Course Project Documentation

CS684 Project

Autonomous attacker / follower robot for moving target

TEAM : 5

Viraj Churi, 10305072
Ketan Mav, 10305030
Niket Bagwe, 10305086
Linga Venkatesh, 10305085

Table Of Contents

1. **Introduction**:

Now-a-days, there are lots of applications where an autonomous follower and attacking(or action) mechanism is implemented such as in Army-defence, war flights fire missiles aiming to some enemy and the missile follows the enemy flight and attack it. Here, the path of missile will be controlled by using radio signals by the attacker.

In this, the missile depends on attacker to take an action which is un-reliable because the connection between the missile and attacker need to maintained through-out.

This limitation has become the motivation to do the project. Here, we come up with system in which we add intelligence to “following-bot”(attacker bot) to follow the enemy autonomously and attack it.

2. Problem Statement:

The aim of the project is to design an autonomous robot which will follow and shoot enemy robot in the battlefield. Battlefield will be a plain restricted area and should be free of any obstacles.

Enemy robot can be stationary or moving within the Battlefield. Enemy robot will carry a cylindrical identification mark of some specific color.

The attacker robot will continuously track movements of enemy robot to follow him. Goal is to reach within specified range from enemy before shooting him with a laser. This is so because the range of the laser is limited.

3. Requirements:

A) Hardware Requirements

1. FireBird : Requires 2 bots (1 Attacker + 1 Enemy)
2. Camera : To view position of enemy in battlefield
3. Marker : For uniquely identify enemy in battlefield
4. Lazer : Used to fire the enemy
5. Zigbee : To maintain communication between attacker and system
6. Proximity Sensor : To measure the distance between attacker and enemy

B) Software Requirements

1. OpenCV : For processing images sent by camera
2. Color Segmentation : An algorithm which is both efficient and fast enough to maintain real time segmentation of images sent by camera
3. AVR studio : To program instruction onto a given bot

4. Implementation:

A) Functionality:

a) **Determining position:** Enemy robot's position would be determined by segmenting the identification mark from the image. Identification mark is detected using color segmentation. From this, the direction in which the robot is to be moved can be determined. Once the enemy robot comes within the range of the proximity sensors, accurate distance can be measured between the robots. From this, we can determine when the laser is to be fired.

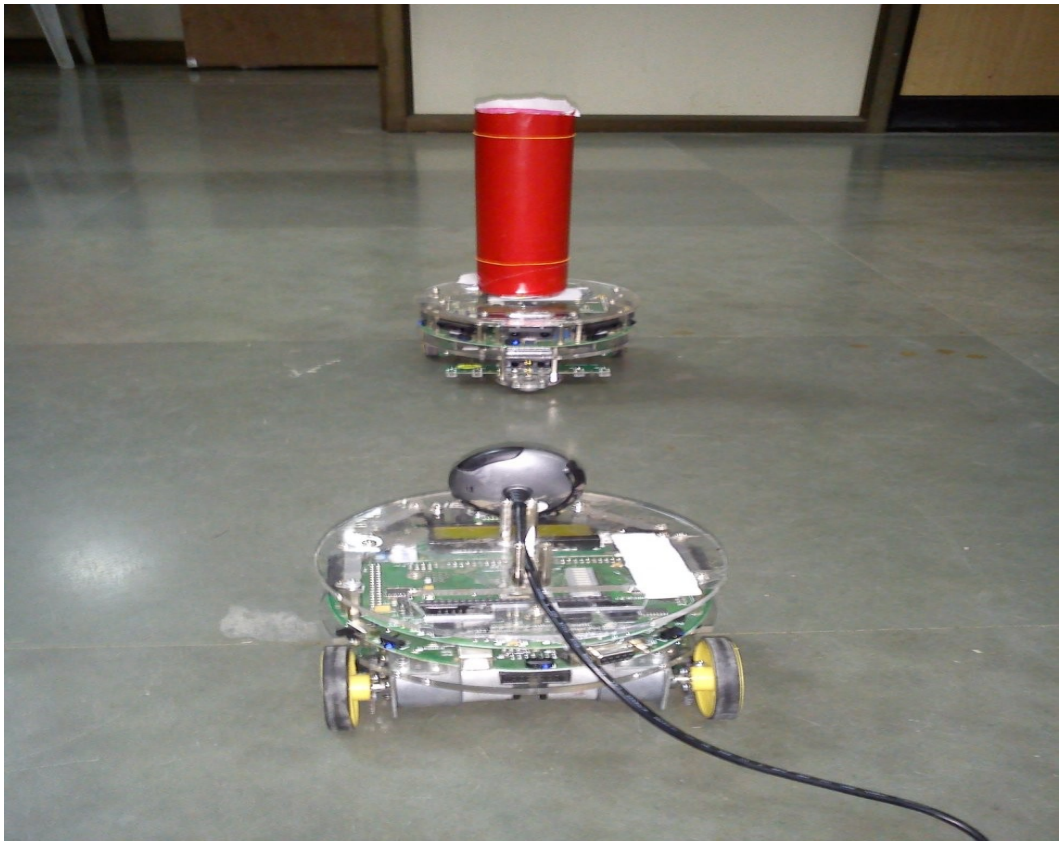
b) **Moving attacker robot:** Depending on the rate at which enemy robots speed is varying, attacker robots speed will vary. Approximate distance between the two robots can be calculated by measuring the size of the identification mark from the image. This will determine how fast the attacker robot should move to chase the enemy bot. There would be some restrictions to the speed of the enemy robot for a successful chase. This can only be determined after examining other hardware dependent factors such as motors maximum speed, processing delay, delay introduced due to communication, robots maneuverability.

5. Testing Strategy and Data:

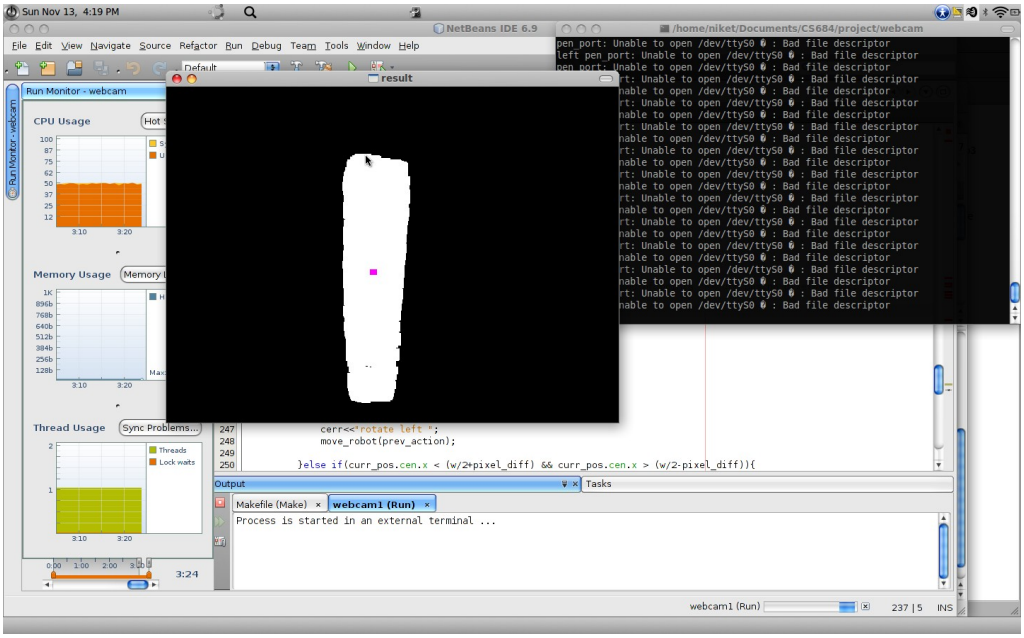
The camera will send captured images at a given frame-rate to the system where image processing techniques are implemented. At the end of processing a window named “result” will be generated(as shown in following figure). The white-colored rectangle in “result” window represent marker which is mounted on enemy bot. A Pink colored pixel seen at the center of rectangle is centroid of the white rectangular area.

- Identification mark is on the center of the image.

Bots orientation:

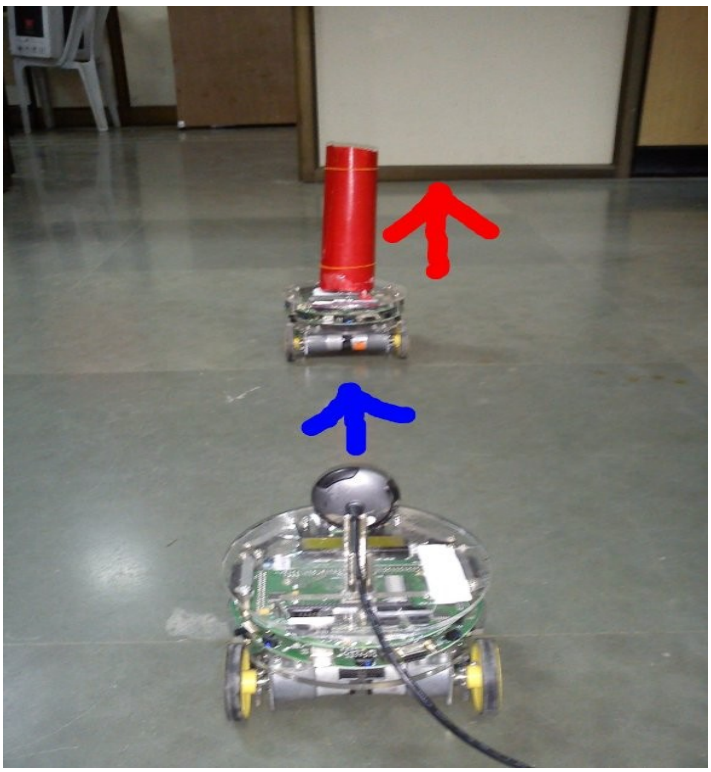


Result after Image Processing:



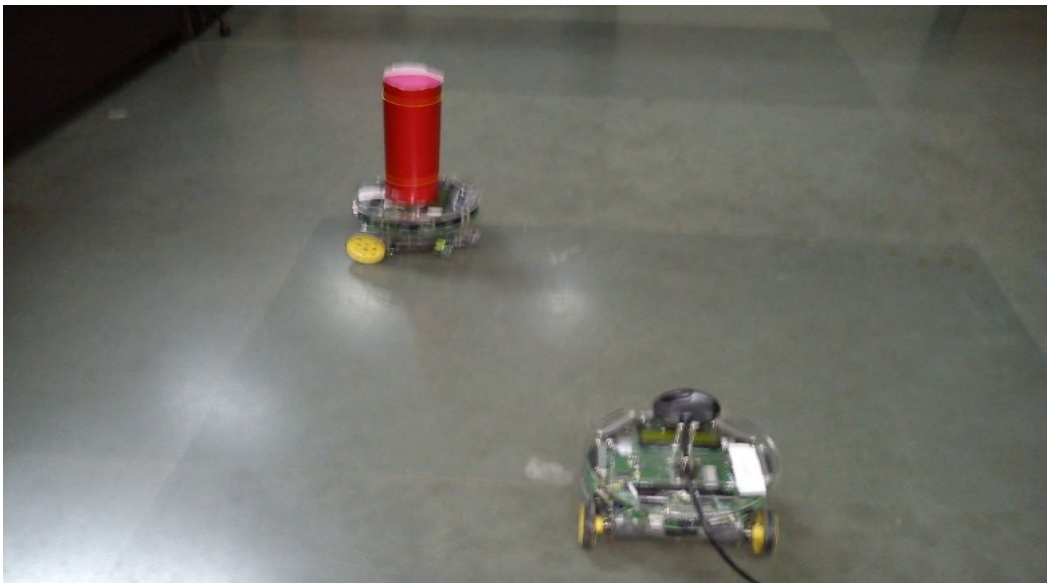
This tells that the enemy is in front of attacker and attacker will shoot at him depending upon the proximity sensor value.

Resulting Orientation of Bots:

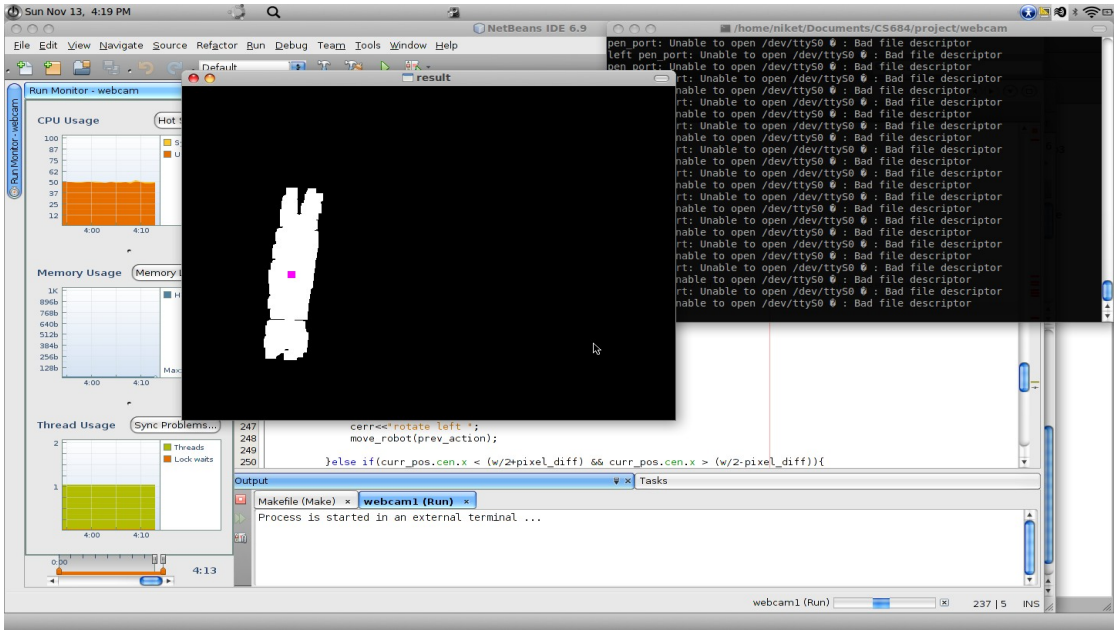


- Identification mark is on the left side of the image.

Situation:

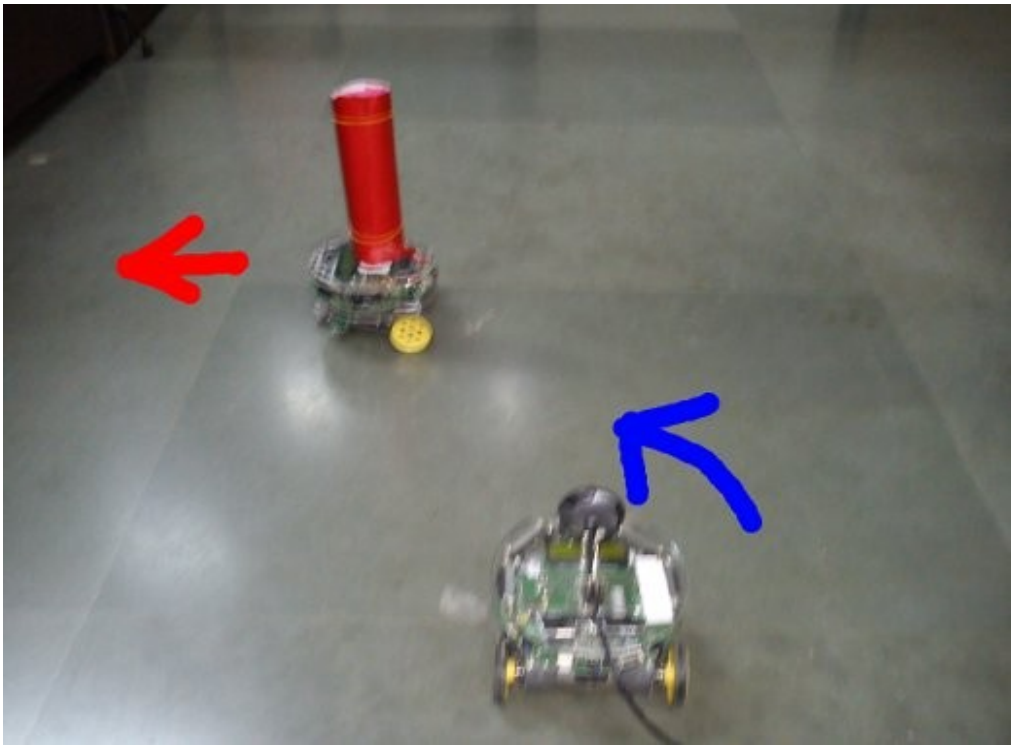


Result after Image Processing:



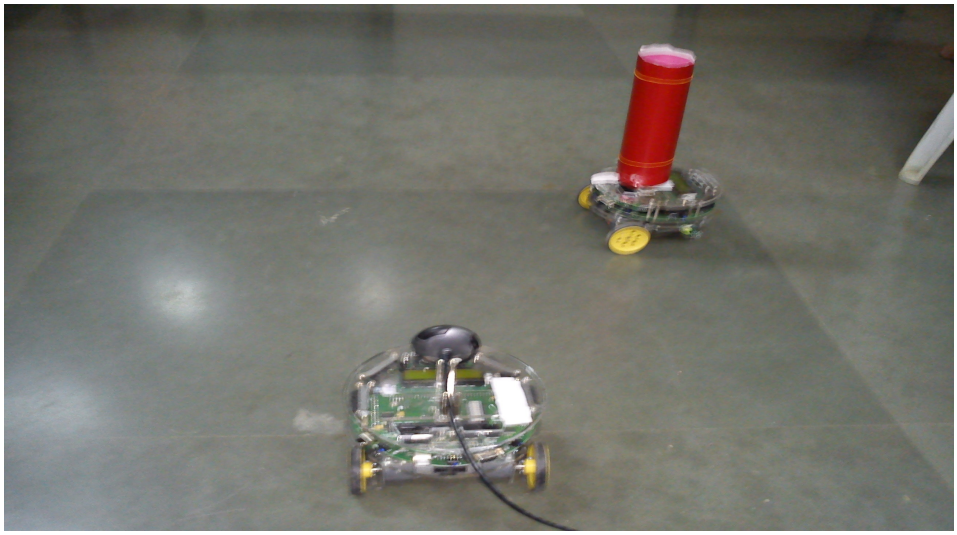
This shows the enemy is left side of the attacker. Attacker moves towards his left side as commanded by algorithm.

Resulting Orientation of Bots:

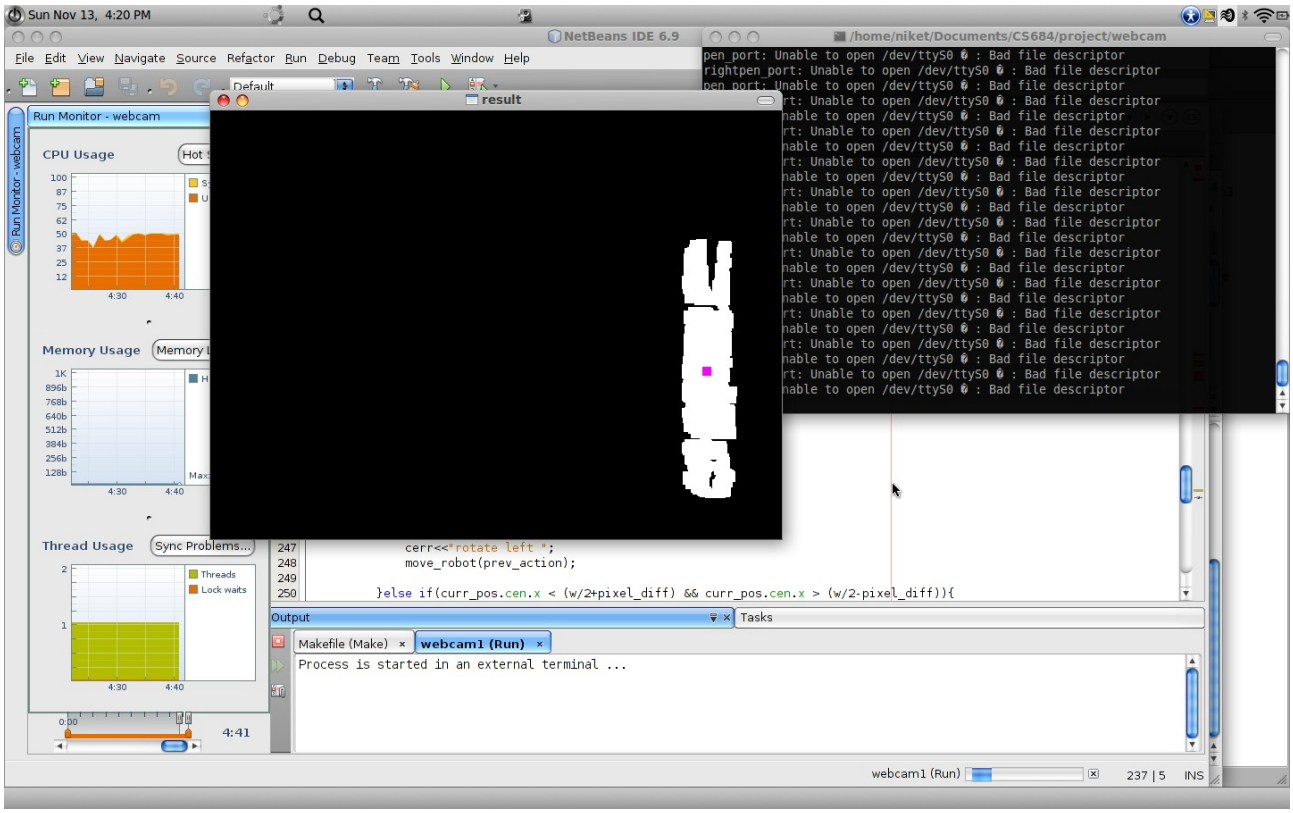


- Identification mark is on the right side of the image.

Situation:

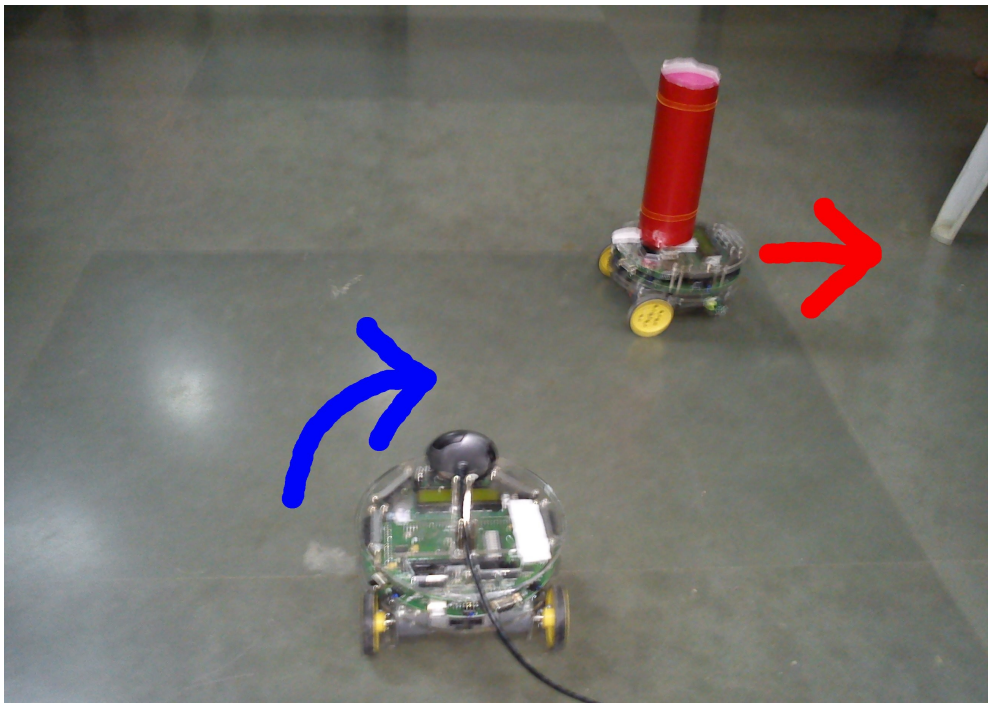


Result after image Processing:

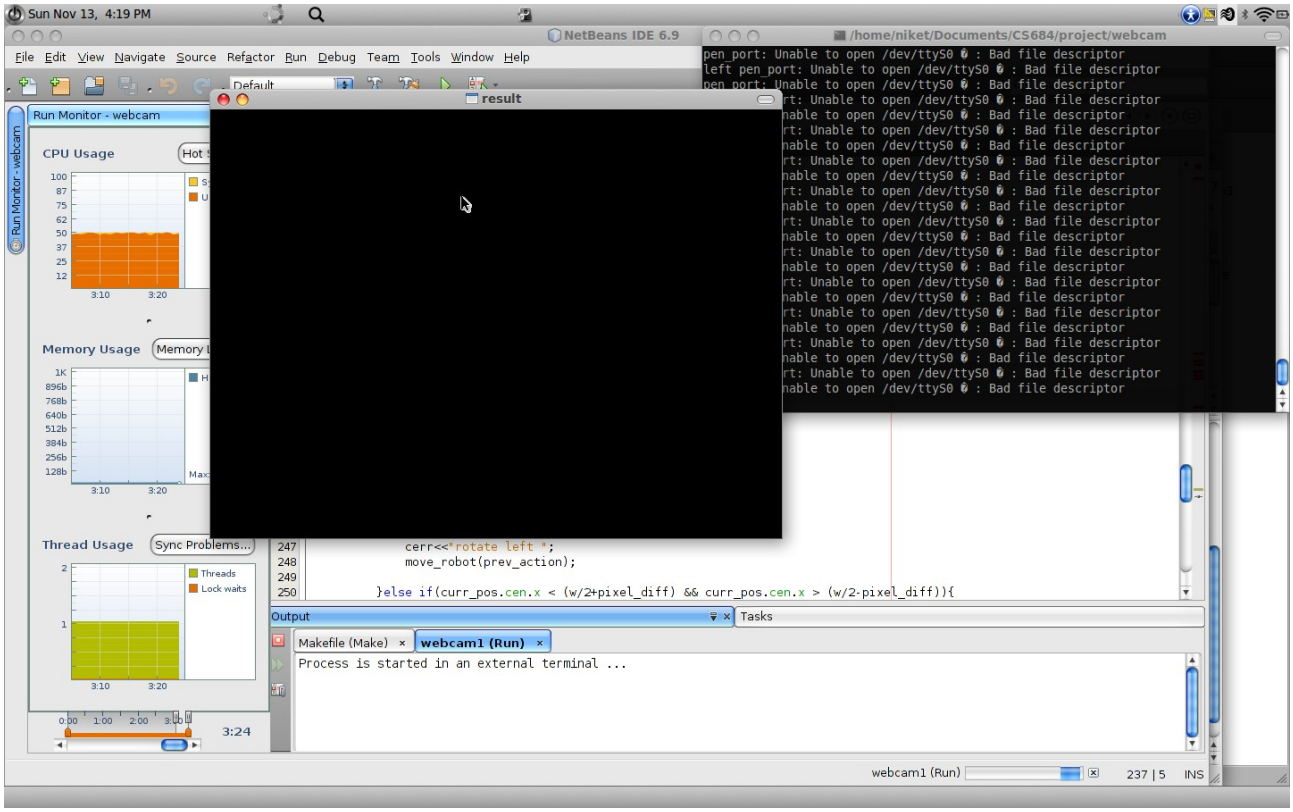


This shows the enemy is right side of the attacker. Attacker moves towards his right side as commanded by algorithm.

Resulting Orientation of Bots:

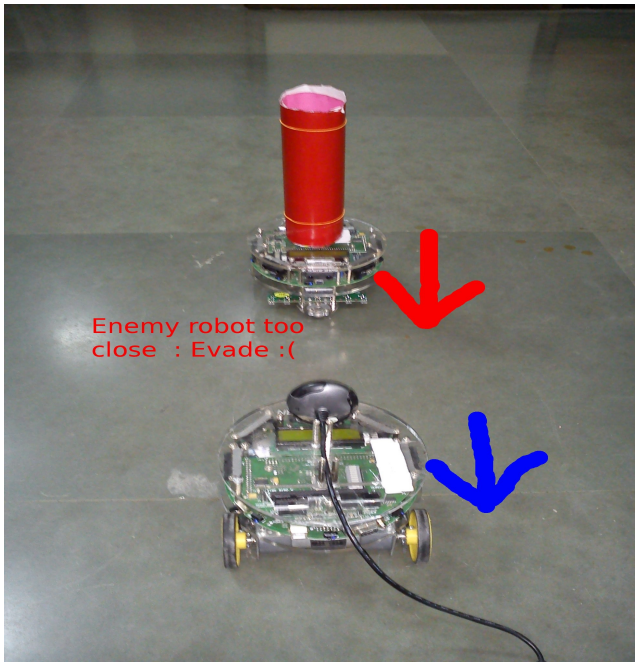


- Identification mark is not visible in the image



In this case the marker(i.e. Enemy bot) is not in visible area of the camera. The attacker bot moves either left or right side depending upon it previous action.

- When Enemy is too close to attacker bot:



6. Discussion of System:

A) What are worked as per plan?

1. Cylindrical Marker on Enemy bot:

Selection of marker shape and color is very crucial. We selected cylindrical marker as it looks rectangle in the image taken by camera from any view point(provided there are no occluding objects). Also the color of marker is selected such that it is not present in battlefield.

2. Efficiency of Segmentation algorithm:

The efficiency and response time of the segmentation algorithm is very important to maintain realtime following of attacker bot. The segmentation algorithm is designed effective with response rate of 8 per second.

3. Zigbee Communication:

Successfully able to communicate with the required bot by using Zigbee and able to take appropriate action at real time.

4. Operating Lazer Gun:

Firing of laser gun is done successful by mounting laser gun on servo pins of attacker bot which is turned ON and OFF depeding upon Proximity Sensor value and the action given by segmentation algorithm to bot Via zigbee.

B) What we added more than discussed in SRS?

1) Previous action based control:

In-order to avoid redundant rotation of attacker bot when the marker is not visible in its viewing area we implemented previous action based controlling of bot i.e. When the marker is not visible on screen then attacker bot will rotate in the same direction of the previous.

2) Obstacle Detection and Avoidance:

We are able to sucessfully distinguish between obstacle and enemy bot by using image processing technique and proximity sensor. Also firing at the non-enemy bots and any obstacles present in battlefield is avoided.

(C) Changes made in plan:

1. OpenCV installation:

Initially, we thought of working in windows platform but the installation of OpenCV didnt worked well. Hence we moved to Ubuntu where the installation is done successful.

Probable Reason:

Managing the dependencies of OpenCV libraries and windows operating system became complicated.

7. Future Work:

A) In our current implementation, we used only 1 camera for attacker, by using more number of cameras each viewing in different directions will improve the performance(response time) of the system.

B) This work can also be extended to have more than one enemy-bots in the arena with same or different color markers on them(all known to attacker bot).

C) We have implemented “abstacle detection” and avoided attacking non-enemy bots or any obstcles in the battlefield. By implementing obstacle avoidance based actions may give better results.

8. Conclusions:

Our work can be generailsed as **“Follow + Action”.** An action can be anything depending upon application. Some of the real-world application in which this can be applied like in entertainment, lots of games can be designed like “Tom-and-Jerry” (bot-pair) in which user controls Jerry(enemy-bot) and Tom (attacker-bot) tries to attack the jerry.

9. References:

1) OpenCv Installation Guide: "https://help.ubuntu.com/community/OpenCV" (Last viewed on: 12 Nov 2011)

2) Scarecrow Robot By E-Yantra Team : http://www.e-yantra.org/ci/projects/code/486 (Last viewed on: 12 Nov 2011)

3) Autonomous Target Acquisition & Engagement By E-Yantra Team http://www.e-yantra.org/ci/projects/code/484 (Last viewed on: 12 Nov 2011)

4) http://opencv.willowgarage.com/wiki/CameraCapture (Last viewed on: 12 Nov 2011)