OBJECT TRACKING THROUGH CENTROID DETECTION AND APPROXIMATION OF ANGULAR DEFLECTION

Vineet Verma
Assistant Professor,
Dept of Electrical and Electronics Engineering,
Anand Engineering College, Agra, India.

Abstract: Object detection is most prevalent step of video analytics. This paper presents the implementation of object tracking using MATLAB based centroid detection and other feature extraction like object eccentricity. This method eliminates the use of learning and correlation algorithms and hence offers an easier implementation. This paper also proposes a feasible way of predicting the angular deflection of servo based on rate of change of centroid deflection.

***Index Terms* - Arduino, Centroid detection, MATLAB, Object tracking, Servo Control,**

I. INTRODUCTION

In order to track any moving object with the help of a servo-controlled camera set-up, in this paper we need to first detect the moving object and then locate its centroid. This variable centroid needs to be given in such a way to the servos of the camera such that the targeted object always remains in sight or the field of view of the camera. Object detection can be done through training the system on the basis of training vectors and then correlating the output for a best match but that is complicated and this report presents a simpler way for object detection and tracking.

II. METHODOLOGY OF OBJECT DETECTION IN MATLAB

The object detection is done through MATLAB using a function called REGIONPROPS that returns the properties of regions detected. The properties of a region detected by REGIONPROPS are classified into Area, Centroid, Bounding Box, Subarray, Major Axis Length, Minor Axis Length, Eccentricity, Orientation, Convex Hull, Convex Image, Convex Area, Image, Filled Image, Filled Area, Euler Number, Extrema, Equivalent Diameter, Solidity, Extent and others.

Continuous regions are also called "objects", "connected components", and "blobs". A label matrix containing contiguous regions might look like this:

```
1 1 0 2 2 0 3 3
1 1 0 2 2 0 3 3
```

Discontinuous regions are regions that may contain multiple connected components and are filtered out by the function.

III. IMAGE PROCESSING FOR OBJECT DETECTION

The function REGIONPROPS works on 1 dimensional image only. Thus, there is a need to convert the input image which is 3 dimensional as it contains red, green and blue vectors. This conversion can be in two ways:

1. First convert the image to a gray scale image which is a 2-dimensional image and then convert the 2-dimensional image to a single one using an edge detection algorithm.

2. Threshold the image using a threshold value between 0 and 1.

Edge detection is basically a part of spatial filtering. Spatial filtering is carried out using basically integrative or derivative type. Integrative type of filters smoothens out the details while derivative type of filters highlights the minute details. One such edge detector is the SOBEL operator, other edge detection algorithms are SOBEL, ROBERTS, PREWITT and CANNY. The SOBEL operator is used in this case as it is faster than the rest. The SOBEL operator also is less sensitive to noise and aberrations.

The reason of using edge detection was basically found out upon trying it several times over different set of images. They are reproduced here to highlight the same.
As it is evident from these images, the SOBEL operated image offers closer resemblance to the original image than the thresholded image. Moreover, object detection algorithm of REGIONPROPS works better with edge operated images than with thresholded images.

It is evident from observing these images that the centroid detection is much better on edge operated images than the thresholded images. Also, the images filtered by “eccentricity” property of REGIONPROPS yield better results rather than when only the maximum area object is detected. We thus proceed further by taking these results into consideration.

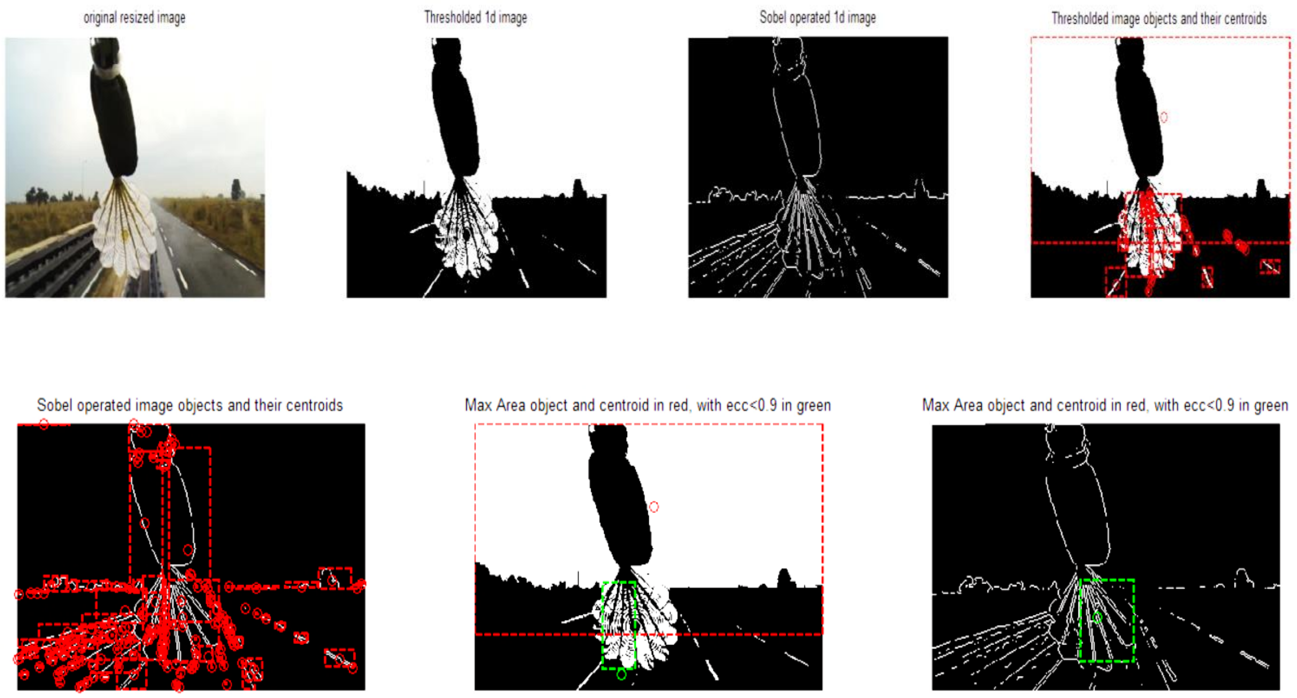

original resized image
Thresholded 1d image
Sobel operated 1d image
Thresholded image objects and their centroids
Sobel operated image objects and their centroids
Max Area object and centroid in red, with ecc<0.9 in green
Max Area object and centroid in red, with ecc<0.9 in green

Figure 1: First set of comparison between centroid detection through Sobel operator (green) VS centroid detection through thresholding (red)

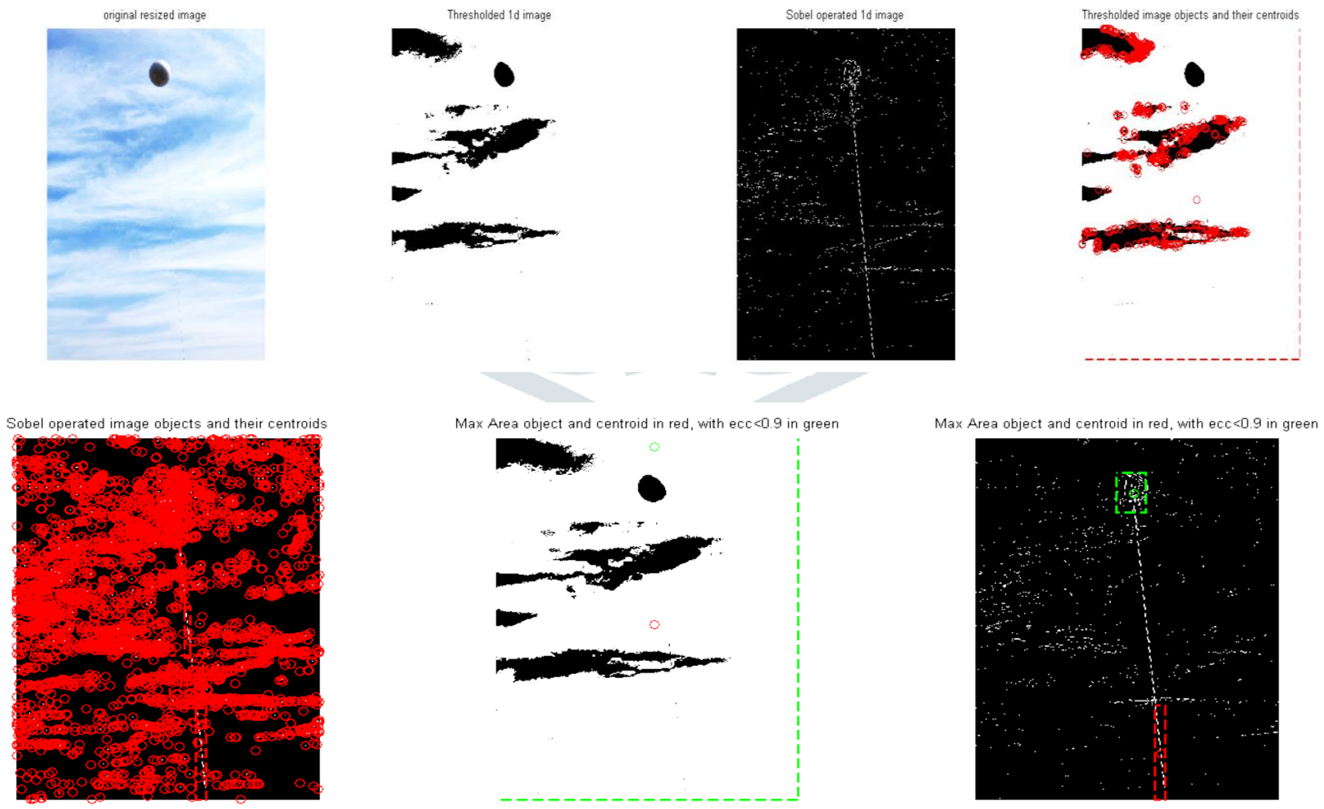

original resized image
Thresholded 1d image
Sobel operated 1d image
Thresholded image objects and their centroids
Sobel operated image objects and their centroids
Max Area object and centroid in red, with ecc<0.9 in green
Max Area object and centroid in red, with ecc<0.9 in green

Figure 2: Second set of comparison

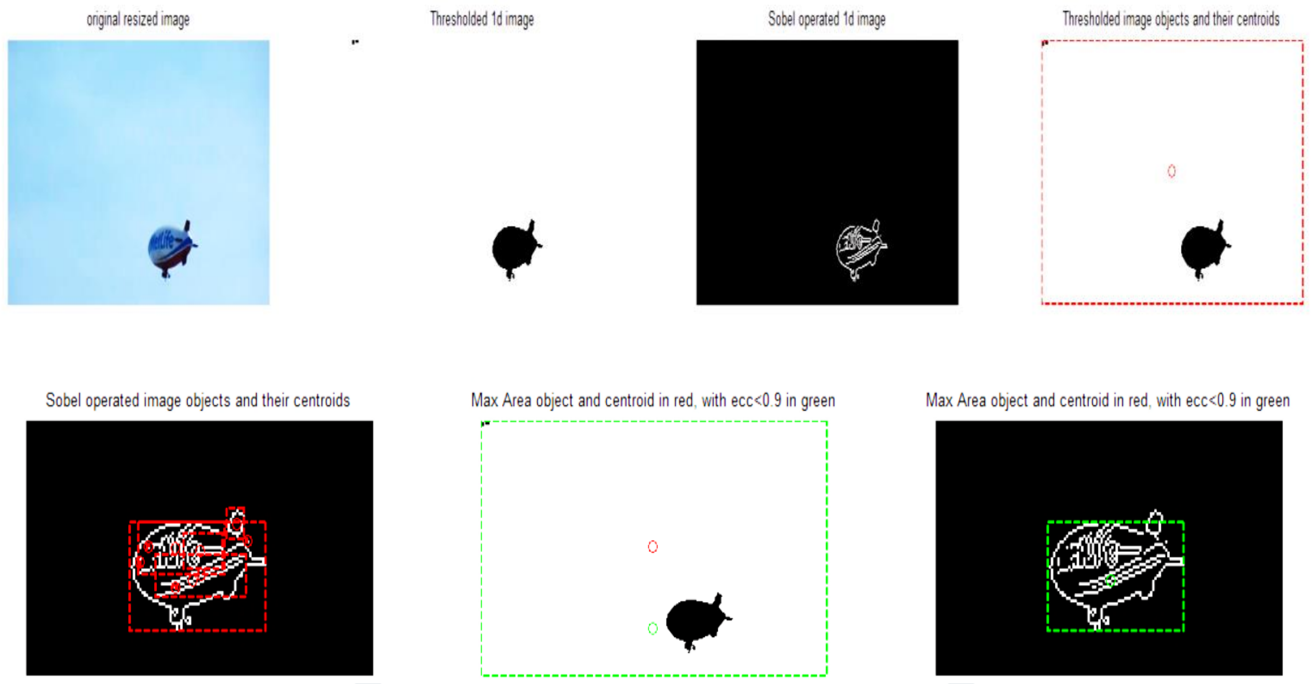

original resized image
Thresholded 1d image
Sobel operated 1d image
Thresholded image objects and their centroids
Sobel operated image objects and their centroids
Max Area object and centroid in red, with ecc<0.9 in green
Max Area object and centroid in red, with ecc<0.9 in green

Figure 3: Third set of comparison that proves the efficacy of edge detection over thresholding

IV. Serial communication between Arduino and MATLAB to facilitate servo control.

In order to communicate between the computer and controller we need to establish serial communication link between the two. The process is implemented by sending strings of data to the controller. The computer waits for acknowledgment from the controller side, i.e. until the deflection of the servo is complete. The complete algorithm is divided into four main objectives.

1. Centroid detection from set of images imitating a moving object
2. Normalized error generation proportional to velocity of the object.
3. Linear control of step size of servo using feedback of normalized error.
4. Arduino side communication for servo control

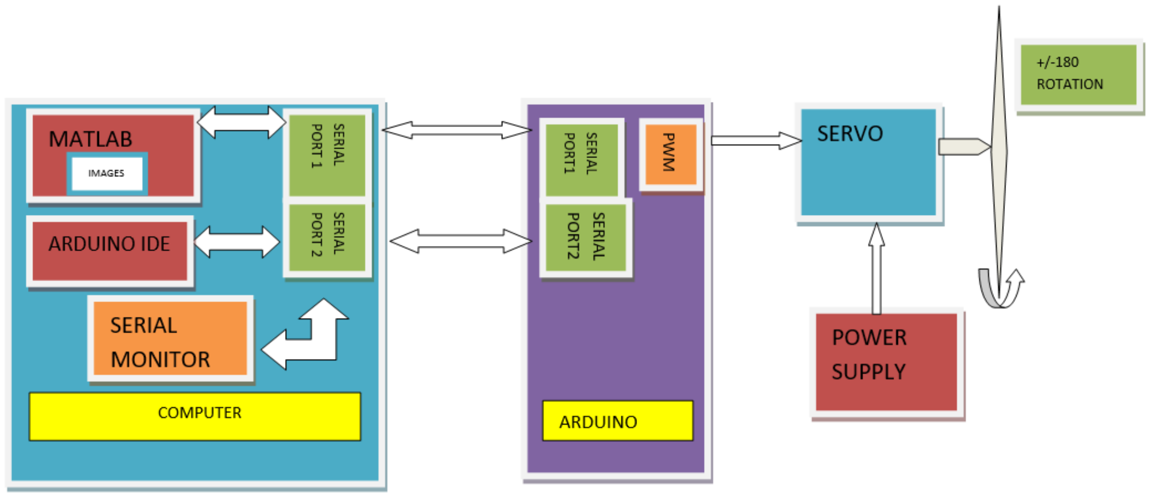


Figure 4: Schematic of communication between MATLAB and Arduino

4.1 Centroid detection from set of images imitating a moving object

This part is where the object is detected and its properties like eccentricity and area were used in order to detect the centroid. Moreover, a serial object namely “port” was set like the previous application to set up serial communication between MATLAB and Arduino microcontroller, since no real camera was deployed with the servo, pre created images were used by the MATLAB software.

4.2 Normalized error generation proportional to velocity of the object

After we detect the centroid, we need to come up with an algorithm to convert this information to the angular deviation of the servo. Now, the deviation of centroid is in pixels and it is easier to make the deviation of servo proportional to the velocity of the moving object rather than its position. This proportionality to velocity is calculated by first normalizing the deviation of centroid so that maximum deviation results in 1 and minimum deviation results in -1. The values are stored in variables "**ipx**" and "**ipy**". This way we are not being dependent on raw pixel values. Upon this normalization, we calculate the same way for the next image and calculate the difference. This difference varies from 2 to -2 and we normalize it again so that is stays in between 1 and -1. Now the absolute value of this is stored in the variable "**mag_fx**" and "**mag_fy**". This value is proportional to the velocity of the moving object, meaning a value of 1 means that the object is moving extremely fast and 0 means that the object hasn't moved with respect to the previous image.

4.3 Linear control of step size of servo using feedback of normalized error

The direction of rotation of the servo is decided by whether the current value of "**ipx**" is greater or lesser than the previous value. If greater it means "move towards positive x" and if lesser it means "move towards negative x". Similarly, the orientation is calculated for y axis where positive y is towards south. The first images have no previous reference hence their "**mag_fx**" or "**mag_fy**" is same as the absolute value of their "**ipx**" or "**ipy**".

If **mag_fx** or **mag_fy** is low it means that the object is varying slowly and if it is fast it means that the object is moving quickly. Thus, we can come up with a possible linear relationship between the **mag_fx** or **mag_fy** and the steps of servo, depending upon the ranges we choose for them

Mag_fx	Range of mag_fx	Step size of servo	Equation
Mag_fx (low)	0 to 0.33	1 to 4 degrees	steps_x = 9.09*fx+1
Mag_fx (med)	0.33 to 0.66	4 to 7 degrees	steps_x = 9.09*(fx-0.33) + 4
Mag_fx (high)	0.66 to 1	7 to 10 degrees	steps_x = 9.09*(fx-0.66) + 7

Table 4.3 Range decision and its corresponding equations

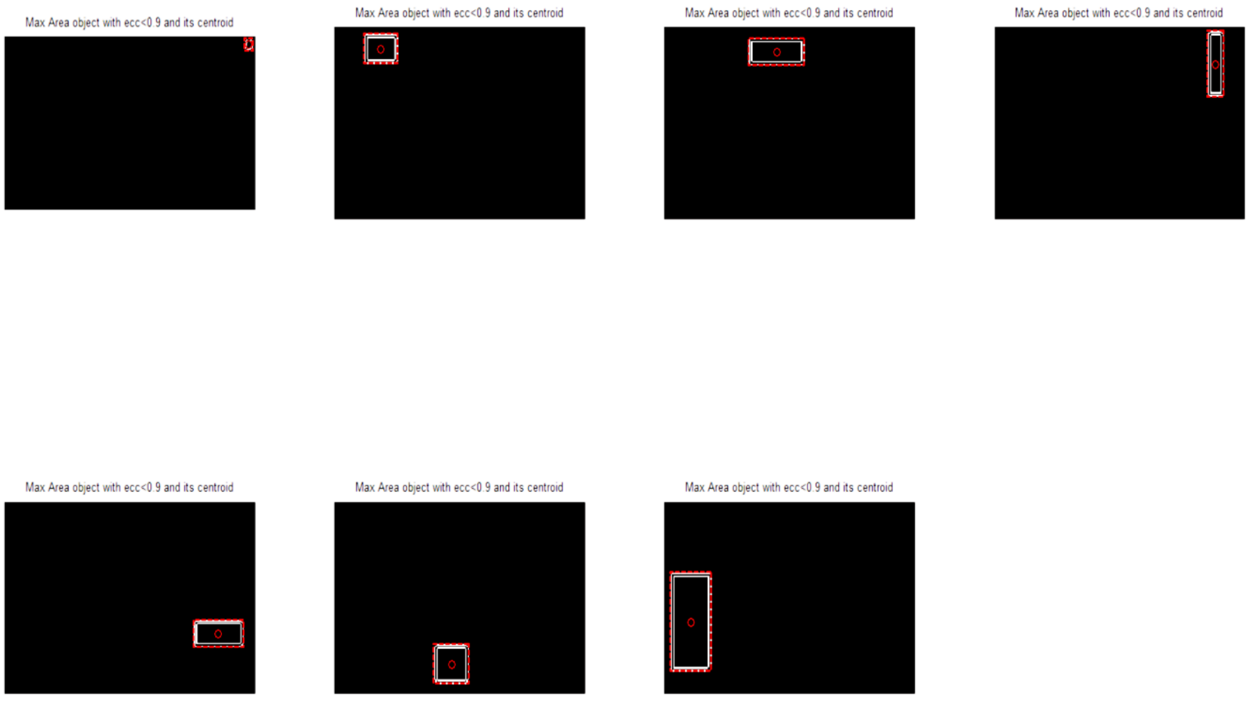
Finally the values are made into a string with "**tx**" being 1 means to toggle the direction of movement for servo in x direction. "**sx**" means steps of servo for x direction and so on for y axis servo. This string is output via the port set up earlier and the next image is fetched upon receiving "#" symbol from Arduino, indicating end of movement of servo.

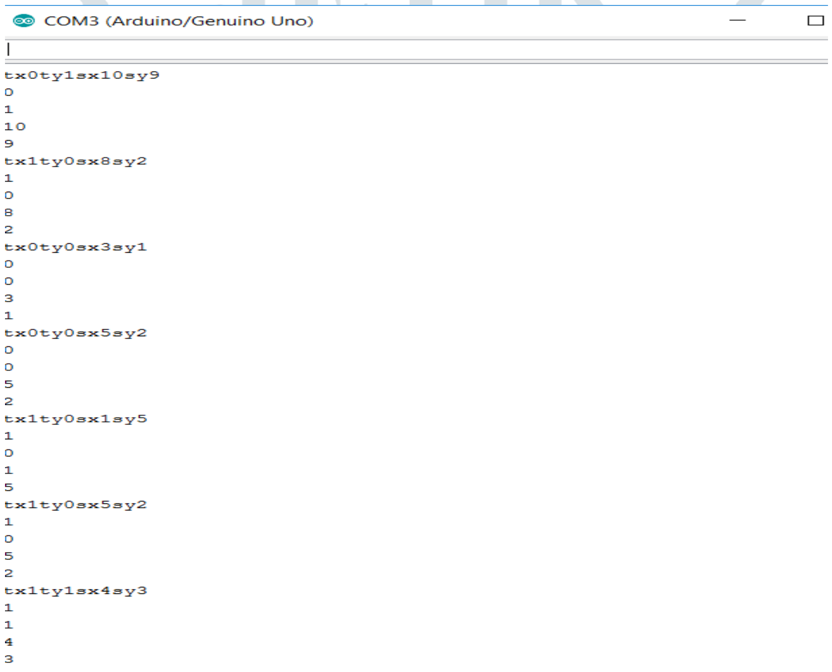
4.4 Arduino side communication for servo control

In this part we receive the strings from the serial port and extract the values and display them at the serial monitor after the strings are extracted. A virtual port is created at pins 2 and 3 for receiving the strings from the computer at 19200 baud. For testing only one servo was attached to the pin number 10 which is a pwm port in Arduino, meaning that a pwm clock is connected to that port. The initial orientation of the servo is set at 90 degrees so that there is freedom of rotation towards positive 180 and negative 180.

There are two loops in the code, one for **tx**=1 and the other for **tx** = 0, when **tx** = 0 the orientation is towards positive x and when it is 1 the orientation is towards negative 180.
Each loop runs for 1000 milliseconds and the delay for each cycle of pwm is 150 milliseconds. This value is so chosen because if the step rate is 10 degrees then for 150 milliseconds delay in 1000 milliseconds loop, we have about 6.667 cycles and that amounts to 66.67 degrees which is close to the field of view of the camera which is about 60 degrees.
In the end of cycle the symbol "#" is sent to signal the end of movement of the servo so that the computer can fetch the next image for processing. The set up was experimented using both self-created random images and real images of balloons and airships. The result of the implementation is as follows.

V. RESULTS


Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid
Max Area object with ecc<0.9 and its centroid


COM3 (Arduino/Genuino Uno)

```
tx0ty1sx10sy9
0
1
10
9
tx1ty0sx8sy2
1
0
8
2
tx0ty0sx3sy1
0
0
3
1
tx0ty0sx5sy2
0
0
5
2
tx1ty0sx1sy5
1
0
1
5
tx1ty0sx5sy2
1
0
5
2
tx1ty1sx4sy3
1
1
4
3
```

Figure 5: Series of Centroid detection in images and corresponding strings sent to controller for deflecting servo

VI. ACKNOWLEDGMENT

I wish to thank Mr. Mahboob Ahmed (Scientist D) ADRDE (DRDO) Agra, under whose guidance I worked as a researcher in the Lab between 2016 and 2019 and due to whom I was provided with useful insight and resources whenever needed.

REFERENCES

[1] Md Masood Ahmad, Multiple Face Recognition using MATLAB for Attendance Management. International Research Journal of Engineering and Technology (IRJET), Volume: 07 Issue: 04 | Apr 2020
[2] Akshay Bochare, Short Range Radar System using Arduino Uno. International Research Journal of Engineering and Technology (IRJET), Volume: 04 Issue: 10 | Oct -2017
[3] Miss. Gaikwad Sayali, Smart Wheelchair with Object Detection Using Deep Learning, International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 12 | Dec 2019
[4] https://www.arduino.cc/en/reference/servo
[5] https://in.mathworks.com/help/images/ref/regionprops.html
[6] https://circuitdigest.com/microcontroller-projects/serial-communication-between-matlab-and-arduino