

Autonomous Motion Planning on Mars *

Wenbo He Chengdu Huang Xue Liu Liqian Luo Min Wu

Department of Computer Science
University of Illinois at Urbana-Champaign
Thomas M. Siebel Center for Computer Science, 201 N Goodwin Ave,
Urbana, IL 61801, USA

Abstract

To confirm whether there is water on Mars, we send mobile robots roving on Mars, and let them go to the target area to do verification. On the surface of Mars, there are Mountains (obstacles) and holes. Due to the protection of bumpers, when a robot hits an obstacle, it doesn't hurt. Only when a robot hits an obstacle (like a mountain), it can detect the obstacle. But if a robot gets into a hole, it cannot manage to get out of the hole, so the robot dies when running into a hole. The robots are able to detect holes by ultrasonic sensors. However, the sensors can only detect holes within 75mm. Our objective is to design an algorithm to autonomously navigate a mobile robot to the target without losing the robot, and make it approach to target as quickly as possible.

We propose an algorithm, called "Evade-Along-Edge". Without the detection of holes or obstacles, a robot always moves to the target directly. When a robot detects a hole or an obstacle, it turns to walk along the edge of the hole or obstacle. When it cannot detect the hole or obstacle any more, it moves to the target again. We design several tactics to improve the performance of the algorithm and manage the false positive and false negative of the ultrasonic sensors. We implemented our algorithm and tested it in "The maRTian

Task" simulation environment.

1 Introduction

After recent new discoveries on Mars, scientists have found evidence that large amounts of water-ice hide within massive sand dunes on Mars (from BBC News). It is important to confirm whether there really exist water, which may help to infer whether there were or are live forms on Mars, and whether humans can survive on Mars. In order to confirm whether there is water on Mars, mobile robots are sent to test the suspected water area. The robots are carried by orbiting spacecraft, which is able to identify the position, orientation of robots and the target location (suspected water). However, the robots are almost "blind". The robots are equipped with ultrasonic sensors, which locate holes in the Mars surface in the neighboring area. Also, robots are equipped with 12 bumpers, which identify the relative direction of obstacles a robot has bumped into. Our objective is to let the mobile robot to get to the target autonomously as soon as possible.

The robot moves 200mm per second and can sense holes in the ground 75mm away. Hence the worst case response should not exceed 375ms. The communication round-trip is 50ms. The robot understands simple commands, such as forward, backward, turn, and

*All the authors contribute equally to the design.

querying its sensors. The robot is designed to cooperate with each other, so it can obtain information from other robots.

Although there are several books (e.g. [1]) or papers (e.g. [2]) address the motion planning problem, none of them has the same assumption about the robot as in this paper. We propose an algorithm, called “Evade-Along-Edge” in this paper.

With different assumptions, the implementable algorithms are totally different. The same strategy may yield different performance too. Consider the case that we know the global information, such as the map of holes and obstacles. Although this is not a realistic assumption, we’d like to show the path finding strategy in this case first. As shown in Figure 1, the property of the “shortest path” is that it passes the corners of holes or obstacles.

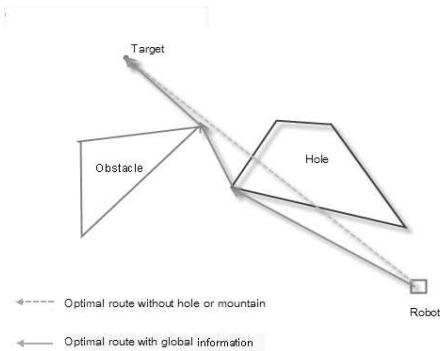


Figure 1: Optimal path for water searching robot

2 Constraints and Strategy

The primary rule of scheduling in this project is to prevent from losing the robot, and then compute the path. To avoid losing the robot, we need to query sonar sensor at least once in every $325ms$ ($375ms - 50ms$). Each robot is equipped with 12 ultrasonic sensors. They are unreliable sometimes. False negative (sensor suggests no hole, when there is one) rate of

each sensor is $\frac{1}{1000}$. False positive (sensor suggests a hole, where there is none) rate is $\frac{1}{50}$. In our design, the robot gets the sensor state continuously. If a sensor issues alarm two out of three successive times, then we think the robot is approaching to a hole.

When robots are on a mission, it roves within a playground. If a robot is out of the boundary of playground, we will lose it. The radius of a robot is $75mm$. If the position of a robot is (x, y) , with the unit of mm and the width and length of the playground are W and L respectively, we should make $0 \leq x - 75$; $x + 75 \leq W$; $0 \leq y - 75$; $y + 75 \leq L$. Therefore, the robot is always walking inside the playground.

In order to make the robot reach the target in the shortest time, shortest path is desired. However, due to the lack of global information, shortest path is impossible. We need to design path searching algorithms to make the robot travel as short as possible. To accelerate the motion of a robot, a robot should move to the target directly if without presence of holes or obstacles; otherwise, it attempts to walk along the edge of the hole or obstacle. So we come up with the idea of “Evade-Along-Edge” algorithm.

3 Motion Planning Algorithms

3.1 Basic Idea

In more realistic cases, when global information is not available, we desire to find a path as short as possible in practice. In this case, holes and obstacles can be detected only when they are in vicinity of robots. Figure 2 illustrates the idea of our algorithm. In the simple case given in Figure 2, we do not need to worry about how to avoid hitting boundaries of the playground, how to avoid dead loop of the searching path, and so on. Our strategy is that when the robot doesn’t detect a hole or

obstacle, it always moves to the target directly; when the robot detects a hole or an obstacle, it turns to walk along the edge of the hole or obstacle.

This idea is feasible. Since there are 12 bumpers and 12 sonar sensors, by checking which bumper is bumped, we are able to know the angle between the moving direction of robot and the boundary of the obstacle. Therefore, we can turn an angle to make the robot turns a certain angle and move roughly in parallel with the boundary of the obstacle. Similarly, we can get the angle between the boundary of the hole and the moving direction of the robot. And the robot can move roughly in parallel with the boundary of the hole.

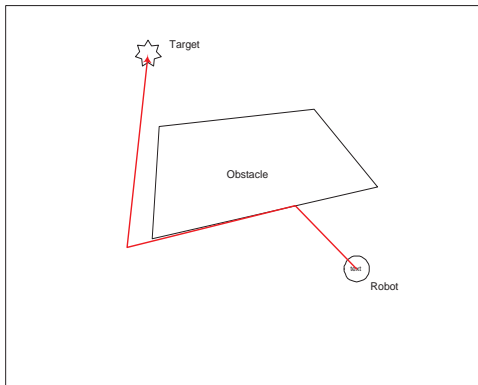


Figure 2: Evade-Along-Edge Algorithm

3.2 Algorithm of Evade-Along-Edge

The robot is modeled as a small circle in a 2D playground. From 3.1, a robot has two modes to ensure that a robot will finally get to the target: *forward-to-target* and *boundary-following*.

In *forward-to-target* mode, we need to get position of target and navigate the robot to the target directly until the robot hits the target. If the robot gets to the target then the algorithm terminates. In the meanwhile, we continuously check the sonar sensors and bumpers to judge if the robot is close to a hole or hit an

obstacle. If so, the robot turns an angle to be in parallel with the boundary of the encountered hole or obstacle, and switches the mode to *boundary-following*.

In *boundary-following* mode, the robot goes $150mm$ at each step. Then the robot turns $\frac{\pi}{2}$ heading to the boundary direction. If after $100mm$, the robot still does not hit the boundary, then it switches to *forward-to-target* mode; otherwise, the robot remains in *boundary-following* mode and turns $\frac{\pi}{2}$ back, thus the robot can head to the direction which parallels to the boundary and continues the operation in *boundary-following* mode.

Algorithm: Evade-Along-Edge

Step 1: The robot in *forward-to-target* mode, navigate the robot to target until algorithm terminates or the robot switches to *boundary-following* mode. If it switches to *boundary-following* mode, goto Step 2; otherwise, goto Step 1.

Step 2: The robot operates in *boundary-following* mode, until it switches to *forward-to-target* mode. If it switches to *forward-to-target* mode, goto Step 1; otherwise, goto Step 2.

3.3 Implementation

There are several issues to be addressed in the implementation of Evade-Along-Edge algorithm.

First, how to turn a robot to move in parallel with the boundary of a hole or an obstacle? For each robot, there are 12 sonar sensors and 12 bumpers on board. We notice that the head of the robot is roughly between sensor number 2 and sensor number 3. If sensor number 2 detects a hole, the angle between boundary of the hole and the moving direction is roughly $\frac{\pi}{12}$. If sensor number 1 detects a hole, the angle

is $\frac{\pi}{12} + \frac{\pi}{6}$. It is not necessary to get the precise angle between boundary of the hole and the moving direction in the first time. However, we need to check the status of sensors continuously, and adjust the moving direction to make the robot heading in parallel with the boundary of the hole. Similarly, when a robot hits an obstacle, we can estimate the angle and turns the robot to moving in parallel with the obstacle boundary.

Second, which way to turn, clockwise or counterclockwise?

By evaluating many possible cases, we conclude that when a robot encounters a hole or an obstacle, it doesn't really matter to turn clockwise or counterclockwise, since the robot are almost blind. We can select either direction to turn the robot. However, always turn the robot in one direction may not yield good performance. Our strategy is to turn the robot and make the robot forwarding to a certain direction after the turn. And keep turning the robot in that direction until the robot reaches a boundary of the playground.

Fourth, how to speed up the robot?

Since our goal is to reach the target as soon as possible, we need to consider how to speed up the motion of robot. We have several policies to improve the performance, such as move the robot directly to target if no obstacle or hole is detected; when we turn the robot in a certain direction, we keep turning the robot that way; when the robot is in boundary-following mode, the distance between robot and the boundary should be very small.

4 Acknowledgement

We, as graduate students in UIUC, appreciate the support from our academic advisors Professor Tarek Abdelzaher, Professor Klara Nahrstedt and Professor Lui Sha. We thank Professor Steve LaValle, who suggested some reading materials for us. We'd also like to thank Qixin Wang and Hui Ding, who pro-

vided lots of feedbacks in our original algorithm design and implementation.

References

- [1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, to be published in 2006.
- [2] S. L. Laubach and J. W. Burdick, "Practical autonomous path planner for turn-of-the-century planetary microrovers," in *Proc. SPIE Vol. 3525, p. 182-191, Mobile Robots XIII and Intelligent Transportation Systems, Howie M. Choset; Douglas W. Gage; Pushkin Kachroo; Mikhail A. Kourjanski; Marten J. de Vries; Eds.*, Jan. 1999, pp. 182–191.