Embedded System

CS-684

Course Project : RCTB Error Detection

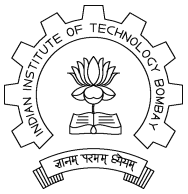
by

Team-10

Deepak Jayanth P M	(10305913)
Pratik Patodi	(10305917)
Firuza Aibara	(p11119)
Alluri Srinivas	(10305076)

under the guidance of

Prof. Kavi Arya



Department of Computer Science and Engineering
Indian Institute of Technology Bombay

November 2011

Contents

List of Figures

List of Tables

1 Introduction

In Navy Environments its common to pratise shooting on Remote controlled Target Ships, and detect the accuracy of the missile launches on it. The accuracy is detected in terms of the distance between the target boat and missile hit location. We will model a small scale prototype of this system using bots in the hope that when this small scale system is solved first, then it would address and solve the real world problem too.

2 Problem Statement

Navy war ships play a vital role to guard our Nation. To practise shooting, the shooting ship lauches its missile on the target ship. The target ship may be controlled remotely to keep moving. The aim of this project is to determine whether the target was hit at the right place or not. And also measure its accuracy. This accuracy is very important during war times. If the target was not hit at the correct position, then the difference i.e. the deviation is to be determined. This means by what value the target position was missed. The challenge is to find this accuracy.

3 Requirements

The hardware and the software requirements for the project are listed below-:

3.1 Hardware Requirements

- One Windows based system with 4 USB ports.
- Three USB Camera
- Three Zigbee Modules.
- Two Spark V robots.

3.2 Software Requirements

- Scilab 5.3.3 (Windows).
- Scilab Image and Video Processing Toolbox(SIVP).
- Serial communication.
- AVR Studio 4.
- X-CTU software to configure ZigBee modules.

4 Implementation

The Project can be classified briefly into following modules/stages based on the functionalities done in order.

- **Image Processing Module** : This module is implemented using Scilab SIVP toolbox. The Input images are captured from the camera and processed frame by frame. Each Frame is then processed for detection of the objects based on color. In our project we are detecting 4 objects based on 4 different colors(red,green,blue,pink). The Functionalities of each module are explained in detail under the topic Discussion of System
- **Serial Communication Module** : This module is responsible for communicating the corrective measures the bot should take, in-order to align itself to the center of the target. Zigbee Communication chip is mounted on both mother and shooting bot, and both are configured to receive the serial communication signals send by the PC on broadcast mode.

5 Discussion of System

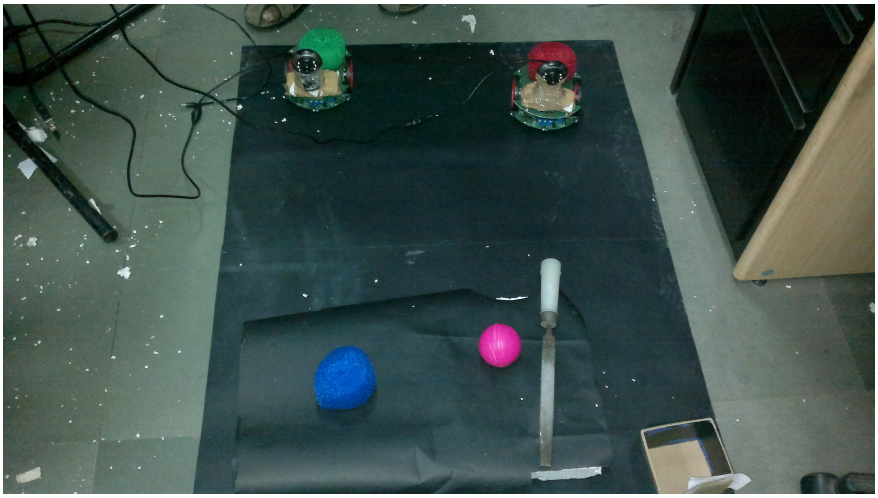


Figure 1: Experimental Setup

5.1 Object detection and Tracking

The following steps will be common to all object detection based on color.

1. The First step in the object detection is to get an input frame from the camera.
2. Convert the input image from RGB(3D array) to HSV color space(3D array).
3. Check the Hue(H),Saturation (S), Value (V) color range of the color we need to track. All 3 values ranges from 0.0 to 1.0
4. When this values are fixed, check all pixels of the 3D HSV array of image satisfying these values.
5. The resultant value gives the index of the detected color. and save them in a black and white image, where white represents the detected index, and black represents the other pixels.

6. To detect the center of the detected object, browse through the each columns to find the maximum occurance of white pixels. the column which gives the maximum number of count is the center of the object along x direction.

7. Do the above step along the row to find the maximum occurrence of white pixel along row wise to detect the objects center along y direction.

8. The coordinates(maxcol,maxrow) gives the center of the detected object.

5.2 Top Camera

Figure1 shows the experimental setup of our system. The first step is to find the distance between the mother ship and target ship(d1), shooting ship and target ship(d2). This camera act similar to the real world RADAR mounted on the ship to find the distance of any target. The top camera detects 4 objects. Each object is detected by means of the 'Object Detection and Tracking Step' mentioned earlier based on the color we need to detect. In this project, the Red Color is the Mother Bot, Green Color is the shooting bot, Blue Color is for the target. And Rose color for splash. Though the splash color is detected from top camera in this step, it is not required to calculate the distance d1 and d2. The Second Image of Figure 2 shows the detected 4 colors.

The Third image of Figure 2 is marked with Red and Green color lines to indicate what values are calculated in this phase. On the main program the function 'topcam(frame)' calculates the values mentioned in this step.

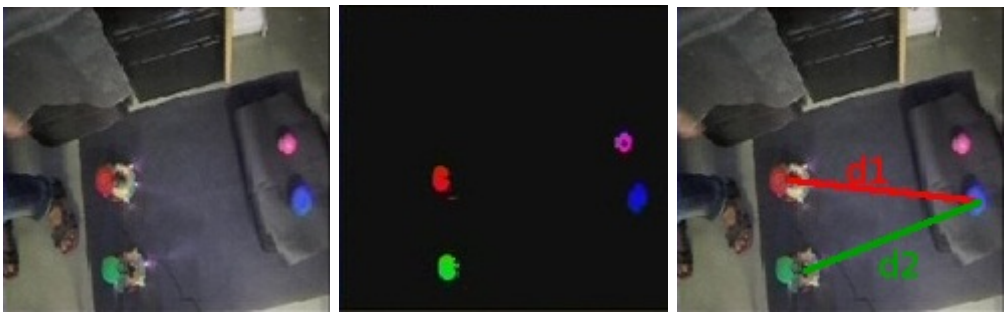


Figure 2: Top Camera View

5.3 Serial Communication

The serial Communication module is mainly responsible for sending the corrective action to the mother and shooting bot inorder to align itself towards the target. The corrective action(message) is broadcasted to the both the bots. Based on the message send either the mother bot or the shooting bot takes the corrective measure to the signals broadcasted, but not both at a time. The message is of two bytes. The first byte contains the node id of the bot. And the second byte contains the control message(left or right or no action). On receiving the message both bots check the node id of the message and if the node id equals the bot id, then it will execute the control message. Otherwise it discards the message. Consider the following images. All the above images are obtained from the on-board camera after it

Figure 3: On Board Camera After Target Detection

had went through the image processing steps mentioned earlier.

- The first image of the Figure shows that the image the target color blue is on the left of its window.

  Corrective Action: direct the bot to move towards right.

- The second image of the Figure shows that the image the target color blue is on the left of its window.

  Corrective Action: direct the bot to move towards left.

- The first image of the Figure shows that the image the target color blue is on the left of its window.

  Corrective Action: No corrective action, just set the variable center=true.

5.4 On board cameras

The Next Step is to detect the angle between the splash(missile landing) and the target ship , from the mother ship$\Theta 1$ and the shooting bot$\Theta 2$. This Step can be done only after the previous step is successful. ie. Two bots set the variable center to true, which means both the bots are aligned towards the target.

5.5 Error detection

Here is a function used to calculate the error (i.e. the distance between the target and the splash)

```
function err = func(motX,motY,targX,targY,shootX,shootY,the1,the2)
      mother=[motX;motY];

      target=[targX;targY];
      shoting=[shootX;shootY];
      theta1=the1*\varPi/180;
      theta2=the2*\varPi/180;
```

```
      splash1=rotate(target,theta1,mother);
      splash2=rotate(target,theta2,shoting);
      a11=(mother(2)-splash1(2))/(mother(1)-splash1(1)+1);
      a21=(shoting(2)-splash2(2))/(shoting(1)-splash2(1)+1);
      a22=-1;
      if (shoting(1) ~= splash2(1) & mother(1) ~= splash1(1)) then,
    c1=(mother(2)*splash1(1)-splash1(2)*mother(1))/(mother(1)-splash1(1));
    c2=(shoting(2)*splash2(1)-splash2(2)*shoting(1))/(shoting(1)-splash2(1));
    A=[a11,a12;a21,a22]
    C=[c1;c2]
    B=A\C;
    x=target(1)-B(1);
    y=target(2)-B(2);
    err=sqrt(abs((x*x)+(y*y)));
      else
    err=0;
      end
endfunction
```

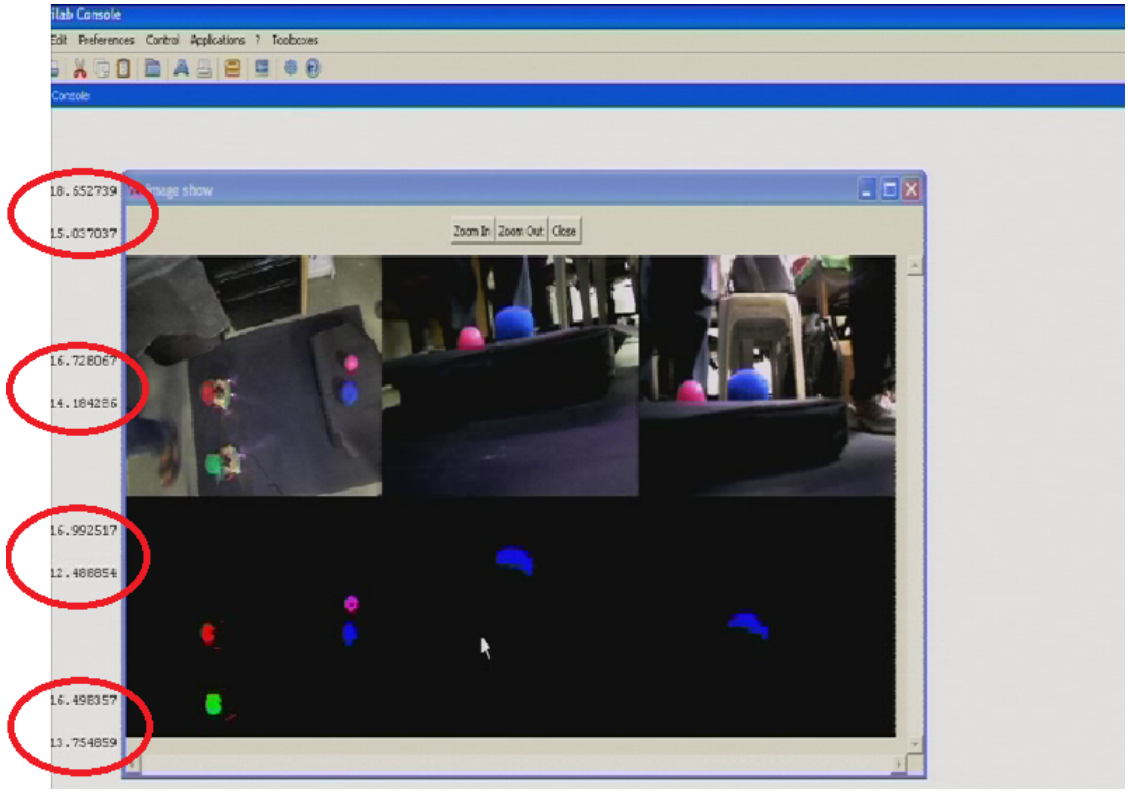


Figure 4: Final Result window

This function takes the co-ordinates of mother bot, shooting bot, target bot, as well as the angles, which is the angular deviation of the splash from the taget bot. We supply both the angles $\Theta 1$ and $\Theta 2$ respectively (i.e. angle with respect to mother bot and angle with respect to shooting bot).Now we rotate the target position by $\Theta 1$ about mother bot position and get the equation of the line passing through mother bot co-ordinates and the new position we have got of target bot after rotation. Similarly we get the equation of line passing through shooting bot and the new position of target bot after rotation by a $\Theta 2$ about the shooting bot. We calculate the point of intersection of these two line which will give us the co-ordinates of the splash. Now we calculate the distance between the splash and target bot which is the error value we want.

Figure4 shows six images, top row (first 3 images) shows the original picture taken by topcamera, motherbot and shooting bot. Bottom row (next 3 images) shows the corresponding filtered images(color detected and tracked) of the original images. The error detected by solving the equations formed by distances d1,d2, and $\Theta 1$, $\Theta 2$, which is the final output of our project. There are two values printed on the screen. Second value is the error detected by solving the equation. First Value is the value detected by calculating error directly from the topcam. This output is encircled by a red color oval. As we see from the results the error detected by our method is same as the error detected by topcam with +/- 3 cms.

5.6 Problems Encountered

The following are the problems which we have faced and ways to tackle it.

- **Case 1** :
  - **Problem**:First we used point-to-point communication i.e. one zig bee adapter for each bot to transmit the control information between the bot and the laptop, as a result we require many Usb ports for laptop. when we executed this by using a usb hub it showed so many errors.
  - **Mitigation Plan**:we used only one zig bee adapter with broadcast communication. The Message contains two parts one is node-id and other is control command. The bot will check if it is intended for it or not. if it is intended for it it executes the command otherwise it discards.
- **Case 2**:
  - **Problem**:While detecting the object using scilab first we searched pixel by pixel for that color as a result it takes a lot of time to search i.e it is not responding in real time.
  - **Mitigation Plan**: We used Scilab Matrix methods since scilab is optimized for matrix operations. In particular we used scilab *find* command to find the predicates satisfying a condition instead of if loop and used the index returned by *find* command to process further.
- **Case 3**:
  - **Problem**: Many times we found that one or two of the connected USB cameras where not detected by scilab.

- **Mitigation Plan**: This problem was mainly reported because whenever USB's are connected to PC, the PC names it such as USB connection 5, USB connection 6, USB connection 7 ..etc. The problem occurs in scilab only if the numbers are not arranged in a proper order, for eg. if the numbers are 5,6,7 there wont be a problem. but if there exists a gap in numbering such as 5,6,8 or 5,7,8 then scilab doesn't detect it properly. So to ensure scilab works correctly we inserted USB's in trial and error methods to see that all USB's are numbered sequentially.

6 Testing Strategy

The following are the test cases, which were used to test our system.

- **Case 1** : When the mother bot,shooting bot have already aligned themselves towards the target.
  - **Expected Result**: This case is an easy case to pass, since there is no serial communication corrective action required by the shooting and mother bots. The Error can be calculated directly from the angle measurements and radar distances.
  - **Result**: This test case was passed.
- **Case 2** : When the target is on the view of sight of the on board cameras but its not exactly at the center.
  - **Expected Result**: In this case, since the bots are not aligned towards the target, the error detection cannot be done unless the two bots align themselves towards the target first. Here the bots align themselves towards the target first by moving left or right according to the corrective action given by zig-bee communication.
  - **Result**: This test case was passed.
- **Case 3** : When target is moving by means of adjusting the position of it in runtime. right.
  - **Expected Result**: In this case whenever the target is moved out of the center of the on-board camera view, the bot should realign itself to make itself facing the center of the target. After both the bots are aligned towards the center the new error value should be displayed.
  - **Result**: This Test Case was also passed.
- **Case 4** : When the splash is towards the right side of the target.
  - **Expected Result**: This test case is similar to case 2 or 3 except the position of splash is changed.
  - **Result**: Test case passed.
- **Case 5** : When the splash is towards the left side of the target. either on left or right.
  - **Expected Result**: This test case is similar to case 2 or 3 except the position of splash is changed.
  - **Result**: This Test Case was also passed.
- **Case 6** : When the splash is aligned itself in-front of the target.
  - **Expected Result**: This is a complicated case since it is occluding the target bot by being in-front of it. In this case there is a possibility that the splash covers

the entire target and makes it difficult for the detection of the target color. The accuracy of this test depends on the size of the splash in comparison to the size of the target. If both are at same size then the splash completely blocks one of the bot from detecting the target.

- **Result**: In our case the target and the splash were almost of same size, so the target almost completely got occulted by the splash. This test case didn't pass successfully because of the inherent nature of the test case.
- **Remedy**: This test case could be made to pass by simply keeping a smaller splash size. due to time constrains we didnt check with a new splash object.

7 Project Scope

In addition to solving the problem of error detection of hit on Remote controlled target boat(RCTB), the code developed for this project can be utilized in many other projects since this is developed on open-source sci-lab. Some of the modules which can be reused are

- **Object Detection and Tracking** : The code detects simultaneously 4 different colored objects, and track them. Since HSV color space is used instead of RGB color space the accuracy of tracking under different lighting conditions remains steady. The ranges of the colors are well set and thresholded so that it does not clash with other objects. since object detection and tracking is the heart of any system based on image processing, this code will be really reused on many places.
- **Real-Time Monitoring** : Since Scilab is optimized for the use of matrix calculations, almost all the calculations are done by means of matrix and vector operations to speed up the calculations. The resulting code is so much fast these methods can be followed on any real time monitoring projects. In addition to this the restriction of only one image of imshow is removed by combining multiple images in matrix method in no time and show the output instantaneously.
  Note: If for loop is used to merge two images it takes around 1 to 2 minutes to merge two images on a canvas, whereas the matrix method does that in milliseconds.

8 Future Work

This project can be extended in the following ways to include additional features

- **Dynamic Splash** : The Splash (pink color object) detected in this project can be dynamically changed by making it an actual projectile by means of a object thrower.
- **Dynamic and evasive target** : The same project should be tested with the evasive bot done by other teams and make that as the target bot and see how the systems copes up with a dynamic and evasive bot which evades the attacker by means of tricky calculations.

9 Conclusions

This project thus models a prototype to solve the problem of error detection of the missile launches on the target bot. In addition to solving the problem, the project made us aware of the problems and challenges needs to be faced when implementing an embedded systems.

Developing this project gave us a good idea on Embedded Systems and made us realise the intrinsic problems in designing, and implementing a real world project, and where things doesn't go completely as planned on a real world since each hardware module has inherent problems on its own which are very difficult to spot and work with. Altogether the entire coding done is well documented and modularized to be reused on many situations.