Progress report

CAMERA BASED LASER BEAM TRACKING AND DRIVING PIEZO FOR BEAM STEERING

INTRODUCTION

Here we will use python and OpenCV to detect a bright laser spot using camera. We will continously obtain the centroid of contour of bright spot and send this data (position of spot on screen) to Arduino Nano via serial library. There this data will be processed and if the new centroid differs from the one in the first frame then PWM signals will be given to TIDA-00856 Piezo actuator driver to position the beam back to the spot/position as in first frame.

Detection of spot (Python):

Firstly we will read a frame from webcam and convert it into grey scale image. Then we binarize the image with the adequate threshold (in our case its from 225 to 255).

```python
def detect_laser_points():

    cap = cv2.VideoCapture(0)


    if not cap.isOpened():
        print("Failed to open webcam")
        return

    while True:
        ret, frame = cap.read()

        if not ret:
            print("Failed to capture frame")
            break

        # Convert the frame to Gray scale image
        gry = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # defining the range or threshold
        mask = cv2.inRange(gry, 225,255,cv2.THRESH_BINARY)
```

After that we will perform morphological operations to remove noise from image. First an erosion is applied to remove the small blobs, then a dilation to regrow the size of the bright object.

```python
# Perform morphological operations to remove noise
mask = cv2.erode(mask, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)
```

Finally we will find contours in this masked image. Using

RETR_EXTERNAL to get only extreme outer flag. Then we will find the area of all contours in frame and the centroid of one with the largest area.

```
# Find contours in the masked image
contours, _ = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMP

# Process each contour
for contour in contours:
    # Calculate the area of the contour
    area = cv2.contourArea(contour)

    if area > 10000:
        # Find the centroid of the contour
        M = cv2.moments(contour)
        centroid_x = int(M["m10"] / M["m00"])
        centroid_y = int(M["m01"] / M["m00"])
        data = f"{centroid_x},{centroid_y}\n"
```

Circles can also be drawn around object with labels. Atlast we will send these coordinates to arduino via COM Port using serial library.

```
# Draw a circle at the centroid in frame
cv2.circle(frame, (centroid_x, centroid_y), round((area/(np.pi))**0.5), (255, 255, 255),
cv2.putText(frame,(str(centroid_x)+','+str(centroid_y)),(centroid_x+20,centroid_y+30),cv

# sending data
serial.Serial('COM4', 9600).write(data.encode())
```

Steering of Beam(Arduino Ide):

Firstly, we will initialize some coordinate variables and PWM output pin.

```
#include <SoftwareSerial.h>
int x_initial; // original coordinates/ coordinates of object in first frame
int y_initial;
int x;          // new/further incoming coordinates
int y;
int pin_x= 9;
void setup() {
  Serial.begin(9600);

  pinMode(pin_x,OUTPUT); // PWM output pin

}
```

Now we will decode the incoming data and store the value of x and y coordinates.

```
int m=0;
int power_x=0;
void loop() {
  if (Serial.available()) {
    // Read the incoming data from Python
    String data = Serial.readStringUntil('\n');


    int commaIndex = data.indexOf(',');
    if (commaIndex != -1) {
      String xStr = data.substring(0, commaIndex);
      String yStr = data.substring(commaIndex + 1);

       x = xStr.toInt();
       y = yStr.toInt();


      Serial.print("X=");
      Serial.print(x);
      Serial.print(", Y=");
      Serial.println(y);

    }
```

Now as arduino will get the coordinates from first frame, we will store it as the original position of beam spot.

```
if(m==0){
  x_initial=x;
  y_initial=y;
  m++;
}
```

And if any of the further coordinates will differ from the original coordinates, we will provide a PWM output (between 0 to 255) with monotonically increasing value if new position is to right and decreasing if new position to lift of original position. We will continue doing it till new position is similar to original position. Also a delay of 10 ms is provided after each iteration.

```
if(x - x_initial >=5){

    power_x=power_x+10;
    analogWrite(pin_x,power_x);
   delay(10);

  }

if(x - x_initial <= -5){

    power_x=power_x-10;
    analogWrite(pin_x,power_x);
    delay(10);
  }
```

Similar if cases can also be written for y-axis.

As TIDA-00856 takes a maximum voltage of 3.3V, therefore we have to use a Voltage divider to reduce 5V PWM output to 3.3V.