

RoboCoog: The CiberMouse Agent

Ashish Kapadia, Arun Chhetri, Ronak Shah, and Dr. Albert M. K. Cheng (Advisor)
ashishbk@gmail.com, achhetri@uh.edu, rshah10@uh.edu, cheng@cs.uh.edu

University of Houston – Department of Computer Science

Abstract: *RoboCoog is the name of the robot from University of Houston that will participate in the RTSS 2007 CiberMouse competition. The competition will be held in discrete time driven simulation environment where the responsibility of each robot is to find a beacon in an unknown environment and come back to the starting position of the robot before the other robots. At the same time, the robot should try to avoid collisions with other robots and obstacles in order to avoid penalties. In this paper we describe our approach to solve various challenges posed by this competition.*

1. Introduction

The main challenge in the CiberMouse competition is to use various sensors available to the robot to learn the unknown environment that it needs to explore in order to reach the goal. The noise in the values provided by the sensors makes the problem particularly difficult and closer to real-world problem faced by the actual robot. The robot is given few actuators via which it interacts with the environment.

We have identified various sub problems that we will face in order to build the robot. We are listing our approach to solve these problems in the subsequent sections. Section 2 describes the approach we are taking to build the map in order to remember the unknown environment that we explore. Section 3 describes various states we've identified

the robot will be in as it attempts to fulfill its goal. Section 4 describes utilizing the scarce sensor resources in order to efficiently manage them. Section 5 will describe our strategy on avoiding the obstacles. Section 6 will describe our approach on locating beacon. Section 7 will describe return strategy employed by the robot to come back to starting position.

2. Map Building

In order to remember various places that robot has visited it is essential that robot builds a map so that it can avoid already visited area in the map. In addition to this, map is essential for the competition since the robot needs to come back to the position from which it started.

Since the size of the arena is known to be 28 unit length wide and 14 unit length long, we can represent the map in a two dimensional array. Various experiments on the simulation environment have provided us data on how many unit length our robot will move when we apply specific power to the motors. Based on this, we've determined to use 0.1 unit length for each cell in the array. In addition to this, since we do not know the actual position of the robot in the map during the start of the competition, we've decided to double the original width and height of the map to be 56 unit length and 28 unit length respectively. Therefore, our map

will be two dimensional array with 560 width and 280 length. The figure 2.1 shows how this configuration takes care of possible extreme cases in which the robot may be placed in the map.

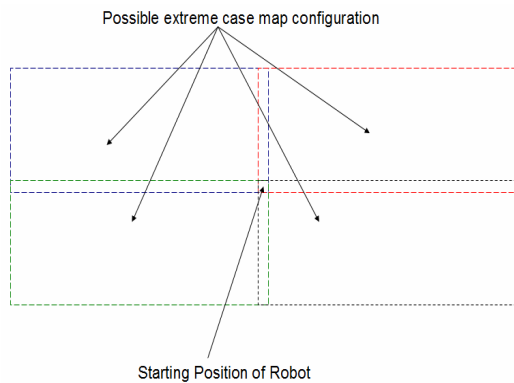


Figure 2.1. Map configurations.

One simple way to represent the map is by marking the map with various states such as: UNKNOWN, VISITED, OBSTACLE, BEACON etc. Even though this approach may work, it does not take noise into account. In addition to this, since there are other robots that are moving in the maze we must also consider a possibility that we may incorrectly mark them as obstacle. A better approach to this problem is to use probability value (belief value) to mark cells as obstacle. On multiple readings by the robot, the robot can increase or decrease these cell values based on whether the cell has obstacle. This probabilistic approach of building the map would help when another robot is incorrectly marked as obstacle. It will also take multiple measurements of the cells into account.

The probabilistic map will help in dealing with noise in some aspect but it is not going to totally eliminate the major problem related to cumulative noise caused by noise values in the

motor. Since the noise is going to be in Gaussian distribution, one can use Kalman filter algorithm to reduce the localization error caused by noisy error. The Kalman filter works particularly well in dynamic system with noisy measurements.

3. Robot States

The following figure shows the states of the robot.

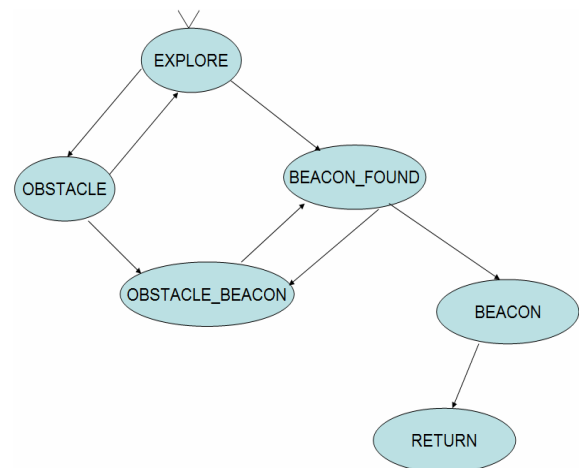


Figure 2.2. States of the robot.

EXPLORE State: This is the starting state of the robot when the robot has not sensed beacon or obstacle. The robot remains in this state until it senses either the beacon or an obstacle. In this state the robot tries to explore the environment and based on information stored in Map, it avoids visiting the same area twice.

OBSTACLE State: The robot comes in this state when it is in explore state and sense an obstacle. In this state the robot avoids the obstacle and then goes back to EXPLORE state to explore more

environment. When the robot senses the beacon in this state it will go to OBSTACLE_BEACON state.

OBSTACLE_BEACON: In this state the robot has the knowledge of the location of beacon but there is some obstacle in between the two. The main aim of robot in this state is to avoid the obstacle and move in such a way that it goes close to the beacon.

BEACON_FOUND: The robot comes in this state from either EXPLORE state or OBSTACLE_BEACON state. In this state the robot only senses the beacon and not obstacle.

BEACON state: This state indicates that the robot has reached the destination and it is on the beacon. This state can only be reached from BEACON_FOUND state.

RETURN state: The RETURN state indicates that the robot is returning back to starting position. Section 7 describes our strategy on returning back to starting position.

4. Sensor Scheduling

We have implemented different scheduling strategies depending upon the state the robot is in. The priorities of different sensors change as we move on from one state to another. For example in EXPLORE state the robot will initially use the beacon sensor in order to locate the beacon so that robot can immediately go into BEACON_FOUND state. We will schedule the beacon sensor after every 9 cycles with highest priority. In the mean time the robot will use center obstacle sensor and then alternatively left and right obstacle sensors to check whether there is any obstacle in the path. If there is an

obstacle then the robot will go into OBSTACLE state.

5. Obstacle Avoidance

The key decision to make when avoiding the obstacle is to decide which direction the robot should move. This decision is made based on taking several factors into consideration such as querying the map to determine the best next move or with the help of left and right center we may be able to detect a non-obstacle on either side.

6. Locating Beacon

Correctly estimating the position of the beacon is an important aspect of the competition. The location of the beacon can be estimated based on previous readings from various location of the robot. We are still exploring the best way to precisely estimate the location.

7. Return Strategy

As we were moving from start position to target place we were building map at each instance by considering all obstacles that we encountered in our path. By using this build map we know which is start position from where robot started the competition and using this details available in the map we can find the path by which robot can come back to start position in such manner that it does not need to visit entire area which robot visited while come to beacon.

9. Conclusion

The CiberMouse competition is an interesting challenge that exposes various problems that are faced by building an actual robot in the simulation environment. The sensor noise particularly makes the task difficult. We look forward on applying the strategies we have described in here in the actual competition.

References

1. Bjorn B. Brandenburg Hennadiy Leontyev. Ramses the Rats: Ciber Mouse Agent
http://www.ieeta.pt/~lau/web_ciberRTSS/docs/RamsesTheRat.pdf
2. Leonid Ryzhyk, David Snowdon, and Stefan Petters. Numbat: an entry to CiberMouse@RTSS2006.
http://www.ieeta.pt/~lau/web_ciberRTSS/docs/Numbat.pdf
3. Ciber-Mouse: Rules & Technical Specification.
http://www.ieeta.pt/~lau/web_ciberRTSS07/docs/ciberRTSS07_rules.pdf