**Autonomous Soccer Playing Robot**

**TEAM: 11**

**PARMINDER SINGH BADWAL**

**ASHISH MAHAJAN**

**SHRIJIT PANDEY**

**Table of Contents**

1. Introduction..................................................................................1

2. Problem Statement.......................................................................2

3. Requirements...............................................................................3

4. Implementation............................................................................4

5. Testing Strategy and Data............................................................

6. Discussion of System..................................................................

7. Future Work................................................................................

8. Conclusion...................................................................................

**1. INTRODUCTION:**

The development of autonomous, intelligent systems has been in the focus of computer science, engineering, and cognitive science since many years. In a wide field of disciplines many techniques have been developed which can serve as building blocks for these kinds of systems. Examples are logical systems for reasoning and representation, visual perception systems, machine learning, and cognitive architectures, among others. However, all of these techniques cover only small subsets of all abilities a fully autonomous system must be equipped with to be able to operate in a complex environment.

The development of information technology in the forthcoming decades will demand informatics to leave applications that are exclusively computer-based and instead occupy itself with the development of daily life objects (e.g. white goods, cars, etc.) that contains “intelligence”. A number of skills are needed in order to construct such objects. The skills integrate different competences and not only the ability to make a perfect software application. In general, skills are needed to construct hardware objects that are governed by “intelligent” or “adaptive” software that continuously interacts with a changing and non-deterministic real world.

A football robot is a specialized autonomous robot and mobile robot that is used to play variants of soccer. This project encompasses various computer science disciplines like Real Time Robotics, Embedded Systems, Artificial Intelligence (up to some extent), communication using ZigBee, Image Processing using Matlab/Scilab, etc. This project is being done under the guidance of ERTS lab, IIT Bombay. Thus, it is a great learning experience for us.

**2. PROBLEM STATEMENT:**

* Designing and developing a method for processing vision data into commands for motion control for autonomous soccer playing robot swarm.

* Autonomous robots are robots that can perform various designated tasks by understanding the environment without continuous human guidance.

* There will be 2 Spark V robots in the arena, one goal-keeper and one striker.

* The goal-keeper exhibits autonomous behaviour so there is no human control over its play except the control of starting and stopping the game. Each team is controlled and coordinated by a controller which makes decisions based on the images of the arena obtained.

* The striker which is humanly controlled always tries for pushing the ball in the goal.

**3. REQUIREMENTS:**

Ø Hardware requirements:

1. Spark V robots (2) which act a soccer player. One Striker and one goal keeper.
2. XBEE module (4) for wireless communication between robots and computer.
3. Camera (1) for Image processing.

Ø Software requirements:

1. Win AVR
2. openCV
3. Windows XP or higher.
4. Code::Blocks
5. cvBlob library

The robots don’t have individual intelligence, they are controlled by a central controller takes the digital image as input and returns commands to robots via wireless medium for motion control.

**4. IMPLEMENTATION DETAILS:**

1. Implementation of color detection.

We used basic thresholding for color detection. Thresholding filters out the pixels of color out of specified range. Thresholding is an operation which highlights pixels above a certain thresh-hold. In order to implement ranged thresh-holding, we have to perform thresholding twice.

Tr = T1 – T2

Ex. Suppose we want to detect ‘PINK’ color.

For HSV color space, pink lies between values 150-170 (for value of h). So we have to perform ranged threshold of range 150-170 to identify PINK. We first perform thresholding for value 150 and then perform thresholding for 170.

Let T1 = Image after thresholding with value 150.

Let T2 = image after thresholding with value 170.

In order to get desired output, subtract T2 from T1.

Thus, the resulting image containing only PINK pixels (Tr) is given by,

Tr = T1 – T2.

// code snippet

Mat getThresholdedImage(Mat img, int lb, int hb)

{

Mat retimg(img.rows, img.cols, CV\_8UC1);

Mat t1(img.rows, img.cols, CV\_8UC1);

Mat t2(img.rows, img.cols, CV\_8UC1);

threshold(img, t1, hb, 255, THRESH\_BINARY);

threshold(img, t2, lb, 255, THRESH\_BINARY);

subtract(t2, t1, retimg);

return retimg;

}

1. Implementation of colored object detection.

Once thresholding has been performed and we get an image with only a specific color highlighted. Now we have to perform region detection in that image. As only PINK colored regions are present in the image, on performing region detection, we would be able to tell where the PINK colored object is present in the field of vision of the camera.

Region detection is performed using connected component labeling algorithms. Algorithm we use is provided by CVBLOB library.

// code snippet

void getPosition(Mat currimg, Point2D \*pos)

{

Mat thresh = processor->getThresholdedImage(currimg, this->threshold\_range.lower\_bound, this->threshold\_range.upper\_bound);

Mat smooth = processor->getSmoothImage(thresh, 3);

namedWindow("BotPos");

imshow("BotPos", smooth);

cout << "last1\n";

IplImage \*dispimg = cvCreateImage(cvSize(currimg.rows, currimg.cols), IPL\_DEPTH\_LABEL, 3);

IplImage orig = smooth;

cvShowImage("fsmooth", &orig);

cout << "last2\n";

cvLabel(&orig, dispimg, blobs);

CvLabel lab = cvb::cvGreaterBlob(blobs);

cvb::cvFilterByLabel(blobs, lab);

cout << "last3\n";

//cout << "number of blobs detected: " << blobs.size();

if(blobs.size() != 0)

{

for (CvBlobs::const\_iterator it=blobs.begin(); it!=blobs.end(); ++it)

{

pos->posx = it->second->centroid.x;

pos->posy = it->second->centroid.y;

}

}

cout << "get pos end reached\n";

cvReleaseImage(&dispimg);

cvReleaseBlobs(blobs);

}

1. Implementation of Wireless communication.

Wireless communication is performed by using a ZigBee transmitter/receiver pair. We have one trans-receiver on robot and one trans-receiver connected to the computer. On the robot, we communicate with xbee module using serial communication protocol (SPI). When connected to the computer, we communicate via a COM Port (parallel port). Both the trans-receivers need to be paired before using. For pairing, we set destination address of one trans-receiver equal to source address of other. We can then write to the allocated COM Port and it will be transmitted automatically to the other paired trans-receiver.

// code snippet for writing to COM port under windows.

public void Communications()

{

portHandle = CreateFile(\_T("COM9"), // Specify port device: default "COM1"

GENERIC\_READ | GENERIC\_WRITE, // Specify mode that open device.

0, // the devide isn't shared.

NULL, // the object gets a default security.

OPEN\_EXISTING, // Specify which action to take on file.

0, // default.

NULL);

}

void sendCommand(int n)

{

DWORD nbw = 0;

int res = WriteFile(portHandle,&n,1,&nbw,NULL);

if(res == FALSE)

{

cout << "Couldnot write command!\n";

}

cout << "Number of bytes written:" << nbw << endl;

}

1. Implementation of soccer defending algorithm.

The algorithm is implemented as follows,

1. Detect the position of ball at time T1.
2. Sleep for some time.
3. Detect position of ball again (at time T2).
4. Calculate slope and intercept for the line passing through the detected positions.
5. Knowing the x co-ordinate of the goal, we can substitute it in the line equation to get the y- co-ordinate along the goal line or the goal point.
6. If the ball position is within the error range, don’t move the goal keeper robot.
7. If goal point is greater than the goal keeper position move forward else move backward.

**5. TEST CASES:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Test Case | Expected Result | Error on failure |
| 1. | Check whether blobs are detected | YES | undetected blobs due to ambient light |
| 2. | Check whether the color of blobs is detected | YES | undetected blobs due to ambient light |
| 3. | Check whether centroid of blob is accurately detected or within margin of error | YES | undetected blob centroid due to ambient light |
| 4. | Check whether centroid of ball is detected. | YES | undetected ball centroid due to ambient light |
| 5. | Check whether centroid of robot marked with colored circles is detected or not | YES | undetected blobs due to ambient light |
| 6. | Check whether XBEE communication between robot and PC takes place | YES | serial communication error |
| 7. | Check whether motion control commands are sent via XBEE from PC to robot | YES | unsent commands and communication error |
| 8. | Check whether motion control code is burnt properly on robot memory and robot does motion control accordingly | YES | serial communication error |
| 9. | Check whether Soccer Defend Algorithm properly detects the goal points. | YES | improper goal point out of goal range |
| 10. | Check whether Goalkeeper robot moves towards the goal point and blocks the ball from entering the goal | YES | robot moves in wrong direction |

**7. FUTURE SCOPE:**

The Project can be modified for various applications including the following:

- Soccer playing swarm with additional intercept states

- Teams can be scaled to higher number of robots

- A mechanism to hold the ball and push it.

**8. Conclusion:**

Hence we have created a system which is completely autonomous in its goal defending capability. We have used a research platform robot called a SPARK V which is has a microcontroller called ATMEGA 16 and some basic sensors and interfaces for any required attachment. The software (OpenCV, Code::Blocks, AVR) have been so programmed and designed so as to support all the different essential functionalities of the system. Important aspects of image processing are handled by OpenCV. AVR STUDIO handles the programming of the microcontroller. XBEE wireless modules are used to handle wireless communication. Thus we have been able to cover so many different technologies and aspects. These technologies together make sure that the system is accurately autonomous.

**9. REFERENCES:**

[1] Spark V ATMEGA Hardware Manual

[2] Spark V ATMEGA Software Manual

[3] manual\_xb\_oem-rf-modules\_802.15.4 for XBEE communication

[4] http://opencv.willowgarage.com/wiki

[5] http://en.wikipedia.org/wiki/ANSI\_C%2B%2B

[6] http://www.e-yantra.org/home/studio/tutorials

[7] http://www.nex-robotics.com/spark-v-robot.html