

# SWIFT – A Scenario based dynamic sensor scheduling robot

Jaya Shanmukha Srikanth Palli, Krishna Konda

Department of Computer Science, Department of Electrical & Computer Engineering

Texas A&M University

College Station, TX 77840

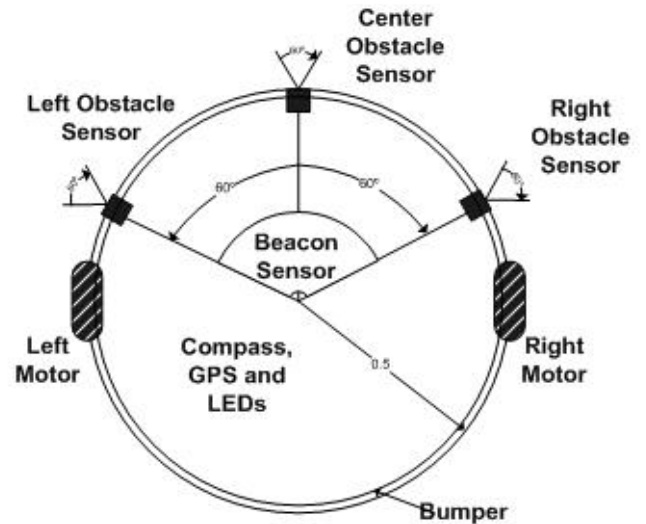
pjssrikanth@gmail.com, kkchaitu@yahoo.com

**Abstract** - In this paper, we discuss about the issues faced and the approaches we used to develop the virtual robot – SWIFT, a scenario based dynamic sensor scheduling robot. Our robot is based on the following strategies: Wall Following Approach, Virtual coordination system, Beacon following, Coordinate Triangulation to predict the beacon location and following the Slope for backtracking.

## I. INTRODUCTION

CiberMouse 2007 is a virtual robotics competition held as a part of the 28<sup>th</sup> IEEE Real-Time Systems Symposium. The virtual environment is provided by a simulation system which creates a virtual maze with a target area along with a beacon to guide the robot agent to the target area impeded by obstacles along the way. The goal of the robotic agent is to navigate around the maze and reach the target area and return to the return as close as possible to the starting position in the maze. It must achieve this by avoiding collisions with obstacles along the way and other robotic agents in as small duration as possible.

The simulation system consists of the simulator engine, the visualizer and the robotic agent. The simulator manages the state of the simulated world by moving the robotic agents according to their motor commands and providing sensory inputs to them according to their position in the maze. The simulation system is discrete and time-driven. During each time step the simulator sends sensor measurements to agents, receives actuations orders, applies them, and updates scores. The visualizer is responsible for graphical display of the robots in the virtual maze, including their states and scores.



**Figure 1. Diagrammatic Representation of the Virtual Robot**

The robot is equipped with obstacle sensors, beacon sensor, compass, collision sensor and a ground sensor. The sensor availability is limited to two sensors per cycle. The robot has two motors which drive its left wheel and right wheel respectively. Both translational and rotational movements are possible, depending on the power applied to each wheel. There are three LEDs, named Beacon or Visiting, Return and End, which are used to signal the three goals. The contest deliverable is the robotic agent which would be evaluated in various mazes to see how well it performs.

The rest of the paper is organized as follows. Section II describes the various challenges that need to be addressed while developing the navigation algorithm. Section III describes the different stages of the navigation strategy and explain how the robotic agent would perform under each stage.

## II. CHALLENGES TO BE ADDRESSED

This section describes the various challenges involved in the development of the navigation algorithm for the autonomous robot (mouse) in an unknown maze.

### A. Sensor scheduling

The robot can request only two sensors value per simulation cycle, out of the seven sensors present in the robot – this puts a greater importance on how sensors are scheduled in this constrained space. While designing our robot, we focused a lot on creating a scenario based schedule.

Depending on the circumstances of the simulation cycle, different sensors should be scheduled. For example, when very close to an obstacle, in order to avoid a collision, we should schedule in the IR (infrared) sensors in that particular direction. In case there is no obstacle detected, then we can schedule the beacon sensor and determine in which direction the destination lays. These are some of the issues to address when scheduling the sensors.

### B. Collision avoidance

Reliable obstacle avoidance is a vital feature, as penalties for colliding with static / moving objects accumulate during every round, therefore punishing the competing robot if it touches objects in the maze repeatedly. In case of a collision, the penalty is 5 points. Due to the high penalty, collisions with either the obstacles or other robots moving around the maze must be avoided or minimized as much as possible. This depends greatly on the obstacle sensor thresholds and beacon sensor values. Since the beacon sensor value is delayed by 9 cycles, this involves keeping track of the last 9 cycles and adjusting the movement in the direction of the beacon since the reading has been requested.

### C. Reaching destination

In order to reach the destination, wall following approach along with the beacon direction and the triangulation approach are used. This technique enables the robot to get around a wall and keep going towards the destination (the target indicated by the beacon). It follows the wall, until it comes around the wall and is able to navigate towards the target.

We have a few check-points in our wall following routine to ensure that the robot does not follow the same wall indefinitely. In order to do this, the robot periodically spins 360 degrees to look for the beacon. If the beacon was seen during this spin, it backtracks to the direction in which it saw the beacon. The robot then moves straight ahead in the direction of the beacon. If the robot has gone a full circle around the wall and still had not seen the beacon, we would break out of wall following and move in the unvisited direction. As the robot navigates through the maze, virtual coordinates are generated in the maze, which assists the robot eliminate navigations in the visited directions.

### D. Return Strategy

Returning as close as possible to the original position in the maze is a very important part of the competition. Since there is no beacon or any similar means to help us reach the destination, some form of virtual coordinates system should be used in order to reach the original position in the maze. Our robot can get the destination's direction, by computing the tangent of the line connecting the current position and destination. Even while returning, the robot takes care to avoid obstacles and incorporates wall following to get around a wall. The collision avoidance and wall following approaches previously discussed are reused in this algorithm.

### E. Noises

Each sensor has additive noise associated with it. This noise should be taken into account if nearly accurate results are to be obtained.

The following sensors have noises associated with them

1. Obstacle sensor (infrared sensors): Each obstacle sensor reading has a small amount of noise associated with it. Identifying the magnitude of the noise component for each sensor reading is important.
2. Motor noise: When power is applied to the wheels, there is a small noise associated with the power that is actually applied to the wheels. This will impact the localization information maintained and that information if not corrected for this noise, there is a chance the robot will not reach the originating position in the maze.
3. Compass noise: When using compass to navigate in the direction of the beacon or in the originating direction, compass noise if not corrected can lead the robot astray.
4. Beacon noise: Since the beacon helps the robot achieve the first goal, the noise must be smoothed out or else the robot may never reach the destination.

## III. NAVIGATION ALGORITHM IMPLEMENTED

The three goals are

1. Identify the destination and reach it
2. Identification of the completion of goal 1
3. Return to the originating position.

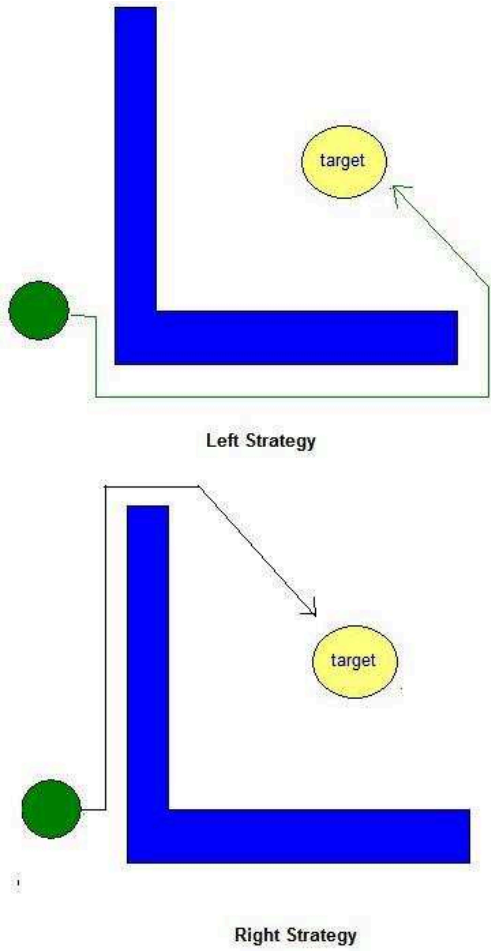
The first goal is achieved by use of obstacle avoidance, triangulation and wall following combined with reading the beacon sensor and modifying the direction of the robot to point in the direction of the beacon. This does not always lead to the desired results since there can be obstacles that can cause the robot to move in infinite cycles preventing the robot from reaching the destination. A cycle avoidance scheme is used to detect and break cycles so that the robot does not end up in infinite loops.

The second goal is achieved by reading the ground sensor and setting the appropriate LEDs after having reached the destination and the return journey should be marked by the robot.

The third goal is reaching the originating point in the maze. This is achieved by use of virtual coordinate system that will help position the robot in the maze and also identify the direction of movement of the robot in order to reach as close as possible to the originating position.

Collision avoidance and wall following are the techniques used in all the three goals. Obstacles can be identified by setting thresholds for the obstacle sensor readings and being able to identify when there is an obstacle directly in front of the robot, when the obstacle is far away but in the direction of the beacon and wall following need to be undertaken until the wall ends and we

can move in the direction of the beacon. Wall following is implemented by using both right strategy and left strategy as shown in figure 2.



**Figure 2. Left and Right wall following schemes**

The localization information that is to be used in the return journey is created during the first and second part of the goals and updated during the final goal. The localization information is calculated from the following equations

$$lOutPow_t = (lOutPow_{t-1} + lInPow_t) / 2$$

$$rOutPow_t = (rOutPow_{t-1} + rInPow_t) / 2$$

$$lNoisyOutPow_t = lOutPow_t * lNoise_t$$

$$rNoisyOutPow_t = rOutPow_t * rNoise_t$$

where

- ◆  $lInPow_t$  and  $rInPow_t$  are the power values supplied to the simulator at time instant  $t$

- ◆  $lOutPow_{t-1}$  and  $rOutPow_{t-1}$  are the power values produced by the motors at time instant  $t-1$  i.e. previous simulation cycle
- ◆  $lOutPow_t$  and  $rOutPow_t$  are the power values produced by the motors at time instant  $t$  i.e. current simulation cycle
- ◆  $lNoise_t$  and  $rNoise_t$  are randomly calculated motor noise
- ◆  $lNoisyOutPow_t$  and  $rNoisyOutPow_t$  are the power values to be applied to the wheels at time instant  $t$

These are the equations used by the simulator in calculating the power to be applied to the wheels based on the power supplied to the motors. From these equations, linear and rotational movements can be calculated by using

$$lin_t = (lNoisyOutPow_t + rNoisyOutPow_t) / 2$$

$$rot_t = (rNoisyOutPow_t - lNoisyOutPow_t) / diam$$

where

- ◆  $lin_t$  and  $rot_t$  are the linear and rotational components of the movement at time instant  $t$
- ◆  $diam$  is the diameter of the robot

Using the above equations, localization information can be calculated as follows

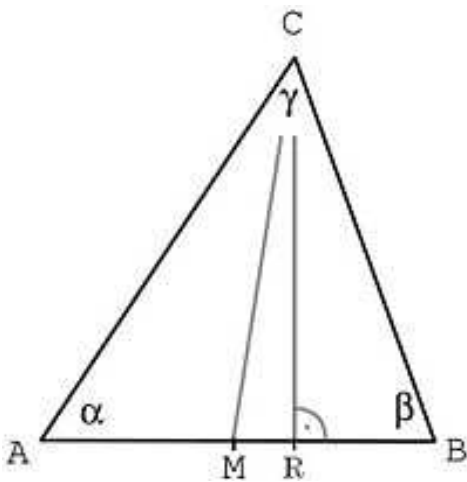
$$X = \sum (lin_t * \cos(rot_t))$$

$$Y = \sum (lin_t * \sin(rot_t))$$

The above equations will help keep track of the current location which will be retained and used to detect cycles or multiple visits to the same location or nearby location and even the maze map can be created indicating where the obstacles lie in the map.

Once the coordinates have been calculated triangulation can now be used after two the beacon sensor values are received. Coordinates of both the locations where the beacon was requested is known - using these coordinates and the angle made by lines joining the beacon with these two points, the beacon coordinates, distance and angle from the present location can be calculated using triangulation. Using this information and the motion information from the previous 9 cycles (when the beacon was requested), direction of travel can be modified so that the robot moves in the direction of the beacon.

Triangulation, in trigonometry and geometry, is the process of finding coordinates and distance to a point by calculating the length of one side of a triangle, given measurements of angles and sides of the triangle formed by that point and two other known reference points, using the law of sines. For example, in the triangle shown in the figure 3, angles  $\alpha$  and  $\beta$  are known along with the distance AB.



**Figure 3. Triangulation example**

Now angle C and distances AC and BC can be calculated by law of sines and law of cosines as shown below

$$\gamma = 180 - \alpha - \beta$$

$$\frac{\sin \alpha}{BC} = \frac{\sin \beta}{AC} = \frac{\sin \gamma}{AB}$$

Cycle avoidance can be implemented by keeping track of the angle of rotation and using X-Y coordinates from the above equations, we can eliminate cycles.

After reaching the destination, the path back to the originating point is calculated using the originating point as (0, 0) and calculating the slope of the line connecting the current position (x, y) and moving in the direction of the origin. While moving towards the origin, the above mentioned wall following, obstacle avoidance strategies can be reused and the slope of the current position with respect to origin gives us the direction to move in order to reach as close as possible to the destination.

#### REFERENCES

- [1] Youngwoo Ahn, Ja-Ryeong Koo, Animesh Pathak, "CiberMouse motion control scheme for effective path finding," *RTSS CiberMouse 2006*.
- [2] Mariano Gomez Plaza, Sebastian Lopez Rodriguez, Sebastisn Sanchez Prieto, and Daniel Meziat Luna, <http://www.industrialcontroldesignline.com/howto/201802296>.
- [3] James Bruce, Manuela Veloso, "Real-Time Multi-Robot Motion Planning With Safe Dynamics".
- [4] Maze Solving Algorithms, <http://www.astrolog.org/labyrinth/algrithm.htm>.
- [5] RTSS CiberMouse 2007 Documentation, [http://www.ieeta.pt/~lau/web\\_ciberRTSS07/techdata.htm#docs](http://www.ieeta.pt/~lau/web_ciberRTSS07/techdata.htm#docs).

- [6] Nageswara S.V. Rao, Srikumar Kareti, Weimin Shi, S. Sitharama Iyengar, "Robot Navigation in Unknown Terrains: Introductory Surver of Non-Heuristic Algorithms," 1993].
- [7] Triangulation, <http://en.wikipedia.org/wiki/Triangulation>