1RE01-Tech-No-Logic

This pdf file contains different program which were used in our project. The programs are as follows-

- Image Processing Program (C++ code).
  The program that we used for image processing basically detects the centroid of a given colored object in HSV format. Then according to the detection of the centroid in different regions, the bot is moved forward, backward, right and left. The whole source code is given below-

```
#include <iostream>
#include "serial.h"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <math.h>
#include <unistd.h>
using namespace cv;
using namespace std;

float dist(Point center1, Point center2){
      float dist;
      dist= sqrt((center1.x-center2.x)*(center1.x-
center2.x)+(center1.y-center2.y)*(center1.y-center2.y));
      return dist;
      }

 int main( int argc, char** argv )
 {
      char c;
      serial_device arduino;
      arduino.initialize("/dev/ttyUSB0");
    VideoCapture cap(0); //capture the video from webcam

    if ( !cap.isOpened() )  // if not success, exit program
    {
```

```
         cout << "Cannot open the web cam" << endl;
         return -1;
    }

    namedWindow("Control", CV_WINDOW_AUTOSIZE); //create a window
called "Control"

int iLowH = 86;
int iHighH = 104;

int iLowS = 0;
int iHighS = 255;

int iLowV = 0;
int iHighV = 255;

//Create trackbars in "Control" window
createTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
createTrackbar("HighH", "Control", &iHighH, 179);

createTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)
createTrackbar("HighS", "Control", &iHighS, 255);

createTrackbar("LowV", "Control", &iLowV, 255);//Value (0 - 255)
createTrackbar("HighV", "Control", &iHighV, 255);

int iLastX = -1;
int iLastY = -1;

int width= cap.get(CV_CAP_PROP_FRAME_WIDTH);
int height=cap.get(CV_CAP_PROP_FRAME_HEIGHT);
cout<<width<<endl;
cout<<height<<endl;

float c1x=width/2, c1y=0;
float c2x=0, c2y=height/2;
float c3x=width/2, c3y=height;
float c4x=width, c4y=height/2;


//Capture a temporary image from the camera
Mat imgTmp;
cap.read(imgTmp);

//Create a black image with the size as the camera output
Mat imgLines = Mat::zeros( imgTmp.size(), CV_8UC3 );;


    while (true)
    {
        Mat imgOriginal;
```

```
        bool bSuccess = cap.read(imgOriginal); // read a new frame
from video



         if (!bSuccess) //if not success, break loop
        {
             cout << "Cannot read a frame from video stream" << endl;
             break;
        }

Mat imgHSV;

cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert the captured
frame from BGR to HSV

Mat imgThresholded;

inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS,
iHighV), imgThresholded); //Threshold the image

//morphological opening (removes small objects from the foreground)
erode(imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(10, 10)) );
dilate( imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(10, 10)) );

//morphological closing (removes small holes from the foreground)
dilate( imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(10, 10)) );
erode(imgThresholded, imgThresholded,
getStructuringElement(MORPH_ELLIPSE, Size(10, 10)) );

//Calculate the moments of the thresholded image
Moments oMoments = moments(imgThresholded);

double dM01 = oMoments.m01;
double dM10 = oMoments.m10;
double dArea = oMoments.m00;

// if the area <= 10000, I consider that the there are no object in
the image and it's because of the noise, the area is not zero
if (dArea > 10000)
{
//calculate the position of the ball
int posX = dM10 / dArea;
int posY = dM01 / dArea;

if (iLastX >= 0 && iLastY >= 0 && posX >= 0 && posY >= 0)
{
//Draw a red line from the previous point to the current point
```

```
circle(imgOriginal,Point(iLastX, iLastY),10 , Scalar(0,0,255),-1);
}

iLastX = posX;
iLastY = posY;
}
else{
     iLastX = -1;
    iLastY = -1;
}

imshow("Thresholded Image", imgThresholded); //show the thresholded
image

imgOriginal = imgOriginal + imgLines;
imshow("Original", imgOriginal); //show the original image
if(dist(Point(iLastX, iLastY),Point(c1x,c1y))<=160)
{
     cout<<"Region 1"<<endl;
     c='a';
     arduino.write_byte(c);

}
else if(dist(Point(iLastX, iLastY),Point(c2x,c2y))<=160)
{
     cout<<"Region 2"<<endl;
     c='b';
     arduino.write_byte(c);
}
 else if(dist(Point(iLastX, iLastY),Point(c3x,c3y))<=160)
{
     cout<<"Region 3"<<endl;
     c='c';
     arduino.write_byte(c);
}
 else if(dist(Point(iLastX, iLastY),Point(c4x,c4y))<=160)
{
     cout<<"Region 4"<<endl;
     c='d';
     arduino.write_byte(c);
}
else
{
     cout<<"Do nothing"<<endl;
     c='o';
     arduino.write_byte(c);
}


        if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If
'esc' key is pressed, break loop
       {
```

```
            cout << "esc key is pressed by user" << endl;
            break;
        }
    }

   return 0;

}
```

- Serial Port Program (C++ code)
  Our image processing programs gives out output in form of different characters (a, b, c, d, o) according to the presence of centroid in different regions. The serial port code writes the characters on the serial port which is then transferred to the arduino mounted on the bot using Xbee.
  The serial port code is as follows-

```
#include <stdio.h>      // standard input / output functions
#include <stdlib.h>
#include <string.h>     // string function definitions
#include <unistd.h>     // UNIX standard function definitions
#include <fcntl.h>      // File control definitions
#include <errno.h>      // Error number definitions
#include <termios.h>    // POSIX terminal control definitions
#include <time.h>
#include <string.h>

using namespace std;

class serial_device {
      private:
            int USB;

      public:
            void initialize(char* port);
            void write_bytes(char*, int);
            void write_byte(char);
};

void serial_device::initialize(char* port) {
      USB = open(port, O_RDWR | O_NOCTTY );
      struct termios tty;
      struct termios tty_old;
      memset (&tty, 0, sizeof(tty));

      /* Error Handling */
      if ( tcgetattr ( USB, &tty ) != 0 )
      {
```

```
      //cout << "Error " << errno << " from tcgetattr: " <<
strerror(errno) << endl;
      }

      /* Save old tty parameters */
      tty_old = tty;

      /* Set Baud Rate */
      cfsetospeed (&tty, (speed_t)B9600);
      cfsetispeed (&tty, (speed_t)B9600);

      /* Setting other Port Stuff */
      tty.c_cflag     &=  ~PARENB;        // Make 8n1
      tty.c_cflag     &=  ~CSTOPB;
      tty.c_cflag     &=  ~CSIZE;
      tty.c_cflag     |=  CS8;

      tty.c_cflag     &=  ~CRTSCTS;       // no flow control
      tty.c_cc[VMIN]      =   1;                  // read doesn't block
      tty.c_cc[VTIME]     =   5;                  // 0.5 seconds read
timeout
      tty.c_cflag     |=  CREAD | CLOCAL;     // turn on READ & ignore
ctrl lines

      /* Make raw */
      cfmakeraw(&tty);

      /* Flush Port, then applies attributes */
      tcflush( USB, TCIFLUSH );
      if ( tcsetattr ( USB, TCSANOW, &tty ) != 0) {
//          cout << "Error " << errno << " from tcsetattr" << endl;
      }
}

void serial_device::write_bytes(char* str, int len) {
      write(USB, str, len);
}

void serial_device::write_byte(char str) {
      write(USB, &str, 1);
}
```

- GUI code (python)
  The GUI code is used to create a GUI which is for the ease of access to the project and using which you are studying this pdf right now ☺ ☺.

The coed for the GUI is as follows-

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'tp.ui'
#
# Created: Fri May 30 19:04:06 2014
#      by: PyQt4 UI code generator 4.9.1
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui
import os

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    _fromUtf8 = lambda s: s

def run():
     os.system("./tp")

def open_manual():
     os.system("gnome-open manual.pdf")

def open_about():
     os.system("python about.py")
class Ui_Form(object):
    def pushButtonClicked(self):
            self.label_3.setText(QtGui.QApplication.translate("Form", "
Enjoy!!!", None, QtGui.QApplication.UnicodeUTF8))

    def setupUi(self, Form):
        Form.setObjectName(_fromUtf8("Form"))
        Form.resize(630, 542)
        Form.setStyleSheet(_fromUtf8("background-color:
qlineargradient(spread:pad, x1:1, y1:0, x2:1, y2:1, stop:0 rgba(0, 0,
0, 255), stop:1 rgba(255, 139, 139, 255));\n"
"background-color: qlineargradient(spread:pad, x1:1, y1:0.0909091,
x2:1, y2:1, stop:0 rgba(0, 0, 0, 255), stop:1 rgba(172, 255, 232,
255));"))
        self.label = QtGui.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(100, 10, 431, 51))
        self.label.setStyleSheet(_fromUtf8("font: 75 16pt
\"Serif\";\n"
"font: 18pt \"Sans Serif\";\n"
"color: rgb(255, 255, 255);\n"
""))
        self.label.setObjectName(_fromUtf8("label"))
        self.label_2 = QtGui.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(480, 50, 101, 21))
```

```
        self.label_2.setStyleSheet(_fromUtf8("color: rgb(255, 255,
255);\n"
"font: italic 14pt \"Sans Serif\";\n"
"font: italic 12pt \"Sans Serif\";"))
        self.label_2.setObjectName(_fromUtf8("label_2"))
        self.label_3 = QtGui.QLabel(Form)
        self.label_3.setGeometry(QtCore.QRect(200, 140, 261, 91))
        self.label_3.setStyleSheet(_fromUtf8("background-color: rgb(0,
0, 0);\n"
"font: 12pt \"Serif\";\n"
"color: rgb(0, 255, 0);"))
        self.label_3.setObjectName(_fromUtf8("label_3"))
        self.widget = QtGui.QWidget(Form)
        self.widget.setGeometry(QtCore.QRect(280, 290, 115, 95))
        self.widget.setObjectName(_fromUtf8("widget"))
        self.verticalLayout = QtGui.QVBoxLayout(self.widget)
        self.verticalLayout.setMargin(0)
        self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
        self.pushButton = QtGui.QPushButton(self.widget)
        self.pushButton.setStyleSheet(_fromUtf8("background-color:
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:1 rgba(61,
156, 235, 255));\n"
"color: rgb(0, 0, 0);"))
        self.pushButton.setObjectName(_fromUtf8("pushButton"))
        self.verticalLayout.addWidget(self.pushButton)
        self.pushButton_2 = QtGui.QPushButton(self.widget)
        self.pushButton_2.setStyleSheet(_fromUtf8("background-color:
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:1 rgba(61,
156, 235, 255));\n"
"color: rgb(0, 0, 0);"))
        self.pushButton_2.setObjectName(_fromUtf8("pushButton_2"))
        self.verticalLayout.addWidget(self.pushButton_2)
        self.pushButton_3 = QtGui.QPushButton(self.widget)
        self.pushButton_3.setStyleSheet(_fromUtf8("background-color:
qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:1 rgba(61,
156, 235, 255));\n"
"color: rgb(0, 0, 0);"))
        self.pushButton_3.setObjectName(_fromUtf8("pushButton_3"))
        self.verticalLayout.addWidget(self.pushButton_3)

        self.retranslateUi(Form)
        QtCore.QObject.connect(self.pushButton,
QtCore.SIGNAL(_fromUtf8("clicked()")), self.pushButtonClicked)
        QtCore.QObject.connect(self.pushButton,
QtCore.SIGNAL(_fromUtf8("clicked()")), run)
        QtCore.QObject.connect(self.pushButton_2,
QtCore.SIGNAL(_fromUtf8("clicked()")), open_manual)
        QtCore.QObject.connect(self.pushButton_3,
QtCore.SIGNAL(_fromUtf8("clicked()")), open_about)
        QtCore.QMetaObject.connectSlotsByName(Form)
```

```
    def retranslateUi(self, Form):
        Form.setWindowTitle(QtGui.QApplication.translate("Form",
"Form", None, QtGui.QApplication.UnicodeUTF8))
        self.label.setText(QtGui.QApplication.translate("Form", "
Image-Processing Controlled Bot", None,
QtGui.QApplication.UnicodeUTF8))
        self.label_2.setText(QtGui.QApplication.translate("Form", "
ITSP 2014", None, QtGui.QApplication.UnicodeUTF8))
        self.label_3.setText(QtGui.QApplication.translate("Form", "
Welcome to Tech-No-Logic!!!", None, QtGui.QApplication.UnicodeUTF8))
        self.pushButton.setText(QtGui.QApplication.translate("Form",
"RUN", None, QtGui.QApplication.UnicodeUTF8))
        self.pushButton_2.setText(QtGui.QApplication.translate("Form",
"INSTRUCTIONS", None, QtGui.QApplication.UnicodeUTF8))
        self.pushButton_3.setText(QtGui.QApplication.translate("Form",
"ABOUT", None, QtGui.QApplication.UnicodeUTF8))


if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    Form = QtGui.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

THANK YOU

☺☺