FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA XIX.

Kolozsvár, 2014. március 20–21.

KÜLÖNBÖZŐ PÁLYATERVEZŐ ALGORITMUSOK MOBILROBOTOK SZÁMÁRA MATLAB KÖRNYEZETBEN

DIFFERENT PATH PLANNING MODELS FOR A POINT-REPRESENTED MOBILE-ROBOT IN MATLAB ENVIRONMENT

CORTES Tarcisio Ribeiro de Oliveira

*Óbuda University – Bánki Donát Faculty of Mechanical and Safety Engineering; 1053
Hungary, Budapest, Kossuth Lajos street 4, 3/1/a; Telephone: +36-30-8261249;
tarcisiocortes@gmail.com*

Abstract

This paper is part of a project work which discusses the importance of path planning for mobile robots, comparing different models regarding their basic principles of operation. It explores different variations of bug algorithms, comparing their respective benefits and disadvantages in order to finally allow the simulation of two of them in MATLAB environment. The simulation considers a point-represented mobile robot, and may be used for educational purposes in the field of mobile robotics. There is an interface which allows the user to choose between the implemented algorithms, as well as setting the starting and goal points and inserting obstacles for the simulation.

Keywords: path planning, mobile robots, bug algorithms.

Összefoglaló

Jelen cikk a projekt munkám egy része, amely különböző pályatervező algoritmusok ténykedését hasonlítja össze a működésük szempontjából. Részletesen tárgyalom a BUG algoritmus-család felosztását és működését, majd elkészítettem egy szimulációs modellt, MATLAB környezetben, amely a különböző elven működő BUG algoritmusokat hasonlítja össze oly módon, hogy ugyanazon környezetben működteti őket. Eredményként közlöm a különböző algoritmusok végrehajtási idejét.

Kulcsszavak: útvonal tervezés, mobil robotok, BUG algoritmusok.

1. Introduction

Path planning is one of the most important processes in the field of mobile robotics. It consists in determining a path that a robot must take in order to achieve a given destination. Usually, the path planning problem consists in finding paths between different locations in an environment which contains obstacles that must be avoided.

Mobile robots need low-level descriptions of how to move, which are defined by algorithms that convert the high-level instructions of tasks set by humans. Besides the path planning, it is also important to consider the orientation of the robot. Although it is sometimes represented as a point in the space (which is the case for the simulation tool developed for this project work).

2. The task

The given task consists of developing a software to simulate the motion planning of a mobile robot. The software can serve as an educational tool for understanding the implemented algorithms. In the simulation, the robot, placed in a given space, is supposed to reach a determined goal, moving freely through a continuous workspace and avoiding possible obstacles. Two variations of bug algorithms were implemented. There is an user interface which allows the user to choose between the desired algorithm and select the starting and goal points, as well as to insert obstacles in the workspace.

3. Algorithms and Models

There is a large number of algorithms and models regarding the motion planning of mobile robots. Choosing the most appropriate algorithm for a given application relies on answering important questions. For instance, defining the type of the sensors that will be used depends on the environment in which the robot is supposed to work [1].

Another aspect that must be considered when choosing an algorithm is whether the planning must be static (the robot is the only moving object) or dynamic (there are dynamic objects such as moving obstacles). There are also distinctions between global and local path planning. In the first case, there is already known information about the environment. In the latter, the robot has to acquire such information from sensors before deciding its next move.

The proposed simulation tool approaches the problem of path planning by using bug algorithms, which do not require prior information about the environment, as the robot obtains it from its sensors. There is the assumption that the goal position is always known. Some variations assume only tactile sensing, while others rely on a non-zero depth sensing.

4. Bug Algorithms

Bug algorithms, as stated before, assume only local knowledge of the environment. It guarantees that, as long as the goal position is accessible, it will be reached [2].

For the aforementioned project work, the workspace was supposed to be a 2D environment with a finite number of obstacles. It might represent a laboratory room, for example, and in fact this was the objective. The sensing model demands two devices: a goal sensor, which indicates the global position of the goal by its distance and direction, and a local visibility sensor, whose depth of sight depends on the adopted variation of the bug algorithms.

4.1. BUG1 Algorithm

BUG1 is one of the simplest strategies from the BUG family. It follows three basic steps:

- Move toward the goal in a straight line (which is called the *m-line*);
- If an obstacle is found, turn left and follow its entire perimeter;
- After following the entire perimeter of the obstacle, go back to the closest point to the goal and continue.

This strategy may lead to unnecessary work. In fact, the robot may travel the boundary of an obstacle up to $\frac{3}{2}$ times [3].

4.2. BUG2 Algorithm

The BUG2 algorithm presents an improvement in comparison to the previous strategy discussed. Instead of circumnavigating the whole object, the robot always attempts to move along the *m-line* (the straight line that connects the starting and goal positions).

Considering this, it is possible to say that the algorithm follows three basic steps:

- Move toward the goal on the *m-line*;
- If an obstacle is encountered, turn left and follow its perimeter until reaching the *m-line* again. If the goal is reached, then stop;
- After reaching the *m-line*, start moving toward the goal again.

However, following only these steps, the robot might reach an infinite loop. In order to avoid this problem, it is necessary for the robot to remember the distance to goal from the point from where it left the boundary the last time. This way, it will only leave the boundary again in case that new point is closer to the goal than the previous one.

BUG1 algorithm examines all possibilities before making a decision, while BUG2 makes the choice which seems to be better, but without analyzing all possible scenarios. Usually, BUG2 has a better performance than BUG1, but the latter is more predictable.

4.3. Tangent BUG Algorithm

The algorithms shown previously consider solely tactile sensing. In reality, however, there are range sensing devices which make the mobile robots capable of looking ahead. Such devices are fundamental for the functioning of Tangent BUG algorithms.

- Move toward the goal until the robot is trapped in a local minimum. If the goal is reached, then stop;
- Move along the boundary of the obstacle, remembering the minimal distance from the goal and the shortest path to the goal from the current position. In case the latter distance is shorter than the first one, stop following the boundary and go toward the goal (back to step 'i').

Tangent BUG offers a much more realistic approach, and presents a considerable improvement in comparison to the strategies shown previously. The images below show the performance of BUG2 and Tangent BUG for a given obstacle:

Fig.1. BUG2 performance

Fig.2. Tangent BUG performance

One important aspect to consider when working with Tangent BUG algorithm is the range of the sensor. It is common to assume an infinite range sensor in order to make a problem simpler; however, this is not possible to execute in real world. Usually, the robots have finite range sensing devices. The point-representend mobile robot moves along the drawn route. The radius shown represents the range of the sensor, which is defined in the source code.

5. Developing the Software

BUG2 and Tangent BUG algorithms were implemented in the simulation software. A graphical interface was developed through MATLAB GUIDE. It includes an axis area in which the user places obstacles, start and goal points. It is also possible to select between the two algorithms. The route is calculated and drawn in the window when the user clicks on the correspondent button. There is another button for starting the animated simulation.

Fig3. Screenshot during animated simulation

6. Conclusion

This paper discusses path planning for mobile robots, focusing on three different algorithms from the bug family. In most cases, BUG2 algorithm presents significant improvements in relation to BUG1, while Tangent BUG outperforms both of the previous variations shown. BUG2 and Tangent BUG were implemented in a simulation software. Considering the environment shown in figures 1 and 2, the following results were obtained:

Table 1. Comparison between BUG2 and Tangent BUG algorithms in a given environment

	Total distance traveled by the robot (cm)
BUG2	14.10
Tangent BUG	8.96

This software allows a user to observe and compare the performance of BUG2 and Tangent BUG algorithms, in an environment defined by the user by choosing obstacles, start and goal points. It may be used for educational purposes by students who are interested in mobile robotics.

Acknowledgement

The author expresses his gratitude for CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the scholarship granted under process number 88888.010526/2013-00. The project was also assisted by the European Union, with the co-financing of the European Social Fund, TÁMOP-4.1.1.C-12/1/KONV-2012-0002.

References

- [1] Buniyamin N., Wan Ngah W.A.J., Sariff N., Mohamad Z.: *A Simple Local Path Planning Algorithm for Autonomous Mobile Robots*, International Journal of Systems Applications, Engineering & Development, Issue 2, Volume 5 (2011), p.1-3
- [2] I. Nagy: *Path Planning Algorithm, Based on User Defined Maximal Localization Error*, Periodica Polytechnica, 2005 49/1-2, ISSN: 0324-6000, Hungary (2005), p.43-57
- [3] S.M. Lavalle: *Planning Algorithms*, Cambridge University Press, University of Illinois (2006), p.667-669