Performance Comparison of the BUG's Algorithms for Mobile Robots

Conference Paper · June 2009

DOI: 10.13140/RG.2.1.3616.6564

2 authors:

Alpaslan Yufka
ASELSAN Inc.
37 PUBLICATIONS **105** CITATIONS
SEE PROFILE

Osman Parlaktuna
Eskisehir Osmangazi University
81 PUBLICATIONS **530** CITATIONS
SEE PROFILE

Performance Comparison of the BUG's Algorithms for Mobile Robots

Alpaslan YUFKA and Osman PARLAKTUNA
Eskişehir Osmangazi University, Eskişehir, Turkey
ayufka@gmail.com, oparlak@ogu.edu.tr

Abstract

In this study, Bug1, Bug2, Class1, Alg1, Alg2 and DistBug motion planning algorithms for mobile robots are simulated and their performances are compared. These motion planning algorithms are applied on a Pioneer mobile robot on the simulation environment of MobileSim. Sonar range sensors are used as the sensing element. This study shows that mobile robots build a new motion planning using the Bug's algorithms only if they meet an unknown obstacle during their motion to goal. Each of the Bug's algorithms is tested separately for two identical indoor environments. At the end of this study, the performance comparison of the Bug's algorithms is shown.

1. Introduction

Based on the environment and the goal location, generating a path for a mobile robot means finding a continuous route starting from the start point, **S**, to the goal point, **G**, which is in a 2D environment with unknown obstacles of an arbitrary shape. Implementing this, the navigation takes an important place and is a general problem for mobile robots.

In the literature, there are several distinct approaches related to the navigation [1]. The Bug's algorithms [2, 3] construct a path to move the robot to **G** in a straight line, if the path to **G** is clear. Robot follows boundary of the obstacle when it encounters an unknown obstacle until the path is clear again [1]. These algorithms have advantages compared with the others [1] that the mobile robot does not have to map its environment. The robot is only interested in reaching its goal location if it is reachable; however, if it is unreachable, the robot is able to terminate the given task. Each described Bug's algorithm has this termination feature [4].

The following Bug's algorithms are considered and implemented in this paper: Bug1 [5, 12], Bug2 [5, 12], Class1 [13], Alg1 [7-8-13], Alg2 [7-8-13] and DistBug [4-6] which are the part of the Bug's family. Bug1 and Bug2 are the original algorithms of the Bug's family which need minimal memory requirements but not to be capable of making the best use of the available sensory data to generate short paths [11]. In contrast, DistBug uses range sensors efficiently to define a new leaving condition and to choose an initial boundary following direction with respect to local range data [11]. Beside this, the algorithms in the Bug's family can exhibit a statistically better performance depending on the structure of the environment, but at different environments the one may have an advantage over another algorithm [4].

This paper presents a comparison among the Bug's algorithms, as well as Bug1, Bug2, Class1, Alg1, Alg2 and DistBug, to show which one exhibits the best performance based on experimental data provided from the simulation environment of MobileSim [14]. A Pioneer mobile robot is used in the simulation. The aim of the study is to compare the algorithms in terms of total path length traveled. The assumptions are that the mobile robot never encounters a localization problem and it uses its odometry to localize itself.

2. The bug's algorithms

The Bug's algorithms are simple planners with provable guarantees [12]. When they encounter an unknown obstacle, they can readily generate their own path contouring the object in the 2D surface if a path to the goal exists. The purpose is to produce a collision-free path by using the boundary-following and motion-to-goal behaviors. In addition, the Bug's family has three assumptions about the mobile robot **i)** the robot is a point, **ii)** it has a perfect localization and **iii)** its sensors are precise [4]. Based on these criterions, it is almost difficult to implement the Bug's algorithms on mobile robots in the realistic world whereas the simulation programs, as well as MobileSim, satisfy these conditions and provides an easy implementation for the Bug's algorithms on Pioneer mobile robots.

2.1. Bug1 and bug2 algorithms

Bug1 and Bug2 algorithms by Lumelsky and Stepanove are not only the original and the earliest sensor-based planners but also offer minimal memory requirements because of their plainness [11, 12]. Their senses are based on tactile sensors but in this study the sonar sensors instead of tactile sensors are used to implement the experiment.

Bug1 is an algorithm in the sense, the mobile robot moves towards **G** directly along a line connecting **S** and **G**, unless it encounters an obstacle, in which case the robot explores the external lines of the obstacle until the motion to **G** is available again [8, 12]. Once it encounters an obstacle as shown in figure 1, it goes around the obstacle in clockwise sense (default), and determines the leave point by calculating the distance between the current position and **G** during travelling around the object. The leaving point is the closest point around the obstacle to the goal. Then, the mobile robot determines the shortest path to this closest point in order to reach the leave point, and changes or keeps its direction of the boundary following according to the shortest path to return to the leave point. Next, it moves to the departure point and leaves the obstacle towards **G** along a new line. When it encounters the second obstacle, the same procedure is applied. This method is inefficient but guarantees that the mobile robot is able to arrive any reachable goal point, [1].

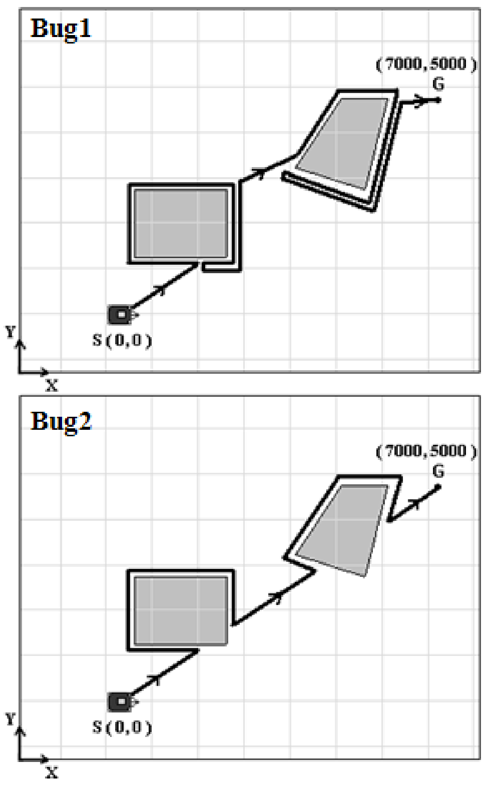

Figure 1. Path generated by Bug1 and Bug2 for the environment1 respectively

Algorithm Bug2 is a greedy algorithm that the mobile robot follows a constant slope computed initially between **S** and **G**. The robot maintains its motion to the goal unless the path on the slope is blocked by an obstacle, in which case, it follows edges of the obstacle by using its sonar sensors in the clockwise sense until it finds its initial slope again. This property leads to shorter paths than Bug1's but in some cases the path may get longer than Bug1's, for example, maze-searching [11]. In figure 1, it is obvious that Bug 2's path is the shorter than Bug1's. On the other side, in the next section 4, the study shows how Bug1 beats Bug2 for environmet2.

2.2. Class1 algorithm

Class1 by Noborio is an algorithm which follows the boundary of the obstacle until the distance to **G** is larger than the minimum distance, whose point is $\mathbf{C_i}$, stored by the mobile robot. In addition, Class1 limits its searching space as the circle whose center is at **G,** and its radius is as the segment **GC** [13]. Beside these, it also constrains its searching space using hit points, $\mathbf{H_i}$. Based on the points $\mathbf{C_i}$ and $\mathbf{H_i}$, Class1 generates its circular loops for the searching space as shown in figure 2 so that it can leave the obstacle and approach to **G** passing from the outer circular loop to inner one which is closer than the previous outer one.

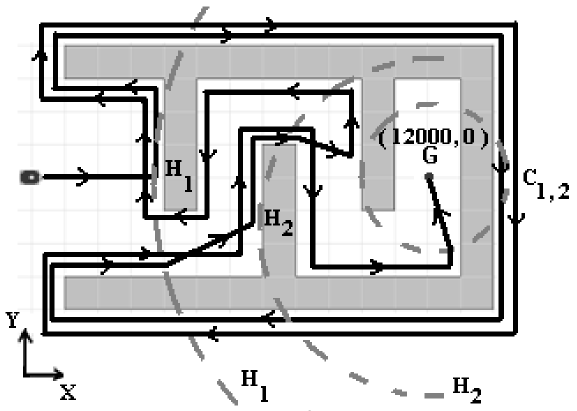

Figure 2. Class1's path for the environment2

2.3. Alg1 and alg2 algorithms

Alg1 and Alg2 by Sankaranarayanan and Vidyasagar are revised versions of Bug2 and Class1 respectively. Both of them intend to shorten the path travelled by the mobile robot. The basic idea is that if the mobile robot encounters a previously experienced hit point, $\mathbf{H_k}$, or leave point, $\mathbf{L_k}$, during its boundary-following action, it compares the lengths of the path which are from $\mathbf{H_k}$ or $\mathbf{L_k}$ to the last hit point, $\mathbf{H_i}$, or the last leave point, $\mathbf{L_i}$, in clockwise sense and counter clockwise sense **(k<i)**. According to the minimum path length to $\mathbf{H_i}$ or $\mathbf{L_i}$, the boundary-following direction is determined and the robot goes to $\mathbf{H_i}$ or $\mathbf{L_i}$ without any recording data of the path length. The record of the path length is only restarted after reaching $\mathbf{H_i}$ or $\mathbf{L_i}$. Then, the robot explores new external lines of the obstacle and maintains its boundary-following action until it finds a new leave point to **G**. Based on above property, Alg1 and Alg2 differ from Bug2 and Class1 respectively. If it faces a different experienced point again, the same procedure is applied [7, 8]. Figure 3 represents the paths generated by Alg1 and Alg2.

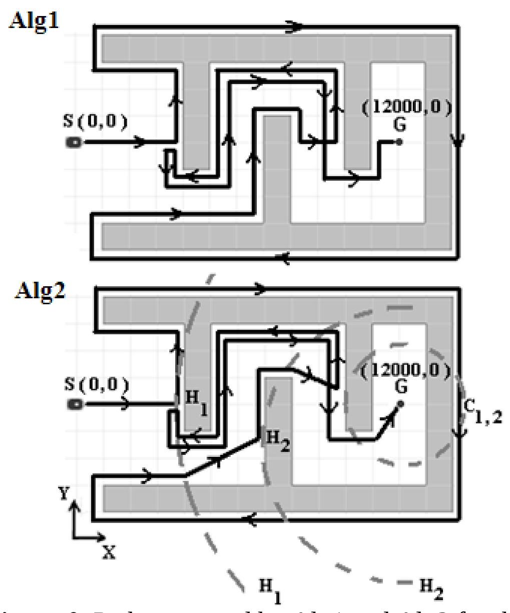

Figure 3. Path generated by Alg1 and Alg2 for the environment2 respectively

2.4. DistBug algorithm

Some Bug's algorithms such as DistBug algorithm by Kamon and Rivlin originate from Alg1 and Alg2. They use different data structures to store the hit and the leave points together with some useful information about the followed path [11].

In this algorithm, the mobile robot has a maximal detection whose range is **R,** and it has two basic behaviors which are the boundary-following and the motion-to-goal actions. At the beginning, the robot moves through **G** until it encounters an obstacle. Then, its boundary-following action is activated in clockwise sense (default). During the boundary following, the robot records the minimal distance, $\mathbf{d_{min}(G)}$, to **G** achieved since the last hit point. The robot also senses the distance in free space, **F,** which is a distance of an obstacle from the robot's location**, X,** in the direction of **G**. If no obstacle is sensed, **F** is set to **R**. The robot leaves the obstacle boundary as shown in figure 4 only when the path to **G** is clear or the equation, $\mathbf{d(X, G) - F \leq d_{min}(G) - Step}$, is satisfied where **d(X, G)** is the distance from **X** to **G,** and **Step** is a predefined constant [6].

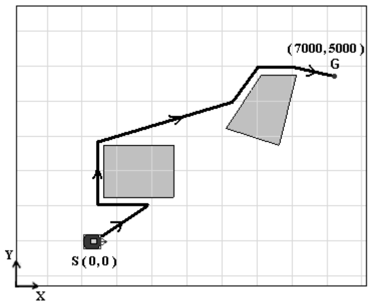

Figure 4. DistBug's path for the environment1

3. Implementation

Generally the Bug's algorithms are published as pseudo code [4]. We have written them as **C++** code executed in Linux platform using **ARIA** library [14] and used **MobileSim** simulator [14] which is compatible with real **Pioneer** mobile robots. Two different indoor-environments, as well as environment1 and environment2, are used for each of the Bug's algorithm as indicated in figure 5, and their maps are created by **Mapper3** [14]. The environment1 is limited in 10m length and 8m width, and it includes two convex-shaped obstacles. On the other environment, the environment2 has a mazelike obstacle whose outer dimensions are 13m length and 8m width. In experiments, $\mathbf{S_1}$ and $\mathbf{S_2}$ are locally **(0mm, 0mm)** for both environments and $\mathbf{G_1}$ is locally **(7000mm, 5000mm)** for environment1 and $\mathbf{G_2}$ is locally **(12000mm, 0mm)** for environment2.

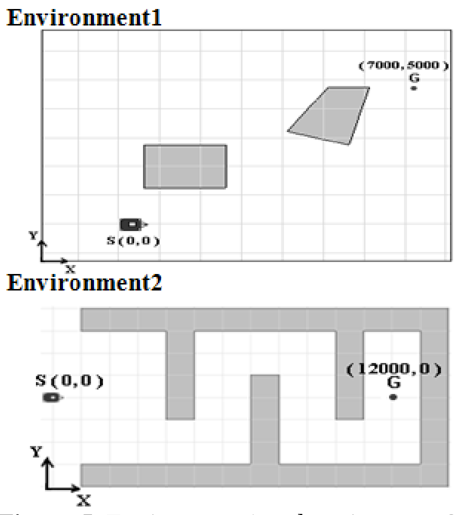

Figure 5. Environment1 and environment2

During the implementation, the mobile robot updates its position and navigation data, uses its sonar sensors, moves towards the goal, and by the wall following or the boundary following, follows the edges of the unknown obstacle.

In the simulation environment, the robot uses the odometry to localize itself and to update its position data. It's important that the mobile robot refreshes its localization frequently in order not to pass over the hit and the leave points and not to skip the slope discussed in Bug2. Note that the localization of the robot is perfect in the simulation whereas in the real world it is difficult to become perfect.

Some Bug's algorithms, as well as TangentBug [12] and DistBug, need range sensors whereas Bug1 and Bug2 do not [4]. Bug1 and Bug2 require tactile sensors, but in this study the sonar sensors instead of tactile sensors are used to sense the environment and to manage the operations such as wall following and detecting the unknown obstacles.

The most important task for the mobile robot is reaching **G**, therefore, it tries to move towards **G** until it replans its motion according to the algorithm.

The obstacle detection and the wall following are implemented by using sonar sensors whose layout is denoted in figure 6. The robot uses front sensors numbered as **1, 2, 3, 4, 5** and **6** to sense obstacles that block the path, and uses sidelong sensors numbered as **7, 8** for the wall following in clockwise sense and **0, 15** in counter-clockwise sense.

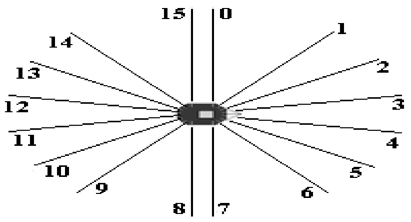

Figure 6. Sonar layout

4. Experiments and results

Twelve applications are carried out to compare the performance of each previously described the Bug's algorithms. In these applications, the environments given in fig.5 are used. The experiment is simulated on a **Pioneer** mobile robot and managed by **MobileSim** simulator. An open source code named as **ARIA** is used to program the mobile robot.

The mobile robot in the 2D surface has a local **S**, and **G** for each tested environments. Based on the bird's-eye view, the shortest distance between **S** and **G** is ***8602.325267mm*** for environment1 and ***12000mm*** for environment2. In this study, Bug1, Bug2, Class1, Alg1, Alg2 and DistBug are evaluated.

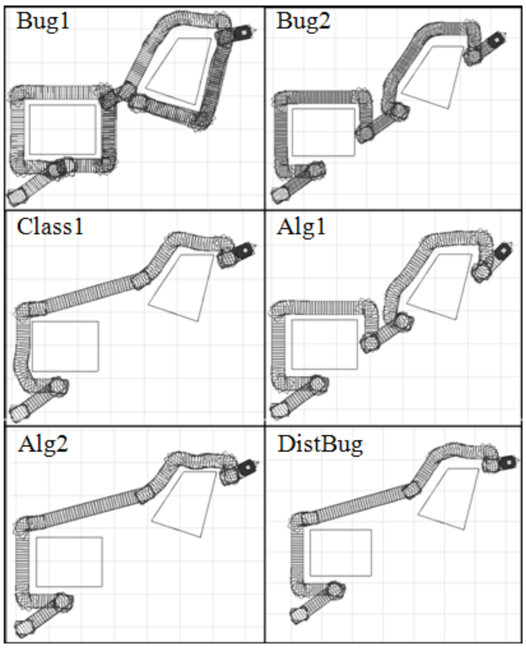

Figure 7. The paths of Bug1, Bug2, Class1, Alg1, Alg2 and DistBug for environment1 respectively

In the first step of the experiment, each described algorithms are tested in environment1 as shown in figure 7. Bug1 is the slowest one among the tested the Bug's algorithms but it guarantees the mobile robot to reach the reachable target. The performance of Bug2 is better than the performance of Bug1 for the environment1 because of its greediness. Alg1's path length is nearly as same as Bug2's because Alg1 behaves like Bug2 at normal conditions. On the other hand, Class1 does not only exhibit a great performance because of constraining its searching area successfully but Alg2 and DistBug are also great. As indicated in table 1 and shown in figure 9, three best for environment1 are Class1, Alg2, and DistBug, and the worst is Bug1.

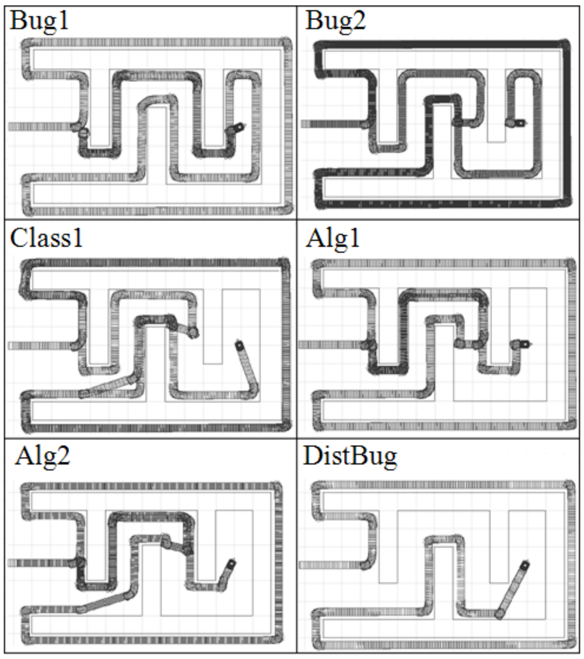

Figure 8. The paths of Bug1, Bug2, Class1, Alg1, Alg2 and DistBug for environment2 respectively

In the second step of the experiment, each described algorithms are tested in environment2 as shown in figure 8. It's obvious that Bug1 beats Bug2 which is the worst one in environment2. Because of some cases, for instance, a maze searching; Bug2 loses its advantage upon Bug1. At that point, Alg2 exhibits its feature upon Bug2 which improves the path length of Bug2 according to experienced hit or leave points. Class1 couldn't maintain its great performance of environment1 in the environment2 because of not being able to change its boundary-following direction. It is one of the disadvantages for Class1. At this time, Alg1 exhibits its property upon Class1 which improves the path length of Class1. In contrast, as indicated in table 1 and shown in figure 9, DistBug exhibits a great performance among others because it provides the mobile robot to use its sensors efficiently, so it is able to find the shortest path whereas others do not.

The experimental data is given in table 1, and is also charted in figure 9.

Table 1. Path lengths travelled by the robot for environment1 (env.1) and environment2 (env.2)

	Env.1(mm)	**Env.2 (mm)**
Bird's-eye view	*8602.325267*	*12000.000000*
Algorithms		
Bug1	*29021.921497*	*113076.893602*
Bug2	*16193.656123*	*150288.508183*
Class1	**12368.997925**	*141355.555627*
Alg1	*16378.524969*	*93916.116569*
Alg2	*12790.592228*	*91713.843486*
DistBug	*12794.794053*	**69485.595117**

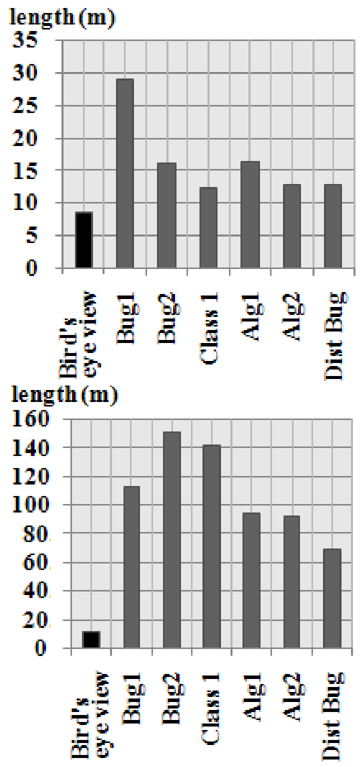

Figure 9. The charts of path lengths for environment1 and environment2 respectively

5. Conclusion and proposals

In this study, based on empirical data, a performance comparison among Bug1, Bug2, Class1, Alg1, Alg2 and DistBug algorithms is realized. The application is carried out on a Pioneer robot providing the simulation environment by MobileSim. In order to compare the algorithms, two environments are used. The results show that Bug1 is able to beat Bug2 for some cases. For the simplex environment, Alg1 and Class1 compete with DistBug, but for the little complex environment, DistBug exhibits a great performance, and Alg1 and Alg2 exhibit their feature upon Class1 and Bug2.

The future work will include different environments to test the performances of the Bug's algorithms and it will contain at least three of them which are Rev1 [10], Rev2 [10], VisBug-21 [2], VisBug-22 [2], HD-I [11] and TangentBug [12] in order to extend the study. After implementing these, the algorithms should be tested on real pioneer robots to see the effects of the real world.

6. References

[1] Maria Isabel Ribeiro, "Obstacle Avoidance", 2005.

[2] V. Lumelsky and T. Skewis, "Incorporating range sensing in the robot navigation function", *IEEE Transactions on Systems Man and Cybernetics*, vol. 20, 1990, pp. 1058 – 1068.

[3] V. Lumelsky and A. Stepanov, "Path-planning strategies for a point mobile automaton amidst unknown obstacles of arbitrary shape", *in Autonomous Robots Vehicles*, Springer, I.J. Cox, G.T. Wilfong (Eds), New York, 1990, pp. 1058 – 1068.

[4] James Ng and Thomas Bräunl, "Performance Comparison of Bug Navigation Algorithms", *Journal of Intelligent and Robotic Systems*, Springer Netherlands, Volume 50 Number 1, 2007, pp. 73-84.

[5] V.J. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment", *IEEE Trans. Automat. Contr. 31.*, 1986, pp. 1058–1063.

[6] Kamon, I. and Rivlin, E., "Sensory-based motion planning with global proof", *IEEE Trans. Robot. Autom.13.*, 1997, pp. 814–822.

[7] Sankaranarayanan, A. and Vidyasagar M., "Path planning for moving a point object amidst unknown obstacles in a plane: a new algorithm and a general theory for algorithm development", *Proc. of the IEEE Int. Conf. on Decision and Control 2*, 1990, pp. 1111–1119.

[8] Sankaranarayanan, A. and Vidyasagar, M., "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane", *Proc. of the IEEE Int. Conf. Robot. Autom. 3*, 1990, pp. 1930–1936.

[9] Noborio, H., "A path-planning algorithm for generation of an intuitively reasonable path in an uncertain 2-D workspace", *Proc. of the Japan–USA Symposium on Flex. Autom. 2,*, 1990, pp. 477–480.

[10] Noborio, H., Maeda, Y. and Urakawa, K., "A comparative study of sensor-based path-planning algorithms in an unknown maze", *In Proc. of the IEEE/RSI Int. Conf. on Intelligent Robots and Sys. 2*, 2000, 909–916.

[11] Evgeni Magid and Ehud Rivlin, "CAUTIOUSBUG: A Competitive Algorithm for Sensory-Based Robot Navigation", *Proceedings of 2004 IEEE/RSJ international Conference on Intelligent Robots and Systems*, Sendal - Japan, September 28 – October 2, 2004, pp. 2757-2762.

[12] CHOSET, Howie, LYNCH, Kevin M. and HUTCHINSON, Seth, "Bug Algorithms", *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge Center, 2005.

[13] Hiroshi Noborio, "Several Path-Planning Algorithms of a Mobile Robot for an Uncertain Workspace and their Evaluation", *Proc. of the IEEE Intelligent Motion Control* , 1990, pp. 289-294.

[14] Mobile Robots Inc, ActivMedia Robotics, LLC, 2008, http://www.mobilerobots.com/ .