce the length of the paths; rather, they try to circumnavigate the obstacles in order to reach the goal. This is the main problem of these approaches, especially when we are dealing with concave and maze-like obstacles.

Figure 2 provide an example of a maze-like environment in which the paths of the Bug algorithm are not effective enough.

Several extensions of Bug algorithms have been proposed in the literature with different advantages and drawbacks and also different objectives. The μNav algorithm was presented in Mastrogiovanni et al. (2009) that is, a novel approach to navigation which, with minimal requirements in terms of on-board sensory, memory and computational power, exhibits way-finding behaviours in very complex environments. The algorithm is intrinsically robust, since it does not require any internal geometrical representation or self-localization capabilities. In Zhu et al. (2012), a new bug-type algorithm termed Distance Histogram Bug (DH-Bug) is proposed for overcoming the existing limitations in previous works such as generating long path, limited to static environments as well as ignoring implementation issues.

We focus on the quality of the generated paths by the bug planner and design an improved bug algorithm which guides the robot to reach the goal position through shorter paths.

Justified Bug Algorithm

One of the most challenging fields in motion planning is to find shorter paths in less time. Although the first objective of a motion planner is to find a free path to the goal, that is, avoiding obstacle collision, the second objective will be to decrease the length of the path and the time which the planner finds the path. A considerable amount of researchers in the field of motion planning are trying to improve the existing algorithms in terms of time and length of the

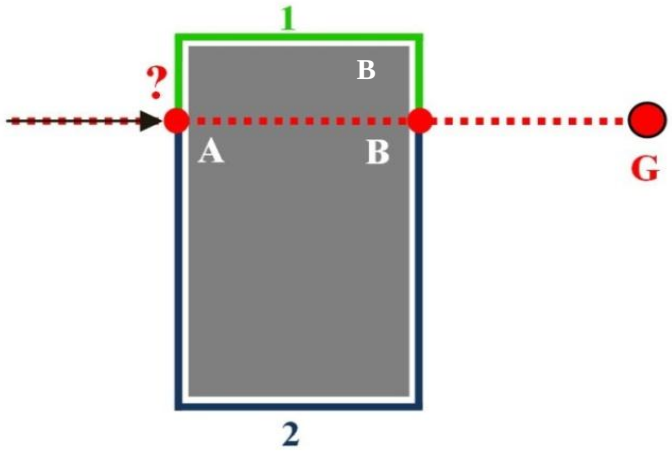


Figure 3. The algorithm cannot determine which course is better.

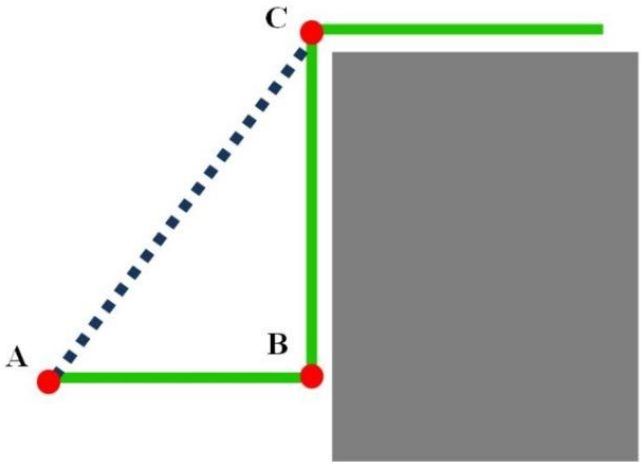


Figure 4. Unnecessary segment which can be omitted.

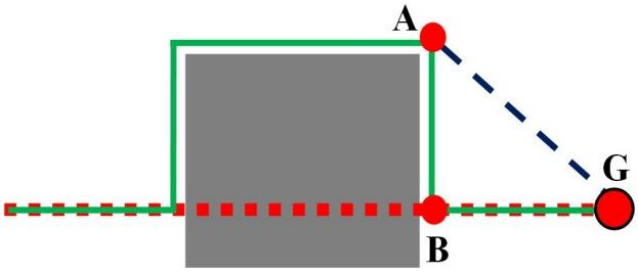


Figure 5. Checking goal access.

path.

As mentioned before, Bug I and Bug II algorithms just try to reach the goal by circumnavigating around the obstacles as there is nothing about path shortening. There are three deficiencies in Bug algorithms which results in generating long paths.

Firstly, when the robot should circumnavigate an obstacle, there is no regulation to guide the robot which course is better for going around the obstacle, that is, clockwise or counter clockwise. This drawback more happens in Bug II. In some cases with complex shaped obstacles, choosing the right course for navigation around the obstacle is a critical factor and can increase the length of the path intensively. In Figure 3, when the robot arrives in point A, there is no component in the algorithm which determines which path is better for going from point A to point B, that is, which path is better; 1 or 2.

Because of the online nature of bug algorithms, that is, there is no previous knowledge about the position and the shapes of the obstacles, and also because of the performance of touch and vision sensors; this problem seems hard to be solved. Until the robot receives enough information about the shape of an obstacle, it could not find the best course. This research is not going to improve this drawback, but it is one of the suggested subjects for future researches.

Secondly, there is no component in the algorithm that omits the unnecessary segments of the path. Figure 4 shows a case where there are unnecessary segments in the path which can be omitted.

As shown in Figure 4, according to the Bug algorithm, the robot navigates from point A to B and then to C (the green lines). But if the robot moves directly from A to C (the blue line), then it can omit two unnecessary segments, that is, AB and BC.

Thirdly, there is no goal access checker which in each step of the algorithm, checks whether there is a direct path to the goal without obstacles collision or not. In Figure 5, when the robot arrives in point A, there is a free path to the goal but in order to follow the algorithm's roles, the robot should first move to the new point on the m-line B, and then traverse to the goal point.

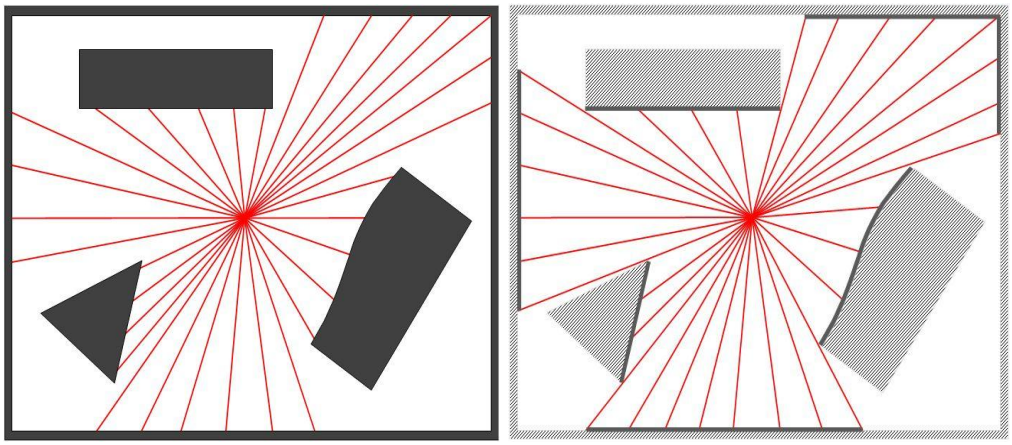
In this paper, we introduce a justifier component which improves the Bug algorithm in terms of its deficiencies. In our method, instead of the touch sensors, the robot is supposed to be equipped with enough number of range sensors which are located on its perimeter. Each sensor is qualified to project a ray to find out its distance from any visible obstacle. Figure 6 shows a robot which is equipped by these sensors.

The adjunct component provides two corrective steps. The first step will be added to the algorithm in order to omit unnecessary segments of the path. Second step is for checking the access to the goal.

First step: This step acts simultaneously with the algorithm. As explained before, in the Bug algorithm, the robot moves directly to the goal through the m-line till it reaches a hit point. There, the robot uses its touch sensors to determine the obstacle's boundary and then circumnavigate around it. In this step of the justifier component, from the start point, the robot uses the vision sensors and detects the visible obstacle's perimeter which interests the m-line. Before any movement through the m-line, the robot detects the hit point and the most far visible points on the perimeter of the obstacle. A visible point is a point which could be detected by the vision sensors. In Figure 6, the end point of each ray is a visible point. If there are more than one alternative point, the robot chooses the point which minimizes a cost function. The cost function is defined as follows:

$$C_i = \alpha d_i^c + \beta d_i^g \quad (3)$$

"Which d_i^c is the distance between current position of the robot and



(a) (b)

Figure 6. (a) Performance of a vision sensor (b) Visible regions for the robot.

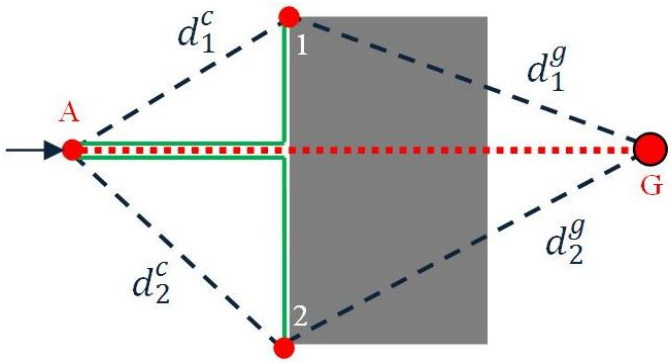


Figure 7. A case with two alternative points.

the ith alternative point, and d_i^g is the direct distance between the ith alternative point and the goal point. α and β are justifier coefficients which help this cost function to be justified for different cases. Figure 8 shows an example of an obstacle with two alternative points.

According to Figure 7, when the robot arrives in point A, instead of using the previous algorithm and moving through the green lines, it determines two alternative points, 1 and 2. For each point, the robot measures d_i^c and d_i^g and by comparing C_1 and C_2, decides which alternative point is better. This procedure will be repeated in each point until the robot reaches the goal.

Second step: Like the first step, this step performs in the same time with the Bug algorithm. In this step, for each point, the robot uses its vision sensors to check whether there is a collision free path directly to the goal point or not. If such path is available, the robot moves directly to the goal and terminates the rest of the algorithm. As shown in Figure 6, when the robot arrives in point A, it finds a free path to the goal point so there is no need to continue to the algorithm. A general scheme of the justified Bug algorithm is presented below.

Check whether the goal point is reachable or not, and if it is, move directly to the goal and terminate the rest of the algorithm. Detect the hit point through the m-line. Find the most far points in the perimeter of the current obstacle. If there are more than one alternative point, use the cost function to choose the best point. Repeat these steps until you reach the goal or find that there is no way to the goal.

RESULTS AND DISCUSSION

The algorithm was run for several problems ranging from simple convex to highly concave polygons and mazes and succeeded in performing effectively. Some of the simulations are shown in Figure 9(a)–(h). The running times were within a few seconds using a 2 GHz Intel Core 2 Duo processor. All the studies were simulated with MATLAB 7.7.0.471 (R2008b).

The simulation results show that the justified algorithm performs successfully in different configuration spaces, especially in maze-like and uncluttered areas. In Figure 9(d), the space is an uncluttered one with several obstacles scattered about the space and the algorithm could guide the robot to navigate among them productively. The performance of the justified algorithm in maze-like spaces is remarkable since it can find a free and relatively short path without being impressed by the confusing shape of the obstacles. Figure 9(h) shows a configuration which is a prepared case for trapping in local minima. The powerful ability of the justified algorithm in such a case is that the algorithm does not insist on moving directly to the goal which is one of the most important reasons for trapping in local minima.

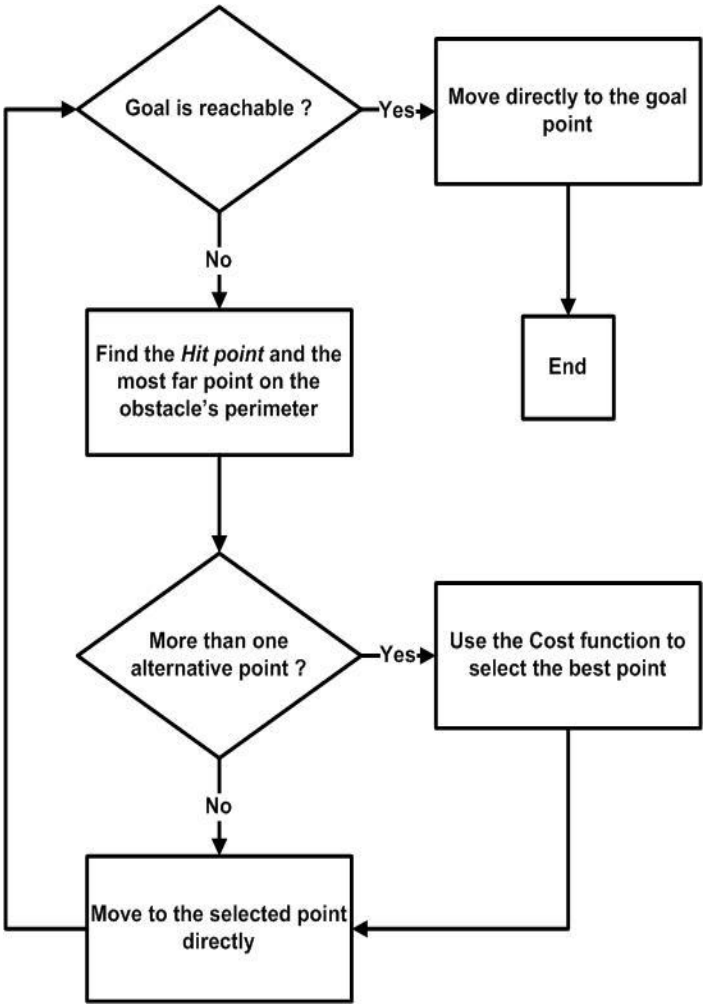


Figure 8. The flowchart of our proposed algorithm.

The performance of this algorithm was compared with several algorithms as shown in Figure 8. Since on-line method acquire their knowledge of environment by sensors and plan their path locally, it would be incorrect to compare off-line and on-line methods in term of processing time.

In the comparison studies, the off-line methods have been used just to show the path generated by each algorithm (Figures 10 and 11). According to Figure 10, the performance of the justified Bug algorithm seems to be more effective than Bug I and Bug II. By comparing the processing time and length of the path which is provided in Table 1, some interesting results become clear.

Among the abovementioned approaches, the only approach which is better in term of processing time is potential fields and the justified Bug algorithm requires less time than the rest of the algorithms. The length of the path in justified Bug algorithm is the shortest one and the percentage of improvement in term of length of the path is 25.69% for Bug I, 31.23% for Bug II, and 16.43% for the average length.

Conclusion

This paper is a research in one of the most challenging fields of motion planning which is online robot navigation. One of the important approaches in this field is Bug algorithm. In this algorithm, the robot move directly to the path until it reaches an obstacle. Then it tries to circumnavigate around the obstacle using its touch sensors. This algorithm suffers many drawbacks. The first drawback is that the algorithm cannot omit the unnecessary segment of the path. The second is ability of the robot to move to the goal directly when there is no

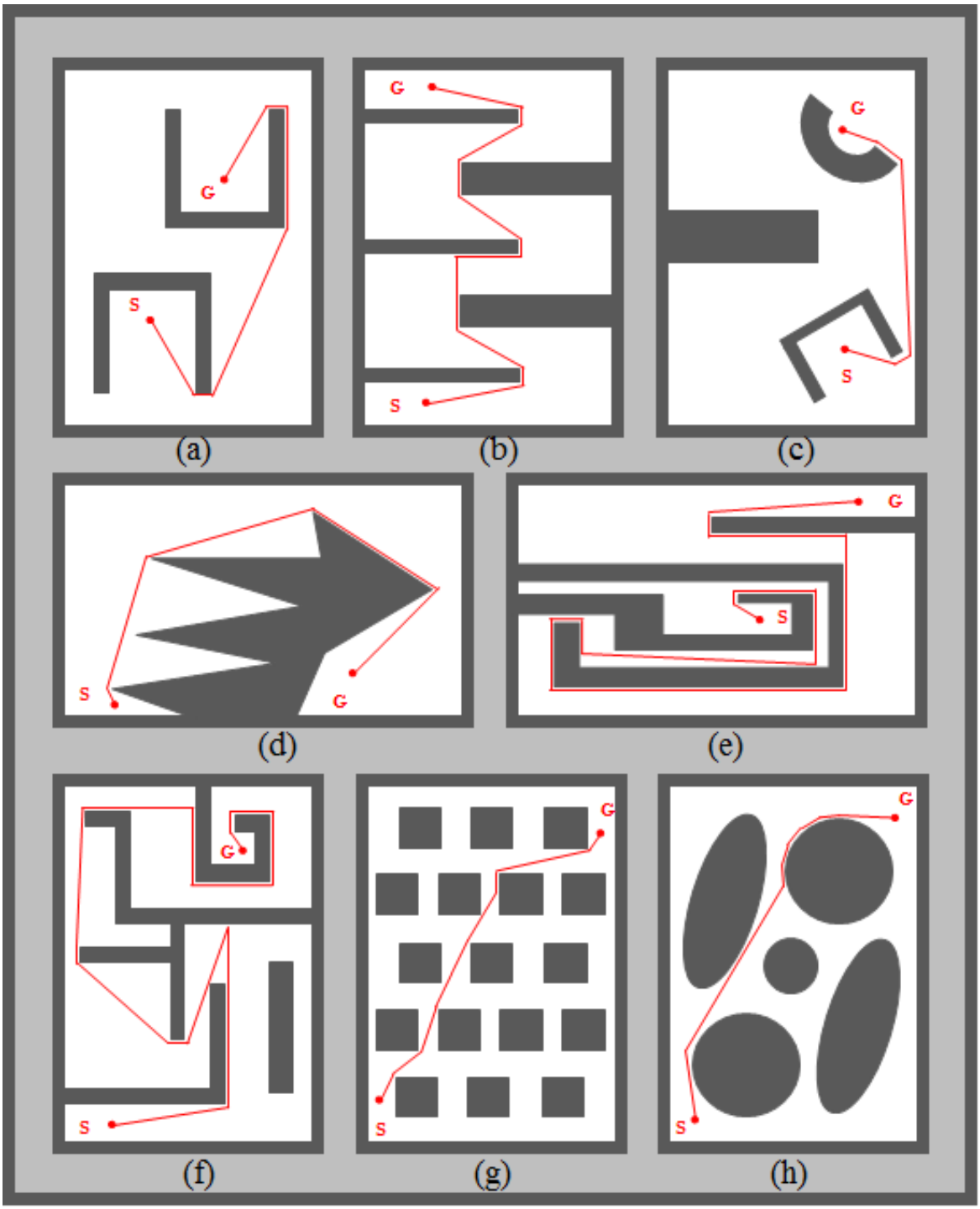


Figure 9. Simulation results for the Justified Bug algorithm in different configuration spaces.

obstacle between the robot and the goal point. In order to improve the Bug algorithm in terms of its deficiencies, this research provides a justifier component which contains two major steps, but the key concept of this justified algorithm is using vision sensors instead of the touch sensors.

In the first step, the robot uses its vision sensors which is located on its perimeter and finds the first hit point on

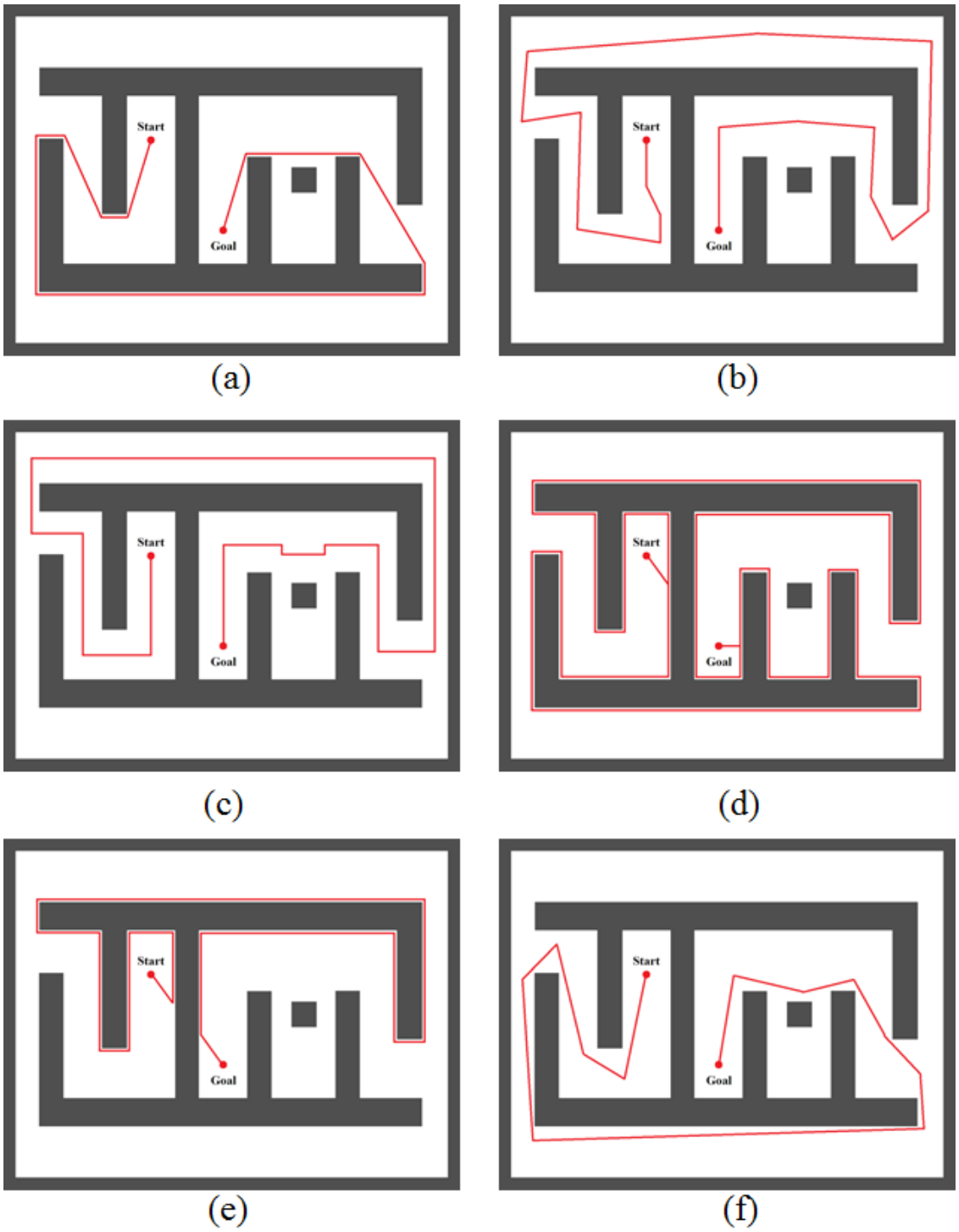


Figure 10. The comparison of our method with different approaches: (a) Visibility Graph (b) Potential fields (c) Voronoi Diagram (d) Bug I (e) Bug II and (f) Justified Bug.

the m-line. After that, instead of moving through the m-line, the robot finds the most far visible points and uses a cost function to choose the best one. The robot then navigates directly to this locally best point. This procedure will continue till the robot reaches the goal point. In the second step the robot checks each point to

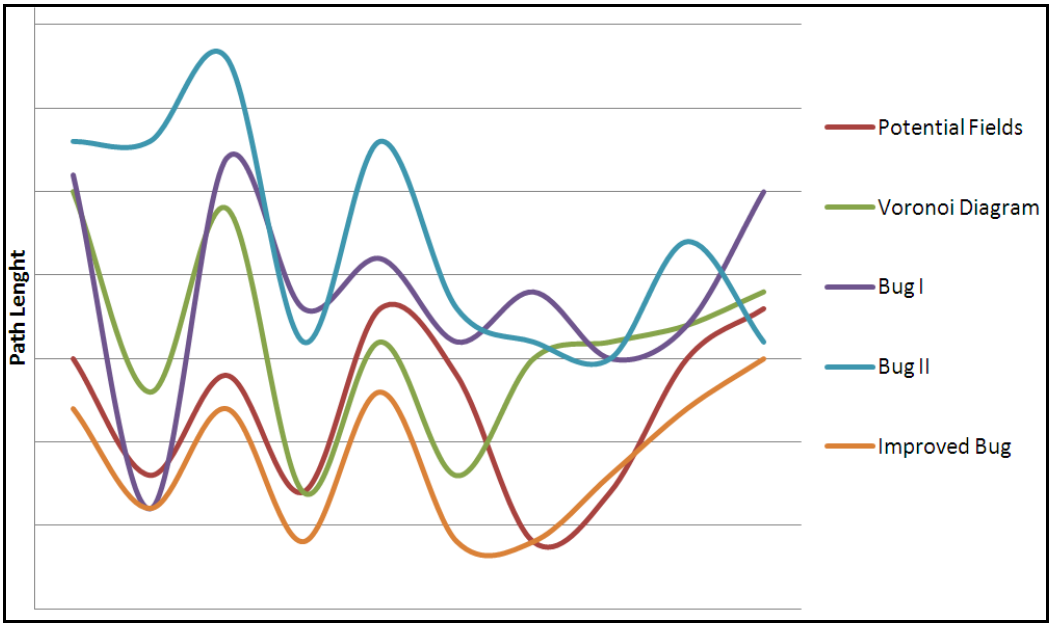


Figure 11. Comparing the path length for different algorithms.

Table 1. Time and length of the path for compared approaches.

	Approach	Time (second)	Length
a	Visibility graph	5.4	27.7
b	Potential fields	2.6	33.1
c	Voronoi diagram	6.1	37.3
d	Bug I	6.4	39.7
e	Bug II	4.4	42.9
f	Justified Bug	4.8	29.5
	Average	4.95	35.3

find out whether the goal point is reachable or not, and if it is, moves directly to the goal and terminates the rest of the algorithm.

These two steps act simultaneously with the Bug algorithm. The performance of the justified Bug algorithm checked with simulation study. The performance seems to be much faster and provides shorter path than the old Bug algorithm. Also the processing time and the length of the path were compared with some important approaches in the field of motion planning. The results show that the justified Bug algorithm remarkably provides better paths.

One of the important deficiencies of the Bug algorithm that was not attended to in this study is what happens when the robot should decide which course is better to circumnavigate around the obstacle, that is, clockwise or counter clockwise. This problem may be solved by using a special kind of vision sensors which is called continuous vision sensor. As the robot navigates along a path, a continuous vision sensor can detect all parts of the terrain that are visible. If this kind of sensor could provide enough information about the complete shape of each obstacle, the robot can choose the best course.

REFERENCES

Antonelli G, ChiaveriniS, Fusco A (2007). A Fuzzy-Logic-Based Approach for Mobile Robot Path Tracking. IEEE T FuzzySyst. 15(2):211-221.

Choset H, Lynch K, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005), Principles of Robot Navigation Theory, Algorithms, and Implementation. Cambridge, Massachusetts: MIT Press.

Erfanian V, Kabganian M (2009).Adaptive trajectory control and friction compensationof a flexible-link robot.Sci Res Essays 4(4):239-248.

Gabriely Y, Rimon E (2008). CBUG: A Quadratically Competitive Mobile Robot Navigation Algorithm. IEEE T Robot 24(6):1451-1457.

Ghatee M, Mohades A (2009). Motion Planning in Order to Optimize the

Length and Clearance applying a Hopfield Neural Network. Expert Syst. Appl. 36(3):4688-4695.
Khaksar W, Tang SH, Ismail NB, Arrifin MKA (2012). A Review on Robot Motion Planning Approaches. Pertanika J. Sci. Technol. 20(1):15-29.
Khatib O (1986). Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.Int J Robot Res. 7(1):90-98.
Koker R (2011). A neuro-genetic approach to the inverse kinematics solution of robotic manipulators.Sci. Res. Essays 6(13):2748-2794.
Lengyel J, Reichert M, Donald BR, Greenberg DP (1990). Real-time robot motion planning using rasterizing computer graphics hardware. In Computer Graphics (Proceedings of ACM SIGGRAPH 1990), 24:327-335.
Li Q, Tong X, Xie S, Zhang Y (2006). Optimum Path Planning for Mobile Robots Based on a Hybrid Genetic Algorithm. Sixth Int. Confer. Hybrid Intell. Syst. pp. 53-56.
Lumelsky VJ, Stepanov AA (1986). Dynamic Path Planning for a Mobile Automation with Limited Information on the Environment. IEEE Trans. Autom. Control 31(11):1058-1063.
Masehian E, Amin-Naseri MR (2008). Sensor-Based Robot Motion Planning: A Tabu Search Approach. IEEE Robot Autom. Mag. 15(2):48-57.
Masehian E, Amin-Naseri MR, Khadem ES (2003).Online Motion Planning Using Incremental Construction of Medial Axis. In Proceedings of IEEE International Conference on Robotics and Autom. 3:2928-2933.
Mastrogiovanni F, Sgorbissa A, Zaccaria R (2009). Robot Navigation in an Unknown Environment With Minimal Sensing and Representation. IEEE T Syst. ManCyB 39(1):212-229.
Minguez J, Montano L (2009). Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot. IEEE T Robot 25(2):367-381.
Motlagh ORS, Tang SH, Napsiah I (2009).Development of a new minimum avoidance system for abehaviour-based mobile robot. Fuzzy Set Syst. 160(13):1929-1946.
Rao NSV, Kareti S,Shi W, Iyenagar SS (1993). Robot Navigation in Unknown Terrains: Introductory Survey of Non-Heuristic Algorithms, Oakridge National Lab., Tech. Rep. ORNL/TM-12410.
Uzer MS, Yilmaz N (2011).A real-time object tracking by using fuzzy controller for vision-based mobile robot.Sci Res Essays 6(22):4808-4820.
Zhu Y, Zhang T, Song J, Li X (2012). A new bug-type navigation algorithm for mobile robots inunknown environments containing moving obstacles. Ind Robot 39(1):27-39.