

ITANDROIDS CiberMouse Agent Description Paper

Daniel L. Baggio, Humberto S. Naves, Denis F. Vitti, Jackson P. Matsuura

Instituto Tecnológico de Aeronáutica, São José dos Campos SP 12.233-066, BRA,
danielbaggio@gmail.com,
<http://www.itandroids.com>

Abstract. This paper describes the main points of ITANDROIDS Ciber-Mouse agent implementation, as well as how it uses an Extended Kalman Filter to estimate states and build a map in a noisy environment. The SLAM approach is explained in detail as well as the behavior of the agent, actuators and sensors. In order to explain better some aspects of the agent, pseudo-code has been specified.

1 Introduction

ITANDROIDS team is formed mainly by undergraduate students from Instituto Tecnológico de Aeronáutica and has the following goals:

- Establishing a strong research group in robotics at ITA, focused in cooperative robots and in the application of the technology and not only in pure research
- Help developing robotics research, knowledge and interest in Brazil

Since Ciber Mouse is aligned to ITANDROIDS goals, this paper describes the team's agent. In Ciber Mouse, several challenges are defined. According to [1], a robot must accomplish two goals in order to conclude its competition. Firstly, it must visit the target area. Inside that area it must turn-on the corresponding Beacon led, which must be turned-off after leaving it. After that, and only after that, the robot must return, as close as possible, to its position in the starting grid. The robot must signal the beginning of its return by turning-on its Return led inside the target area. The robot must also turn-on its End led to signal its competition is over. There is a time limit to accomplish the two goals, ranging between 1800 and 3600 u_t . The time limit can vary from scenario to scenario.

2 Sensing

The robots are equipped with several elements: 4 obstacle sensors, 1 beacon sensor, 1 compass, 1 bumper(collision sensor), 1 ground sensor, and a GPS. Ground, obstacle, beacon and compass sensors are only available on request, with a limit of two per cycle. The GPS is not available during competition but it is very useful for verifying accurate measures during debugging time.

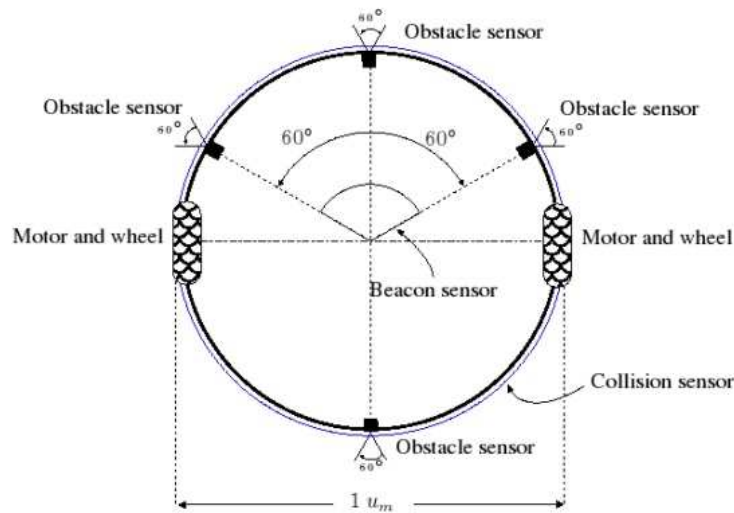


Fig. 1. Body of virtual mouse

2.1 Sensing obstacles

The robot is equipped with 4 *obstacle sensors* to detect walls and other robots. They can be placed in any position along the robot's periphery, and each has a 60-degree aperture angle. The measurements are inversely proportional to the minimum distance to the detected obstacles, and range from 0.0 to 100.0. There is also normal noise in the measurements, with a mean of 0 and a standard deviation of 0.25. The sensors have no delay.

2.2 Sensing beacon

As shown in Figure 1, there is a beacon sensor in the center of the robot. According to [1], it is turned forward with an aperture angle of 120 degrees, 60 degrees to each side. The beacon sensor is useful for finding the goal area. Besides finding the target within the aperture angle, the robot must not be in a shadowed area, which occurs if there is a high wall between it and the beacon. In order to build the world model of the maze, the robot takes the approach specified in 2.3.

2.3 SLAM

Simultaneous localization and mapping (SLAM) is a technique used by robots and autonomous vehicles to build a map within an unknown environment while at the same time keeping track of its current position. Since the information collected by the sensors has some noise, this is not a trivial task.

The basic idea around SLAM is that if there are inaccuracies in the first measurements and further observations do not verify their correctness, incorrect data is built up cumulatively, distorting the map and the robot's ability to

to know its precise location. Most of the techniques used to compensate these errors follow statistical approaches and scan matching of range data.

To model the robot, we have considered the following vector, which represents the position and orientation of the robot at time step k , as in [3],[5]:

$$x_R(k) = [x(k), y(k), \theta(k)]^T \quad (1)$$

Also, we define the vehicle's motion through a plant of the form:

$$x_R(k+1) = f_R(x_R(k), u(k)) + v(k) \quad (2)$$

where $u(k)$ is a control input, $v(k)$ is a noise disturbance, and $f_R(x_R(k), u(k))$ is the (non-linear) state transition function determined by the kinematics of the vehicle.

The environment is modeled with two types of targets: points and lines. The points encompass both corners and edges, while lines represent 3-D planes. The location of feature i at time k is specified by the parameter vector $x_i(k)$. For the points we have the vector $x_i(k) = [x_i(k), y_i(k)]^T$ where $x_i(k)$ and $y_i(k)$ specify the two dimensional location of the point. A line is represented by the vector $x_i(k) = [R_i(k), \theta_i(k)]^T$ where $R_i(k)$ is the perpendicular distance from the origin of the global coordinate frame to the line and $\theta_i(k)$ is the orientation of a perpendicular drawn from the origin to the line.

The data set $Z(k)$ is then built ($Z(k) = z_j(k) | 1 \leq j \leq m(k)$) from $m(k)$ measurements, following the measurement model described in [3]. Then, using Extended Kalman Filter (EKF) it is possible to find the Minimum Mean Square Error estimation for $x(k)$. The following procedures are then executed:

1. vehicle position update(relocation)
2. map update (fusion)

This way, an accurate map is generated although noisy measures are made.

3 Moving

In order for the robot to move, there are two diameterly opost motors, which might rotate the robot if asymeric power is set to them. Values ranging from -0.15 to $+0.15$ might be applied to them. As the motor is subjected to inertia and noise, the following equations are applied to them:

$$lOutPow_t = (lOutPow_{t-1} + lInPow_t)/2 \quad (3)$$

$$rOutPow_t = (rOutPow_{t-1} + rInPow_t)/2 \quad (4)$$

$$lNoisyOutPow_t = lOutPow_t * lNoise_t \quad (5)$$

$$rNoisyOutPow_t = rOutPow_t * rNoise_t \quad (6)$$

To rotate while walking forward, one approach is to mantain one of the actuators stopped while the other runs, so to turn right, one should apply 0.0 to the right motor and 0.15 to the left one.

A pretty simple algorithm might be followed to avoid collision. This is described by the following pseudo-code:

```
if(FrontSensor > 4.0 or LeftSensor > 4.0 or RightSensor > 4.0)
    // front, left and right have obstacles
    DriveMotors(0.1,-0.1);    // only rotate
else if(LeftSensor > 1.0)    // obstacle left
    DriveMotors(0.1,0.0);    // turn right
else if(RightSensor > 1.0)  // obstacle righth
    DriveMotors(0.0,0.1);    // turn left
```

4 Behavior

The core behavior of the robot is to avoid collision with other robots and always look for the beacon.

The algorithm used to go around obstacles is described in [4]. It says that if the robot counts 2 corners more than the number of edges the object has been transpassed.

5 Results

Currently, the implementations are being tested, so, all the results are partial. As this agent includes some features which have not yet been observed in other teams that have already competed, some interesting results may come from this implementation.

6 Conclusion

The agent implemented by ITANDROIDS has incorporated several features that may be applied to a real robot and it shows that the Extended Kalman Filter approach for state estimation is a powerful tool to be used in a noisy environment.

References

1. CiberMouse Rules
http://www.ieeta.pt/~lau/web_ciberRTSS/docs/ciberRTSSrules.pdf
2. L. Almeida, J. L. Azevedo, B. Cunha, P. Fonseca, N. Lau, A. Pereira: Micro-Rato Robotics Contest: Technical Problems and Solutions (2006)
3. Leonard, John J.; Durrant-Whyte, Hugh F. "Simultaneous map building and localization for an autonomous mobile robot". Proc. IEEE Int. Workshop on Intelligent Robots and Systems (1991) 1442-1447.
4. Pedro, Lus; Bruno, Martins; Pedro, Almeida; Valter, Silva; "Detecção de Configurações de Obstáculos Perigosas: Aplicação no Robô EnCuRRalado"
5. Arleo, A., del R. Millan, J., and Floreano, D. (1999). Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. IEEE Transactions on Robotics and Automation, 15(6):991-1000.