2018

ECEN 5623-Real-Time Embedded Systems

Virtual Goalkeeper

Nagarjuna Reddy Kovuru
Srikant Gaikwad

Appendix

INTRODUCTION

There are many challenges involved in digital image processing such as detecting and tracking moving objects. A simple algorithm based on OpenCV can somewhat rectify and help solve at least some of the issues currently faced. This idea of object tracking in a real-time deadline situation motivated us to pursue research in this direction. There are various services that can be achieved using object tracking and we plan to implement a few of them having in mind resource constraints. The algorithm that we plan to use is the object tracking based on color. The deadline for the services is determined based on the time taken to capture image, detect the object based on color, determine the centroid of the object to track it and further align the goal keeper to intercept the object.

The idea of our project revolves around controlling the goal-keeper by tracking the object based on its color. So, when the object(ball) is moved, with the color of interest, will detect the object, find its centroid, use it to track the object and control the goal-keeper according to the object movement. While we are trying to build the basic mechanism, the bigger picture offers a variety of applications. It can be used in Scientific and Medical researches by using a Nano-camera and applying the same concept in it.

FUNCTIONAL(CAPABILITY) REQUIREMENTS:

The different parts of the project include capturing of frames, detecting the object and its centroid, displaying the position of the object, goal keeper movement and finally capturing of the video. Once the image is captured processing is done on the same to remove the noise. After processing the object is detected and the centroid of the object is object is detected to track the same and display the position of the moving object and the direction of movement. Tracking is done by drawing the line in the path of motion of the object. The centroid calculated earlier is passed on to the goal keeper to align the goal keeper along the object.

The major functional capability requirements for this project are as below,

Frame Capture:

We are capturing the image of the object by using camera and its done automatically when we execute the code. And, these images are captured at a selected frame rate so that enough time will be given for processing of the frame plus a buffer margin. This selection is done by OpenCV image capture functions.

Image Processing:

The captured image is processed to remove the noise in the image by using filter operations. And, to enable the chances of object detection eroding backgrounds are removed and the required characteristic object is isolated for better detection.

Object Detection and Centroid Calculation:

After removing the noise and eroding backgrounds of the captured image, the object detection process starts. This is done based on the provided detection algorithm. Currently, we are using "color" as a criteria of object detection. And, we can set these criteria using OpenCV object detection functions. Once the object is detected, image processing techniques (contours) are used to determine the moving object and calculate its centroid.

Position Display:

The centroid of the object detected in the previous process is displayed in the form of (x, y) coordinates. And, the movement of the centroid is taken into account to find direction of the object movement. Based on the previous coordinates in the resolution frame width for different frames, the direction of motion is determined such as whether the object is moving north, south, east or west.

Obstacle(Goal-Keeper) Movement:

The centroid coordinates calculated in the previous process are passed on to the "Obstacle Movement" process to align the goal keeper according the movement in the object. Instead of using hardware robotic goal keeper we are using a software display showing the cursor moving based on the object movement.

Video Capture:

Each frame captured by camera is stored on a disk to be displayed as video. We want this video to be played in real time with the object movement. So, we are using multicore environment in Jetson board to offload the processing to other core.

REAL-TIME REQUIREMENTS

Smart goal keeper which includes object tracking and tracking is one of the best examples of the Real-Time concept. If we consider the project, the detection of object, finding out the centroid of the object, tracking the centroid of the object, direction of movement of the object and finally aligning the goal keeper based on the movement of the object all must occur before the next frame is captured. This is kind of essence of the programming in the real-time environment.

The core of the real-time requirement are as follows:

The capturing of frames, detecting the object, finding centroid, tracking the object and aligning the goal keeper. While all the services are soft real-time implementations but the hard-real time deadline is imposed on the fact that all the services must occur before the next frame capture occurs.

FUNCTIONAL DESIGN OVERVIEW

The functional system block diagram includes Logitech C200 Camera, Object to be detected and Linux System (Jetson TK1).

Camera is used to capture the image of the object, Code is executed on the Jetson TK1 board. The object is moved in front of the camera so that to give the code frames to process.

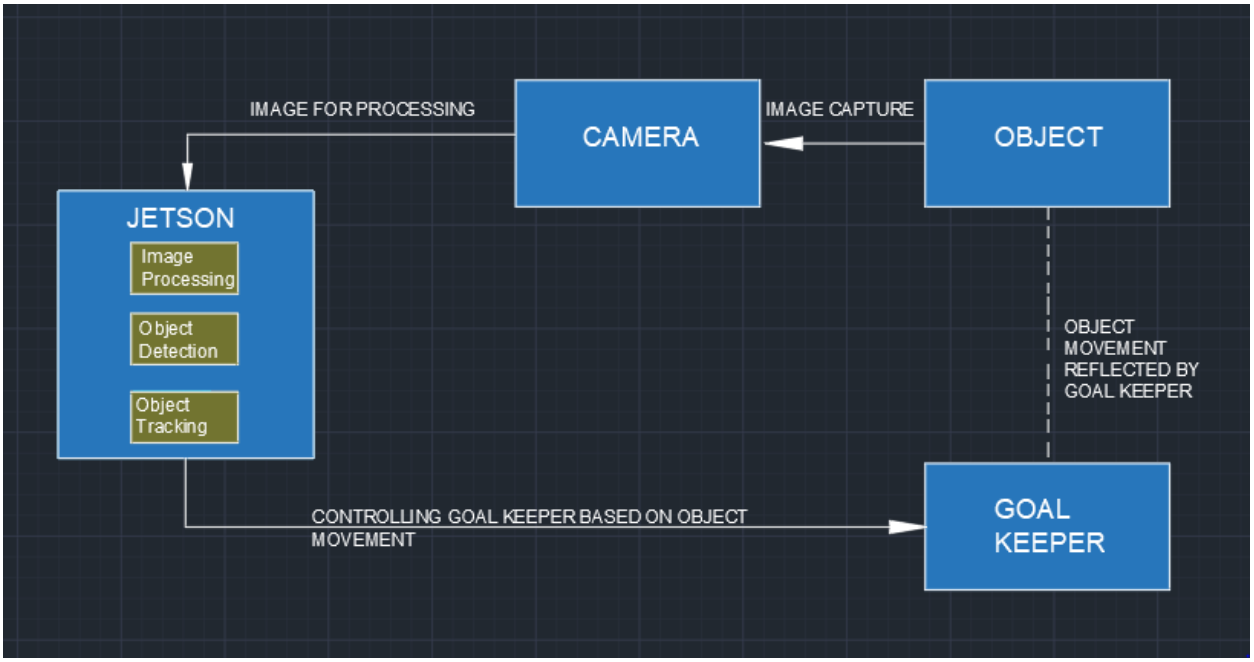


Figure1: System Block Diagram

The entire project code is divided mainly into four threads, which denotes the number of services we are using in the project. The functionalities of all the threads are divided and the roles are assigned as follows:

The below diagram is known as entity relationship diagram. In this we are combining the centroid calculation and displaying the co-ordinates in the object detection thread. There is an obstacle movement thread which include the functionalities like tracking the co-ordinates, finding the object movement and aligning the goal keeper and finally there is video capture thread which stores the video generated to the memory.

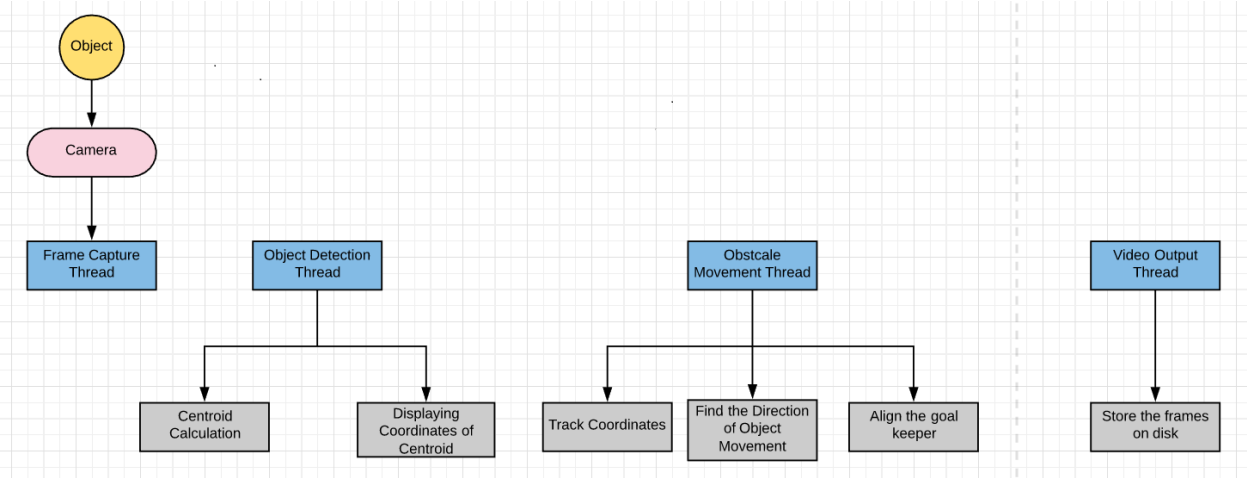
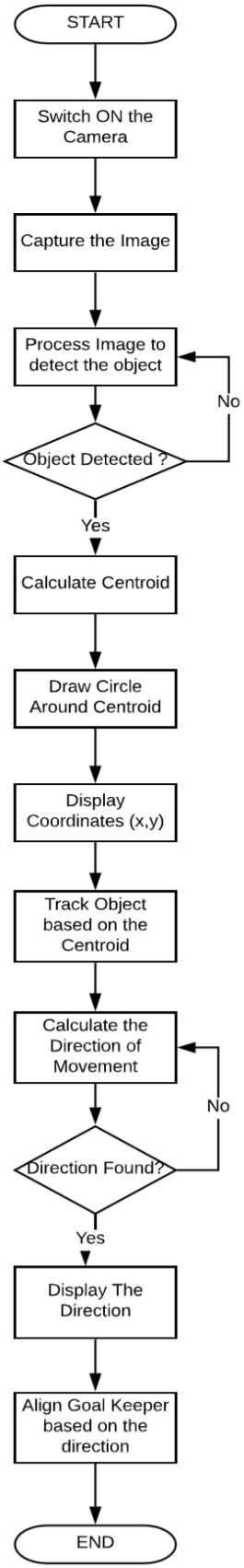


Figure2: Entity Relationship Diagram

FLOW CHART OF THE CODE:


START
Switch ON the Camera
Capture the Image
Process Image to detect the object
No
Object Detected ?
Yes
Calculate Centroid
Draw Circle Around Centroid
Display Coordinates (x,y)
Track Object based on the Centroid
Calculate the Direction of Movement
No
Direction Found?
Yes
Display The Direction
Align Goal Keeper based on the direction
END

REAL TIME ANALYSIS AND DESIGN

The analysis of the project should begin with the real-time services which are part of this project. The services are as listed below with the necessary specifications.

The parameters which are considered for the real-time analysis are services Si, Execution time Ci, deadline Di, period Ti and worst-case execution time of the services.

We used below services in this project,

1) Frame Capture: Captures image using Logitech camera. And, the timings corresponding to this service are,
   C1 = 19 ms, WCET = 25, D1 = 100 ms, T1 = 100 ms.

2) Object Detection: Process the image to determine the moving object and calculate its centroid. And, the timings corresponding to this service are,
   C1 = 29 ms, WCET = 40, D1 = 100 ms, T1 = 100 ms.

3) Obstacle Movement: Align the goalkeeper to intercept the moving object based on the centroid obtained in the previous step. And, the timings corresponding to this service are,
   C1 = 25 ms, WCET = 30, D1 = 100 ms, T1 = 100 ms.

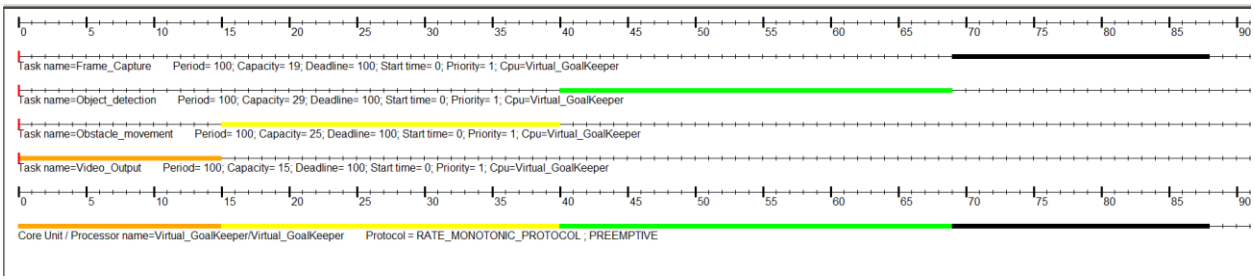
4) Video Output: Capture camera output and save it as a video on the disk. And, the timings corresponding to this service are,
   C1 = 15 ms, WCET = 20, D1 = 100 ms, T1 = 100 ms.

Time Analysis:

We executed these services in Linux Environment(Jetson) using SCHED_FIFO scheduling Algorithm. And assigned priorities in such a way that all the services complete execution before the 1st service "Frame Capture" happens again. The time periods and deadlines are calculated to accommodate all the services within the time frame. And, we calculated the worst-case execution time by considering the preemption due to the high priority services and different loop cases. As, we are using FIFO, the service deadlines and time periods are same.

All the services are executed as different threads using POSIX Library. And, we are using Mutexes and Semaphores to synchronize the services while sharing global data such as Frame captured and Centroid Coordinates.

Cheddar Analysis:



Scheduling simulation, Processor Virtual_GoalKeeper :
- Number of context switches : 3
- Number of preemptions : 0

- Task response time computed from simulation :
  Frame_Capture => 88/worst
  Object_detection => 69/worst
  Obstacle_movement => 40/worst
  Video_Output => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

WCET Analysis:

Scheduling feasibility, Processor Virtual_GoalKeeper :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 100 (see [18], page 5).
- 12 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.88000 (see [1], page 6).
- Processor utilization factor with period is 0.88000 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.88000 is equal or less than 1.00000 (see [19], page 13).

2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time : (see [2], page 3, equation 4).
  Frame_Capture => 88
  Object_detection => 69
  Obstacle_movement => 40
  Video_Output => 15
- All task deadlines will be met : the task set is schedulable.

Scheduling Point, Completion Test analysis and safety Margin Analysis:

We are running tasks sequentially as the tasks are dependent on the previous tasks. For example, obstacle task is dependent on object detection task and the object detection task is dependent on frame capture task so all tasks are interdependent and needs to be run sequentially. For doing this, we are using sequencer task which runs other tasks sequentially with certain period.

As mentioned in the above table, we are getting period nearly 100ms for all the tasks. And, all the tasks need to be completed before this time. There are no individual deadlines. As long as the sum of the execution times of individual services is less than 100 ms, it should be fine.

As we are getting new frame every 100ms, Analysis of WCET over one period should be sufficient.

Timestamp Tracing:

In the below screenshot, Ci, Ti, Di and WCET are displayed for all the tasks.

```
************Completion Test Feasibility Test*********
U=0.92 (C1=19, C2=29, C3=25 C4=15; T1=100, T2=100, T3=100, T4=100;): FEASIBLE


************Scheduling Point Feasibility Test********
U=0.92 (C1=19, C2=29, C3=25, C4=15; T1=100, T2=100, T3=100, T4=100): FEASIBLE
```

In below table, we have listed the Ci, Ti(=Di) and WCET for all the services. These timings are listed for multiple frames captured to get the average instead of a single value to get accuracy in timing.

Frame	Service	Ci(ms)	Ti(ms)	WCET(ms)
1	Frame Capture	19	100	25
	Object Detection	29	100	40
	Obstacle Movement	25	100	30
	Video Output	15	100	20
2	Frame Capture	20	100	30

	Object Detection	30	100	44
	Obstacle Movement	20	100	35
	Video Output	20	100	20
3	Frame Capture	20	100	20
	Object Detection	30	100	35
	Obstacle Movement	20	100	20
	Video Output	15	100	20

Output Showing Object Detection (Test Case):

In below figure, the object is detected and displayed the coordinates. We configured the object color as yellow. The centroid can be seen in the image.

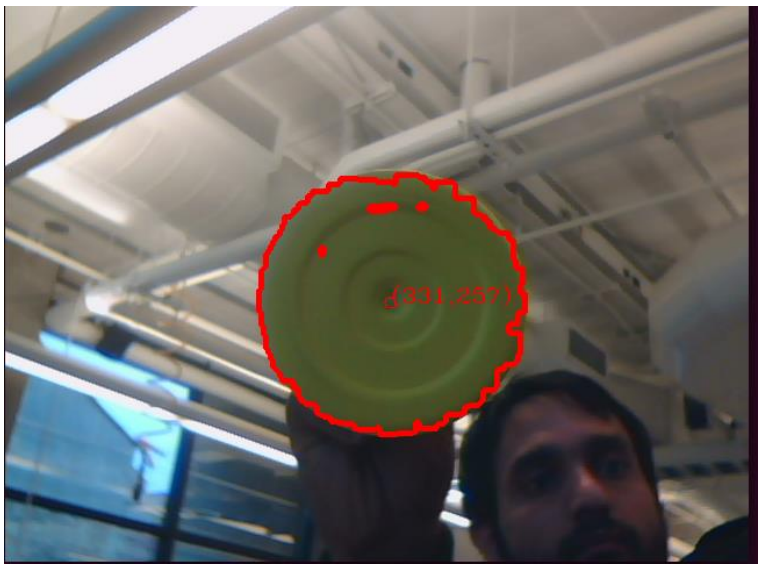


Figure3: Centroid Calculation

In the figure below, we can see that object is being detected, the path of movement is traced and the goal keeper is aligned according to the position of the object which is represented by its x-axis.

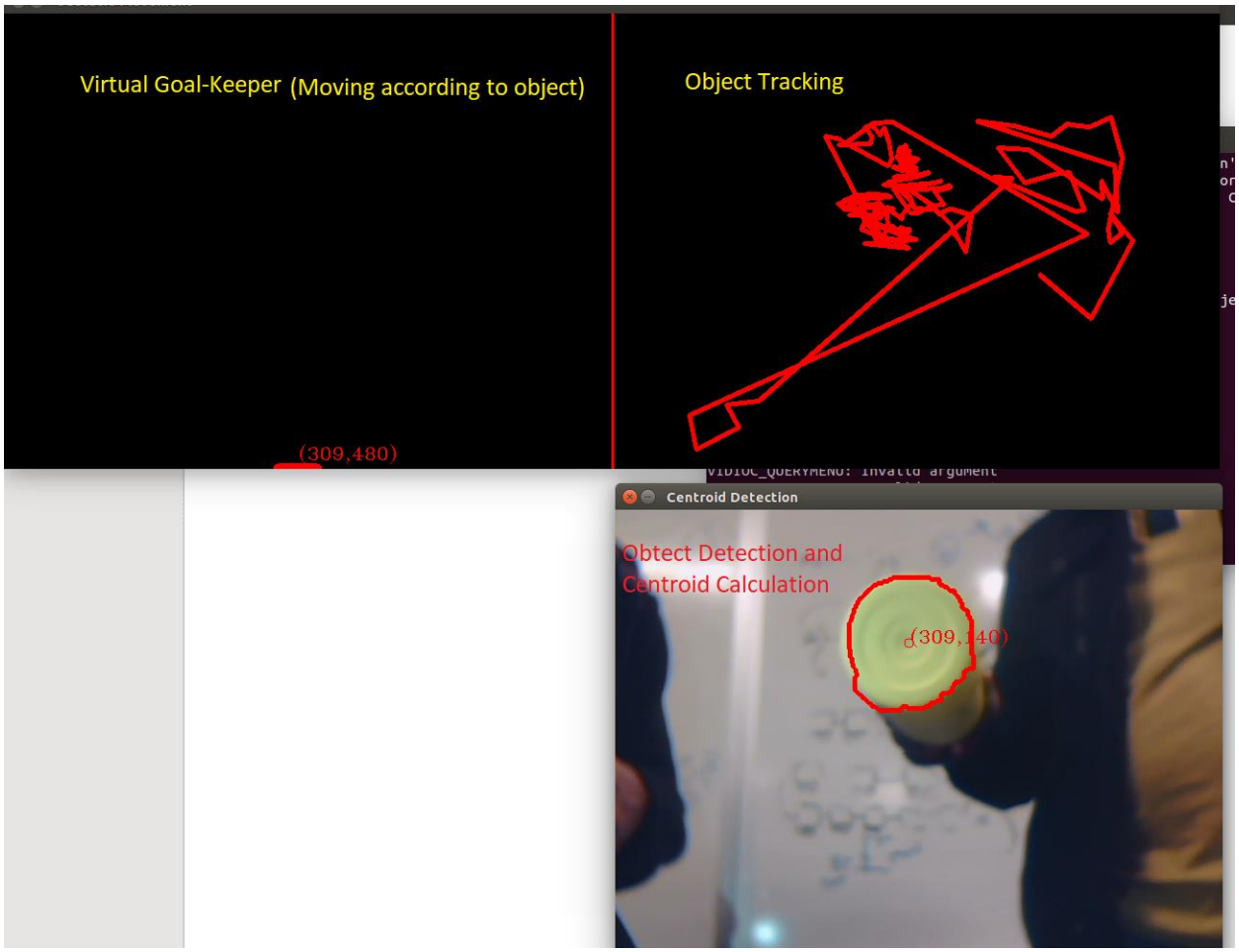


Figure 4: Output1 showing object detection, tracking and aligning goal keeper

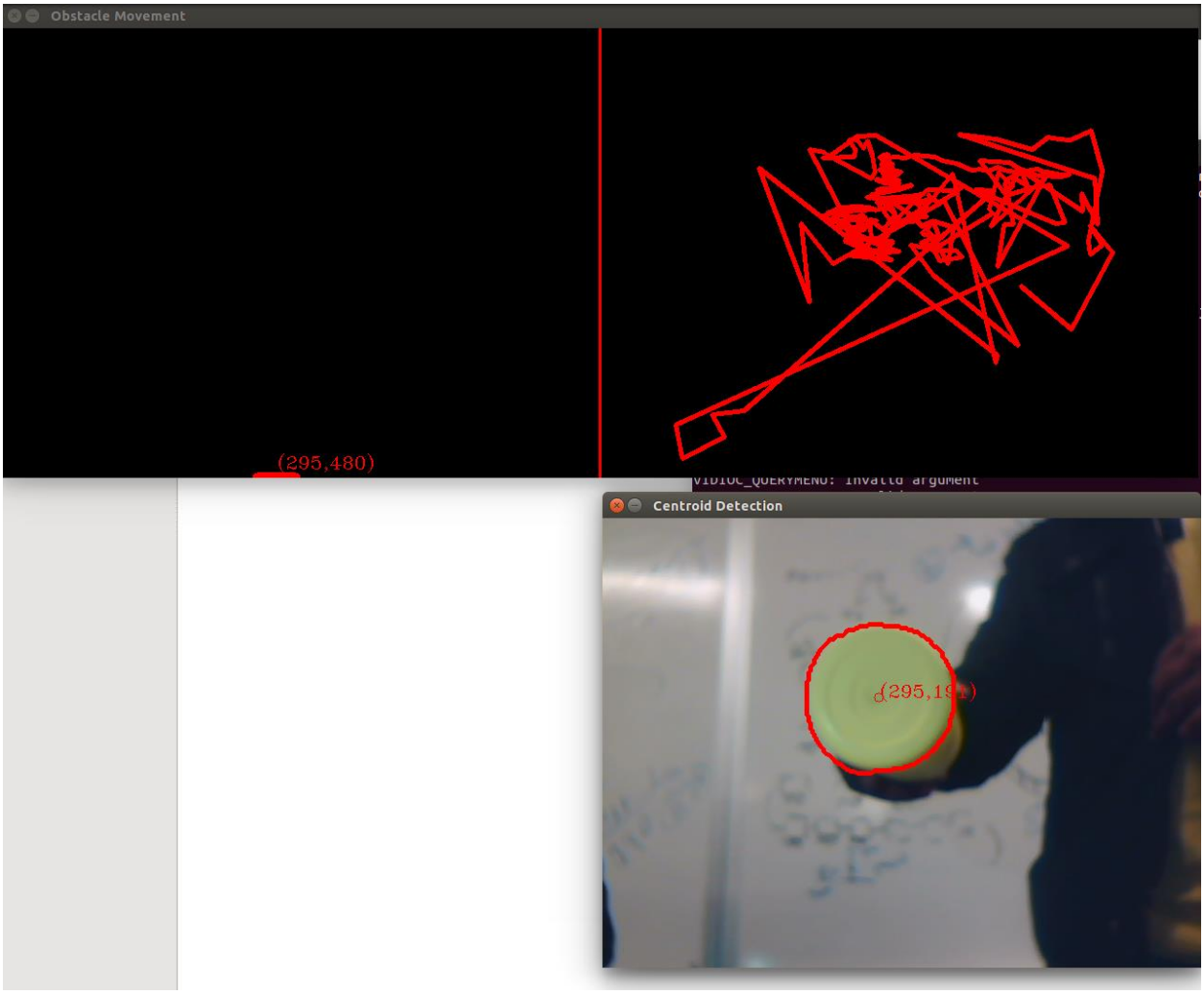


Figure 5: Output2 showing object detection, tracking and aligning goal keeper

ISSUES FACED AND LESSONS LEARNT:

- As all the tasks are interdependent, we had to maintain a sequence in the execution of the threads, so we created another thread inside which provided delays in between each task.
- As we were new to OpenCV library, had to go through the manual to find out the exact functions to be used for writing video, capturing frames and configuring object color etc..
- The execution times were different for each task, as the tasks were preempting in between. So to get exact timing we separately calculated execution times for each function by disabling other threads.
- We had to configure hue, saturation values and color accurately to get the object.
- Direction and Position were overlapping at different instances of the code execution.
- We were getting issues with camera to capture images.
- The camera also had latency issues, a few frames were captured way past their deadline.
- We had to enable multiple cores present on the Jetson board to increase efficiency and fastness so that real-time services are executed within deadline.

Conclusion:

The main objective of our the project was to detect objects of specific color and track its movement across the frame captured. And, align the goal keeper(cursor) according to the movement in the object(ball). The implementation of the same was successfully executed. We have 4 real time services and all were executed successfully within the required deadline. And, we used SCHED_FIFO scheduling policy on Jetson Board to achieve. Same priority tasks are scheduled using Round-Robin policy. Each service was implemented as a thread and the object was detected by calculating the centroid position, drawing a circle around it and tracked using the calculated centroid position. Using the past and current position data, the direction was also determined. According to the obtained direction goal-keeper is also moved to intercept the object.

Citations & References

1. ***Color Name & Hue:*** Consulted how to detect the object using color. http://www.color-blindness.com/color-name-hue/

2. ***Introduction to OpenCV Tracker:*** Consulted how to track the object using OpenCV library. https://docs.opencv.org/3.1.0/d2/d0a/tutorial_introduction_to_tracker.html

3. ***OpenCV Tutorial C++:*** Contains color detection and object tracking. https://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.html

4. ***RT_Clock:*** Professor Sam Siewart's resource to configure real-time clock. http://mercury.pr.erau.edu/~siewerts/cec450/code/RT-Clock/

Note: Code is submitted in Code_Dump.