JINAV: Journal of Information and Visualization *ISSN: 2746-1440 (online)*
Vol. 1 No. 1 (2020)
https://doi.org/10.35877/454RI.jinav155

Machine Vision Enabled Bot for Object Tracking

Rupali Patil*, Adhish Velingkar, Mohammad Nomaan Parmar, Shubham Khandhar, & Bhavin Prajapati

KJ Somaiya College of Engineering, Mumbai, 400077, India

Abstract

Object detection and tracking are essential and testing undertaking in numerous PC vision appliances. To distinguish the object first find a way to accumulate information. In this design robot can distinguish the item and track it just as it can turn left and right position and afterward push ahead and in reverse contingent on the object motion. It keeps up the consistent separation between the item and the robot. We have designed a webpage which is used to display live feed from the camera and the camera can be controlled by the user efficiently. Implementation of machine learning is done for detection purpose along with open cv and creating a cloud storage. Pan-tilt mechanism is used for camera controlling which is attached to our 3-wheel chassis robot through servo motors. This idea can be used for surveillance purpose, monitoring local stuff and human machine interaction.

Keywords: Raspberry-Pi; Pan-Tilt; Tracking; Tensor Flow; Open CV; Webpage..

1. Introduction

Robot vision refers to the potential of a mechanism to visually understand the atmosphere and use this data to execute completely different tasks. Visual feedback has been used extensively for mechanism navigation and obstacle dodging. In recent years, there have conjointly been examples that embody interaction with folks and manipulation of objects. Vision in artificial intelligence review aspects of mechanism vision from its early beginnings to the foremost recent analysis. Vision in artificial intelligence is a perfect and current reference for researchers and professionals operating within the fields of machine vision, image process and pattern recognition. Recent advancements in the field of robotics along with the mechanized frameworks have prompted the development of self-governing capacities and foundation of clever machines being used in perpetually fluctuated applications (Lopes & Santos-Victor, 2005; Ju, et.al, 2014).The capacity of adjusting to changing conditions is a fundamental ledge for task summed up robots (Konidaris, et.al., 2012; Kober & Peters, 2010). AI, which is a subset of Man-made consciousness (artificial intelligence), assumes a key job in making such broadly useful automated arrangements. AI is a utilization of information investigation to foresee conduct of framework dependent on existing information. In any case, most robots are as yet produced for reason or deliberately, in view of master information and prerequisite of the application foundation. Even though this is viewed as a compelling strategy, it is a troublesome and additional tedious methodology, which has critical impediments for summed up materialness.

Object discovery is breaking into a wide scope of ventures, with use cases extending from individual security to profitability in the work environment. Article identification and following is applied in numerous zones of PC vision, including picture recovery, security, observation, computerized vehicle frameworks and machine examination. Huge

* Corresponding author:
E-mail address: rupalipatil@somaiya.edu (Rupali Patil)

difficulties remain on the field of item identification. The conceivable outcomes are inestimable with regards to future use cases for object discovery. And its live tracking feature would be an added boost to the object detection. To deal with the robot framework the accompanying hardware was chosen: Raspberry-Pi, to control the entire process of detection-tracking and actuators of robot having acted like the cerebrum of the framework; shield driver to control the DC engines; Pan-Tilt mechanism for camera attachment with two servo motors accordingly; Battery module for powering purpose.

2. Methodology

The main aim of our work is to build a Raspberry-Pi based bot integrating with the web interface that would be responsible for detection and tracking of an object. The webpage would handle the manual and automatic control of our bot. Basically the buttons created on the webpage would pass processed command to Raspberry-Pi and PC to make the bot move and track in accordance with the help of machine learning and open cv. Developing the bot is the thing that starts things out in the pipeline of our task which would inevitably be following items. For our work, we expected to control two dc motors, two servo motors and react to remote commands through the webpage created all at the same time.

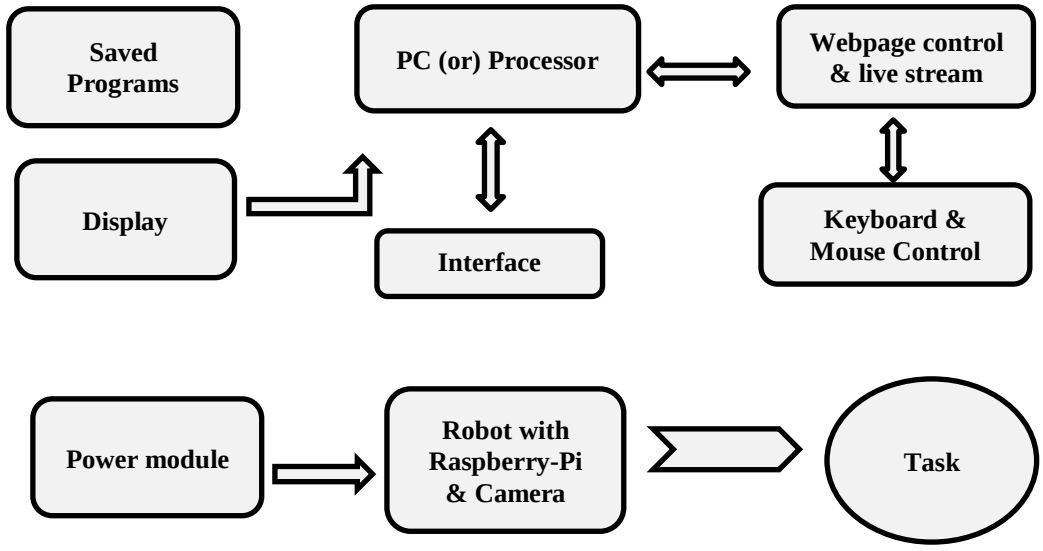


Fig. 1. System block diagram. The given object detection task can be completed using robot, Raspberry-Pi and camera. The robot is remotely controlled using webpage

As we can see from figure 1 that workflow is pretty much clear and since our title looks like the name itself as machine vision which is a sensor utilized in the robots for survey and perceiving an article with the assistance of a PC. It has a few parts, for example, camera, PC, digitizing equipment, and an interface between equipment and programming.

2.1 Construction and assembly

We have used Raspberry-Pi which comprises of a solitary board PC which was developed by the RPI foundation. The first Pi had comprised of single center 700MHz CPU and only 256MB RAM, and the most recent model has a quad-center 1.4GHz CPU with 4GB RAM which is the cerebrum of our bot which would settle on choices in controlling the wheels to follow/following as shown in figure 2. The board is power sourced with an alternate battery module.

The Raspberry-Pi PC that runs Linux, additionally gives a lot of GPIO (broadly useful information/yield) sticks that permitted us to control electronic segments for incorporating the rest of the parts. The Raspberry-Pi board cannot straightforwardly control the motors of the bot, as it includes specifically running forward or in reverse that requires trading force and ground inputs. We utilized a specific circuit called a H-Extension which is required to get this going and we have doled out this undertaking to an engine driver that is essentially a momentum enhancer which takes a low-ebb and flow signal from the controller and gives out a relatively higher ebb and flow signal which can control and drive an engine.

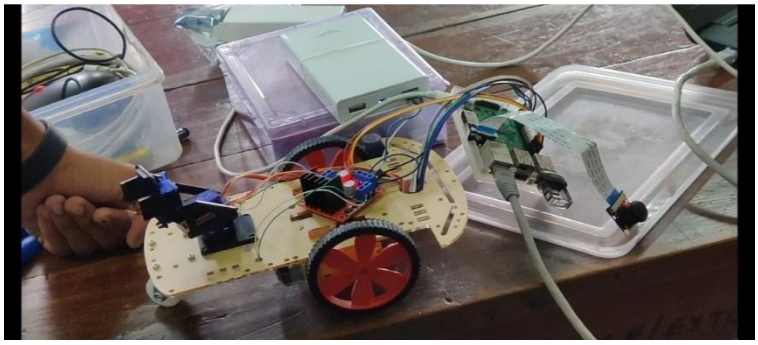


Fig. 2 Temporary assembly of the bot

All the needs of mounting the camera as an imaging sensor was to operate the pan and tilt mechanism. The whole assembled setup of the pan and tilt module has two axes movement using two 9g servo motors. These servos can be easily controlled using any Raspberry-Pi. There is a kit for Pan-Tilt mechanism designed specifically for Pi Camera which is easily available. Raspberry Pi Camera used here is of 8MP.It provides resolution of 3280x2464 pixels. Some of its features are – supports frame rates up to 90fps, picture formats are jpeg, gif, bmp, jpeg+raw, has Fisheye lens which offers wider field of view. Flask is been used here as a webpage support framework for Raspberry-Pi. Flask is a smaller scale web system written in Python. It is delegated a micro framework on the grounds that it does not require instruments or libraries. It has no database reflection layer, structure approval, or whatever other segments where prior outsider libraries give regular capacities.

2.2 Control Mechanism

The project design comprises of two parts, software part: Object Detection with Raspberry-Pi which is one of the main objectives is done here along with the development of Webpage and hardware part: This includes designing of the robot, movement of the robot, camera movement while continuously tracking the object, and its interfacing with the Raspberry-Pi. Figure 3 and 4 shows the flowchart for object detection and bot control respectively.

As we can see from figure 5 that webpage control of our bot for its pan-tilt mechanism (camera control) as well as the dc motors (bot control) later we have also added two additional buttons namely manual and automatic which can be used accordingly by the user. Once manual button is pressed automatic button will be disabled and thus the user can control through the bot through the keyboard manually for moving the bot. After that if automatic button is pressed manual button will be disabled and thus the bot will perform the task of object tracking with the help of buttons related to ML on the webpage created.

2.3 Object classifier using Machine Learning (ML)

For the machine learning part, we used Tensor Flow which is a free software library created by Google. Tensor Flow derives its name from the multidimensional arrays known as arrays, which are used by the neutral networks for different operations. Tensor Flow works by figuring out layers of information (otherwise called hubs) as a component of learning. In the primary layer, the framework decides the essential highlights of the item. It gives clients

increasingly important data as the arranging of the pictures is done at the quicker rates. The calculation in Tensor Flow are accounted for as stateful dataflow diagrams. Additionally an open source PC vision and AI programming library is utilized by us which is OpenCV with the assistance of its calculation can be utilized to distinguish and perceive faces, track camera minutes, track moving items, separate 3D models of articles, produce 3D point mists from sound system cameras, recognize objects, and so on.For object detection part we have prepared our own convolutional neural system object location classifier from scratch. Basically, we installed TensorFlow v1.5 with the necessary packages required for it and installed python version 3.6.8 and its libraries which are needed while training and testing our data. The TensorFlow object location programming interface requires utilizing the catalog structure. It likewise requires a few extra python bundles, explicit increments to the way and pythonpath factors, and a couple of additional means, orders to get everything set up to run or train an item identification model.

We have downloaded the full TensorFlow object identification storehouse from github.com/TensorFlow/models. TensorFlow gives object identification models (pre-prepared classifiers with explicit neural system designs). A few models, (for example, the SSD-Portable Net model) have an engineering that takes into account quicker recognition however with less exactness, while a few models, (for example, the Quicker RCNN model) give more slow location yet with more precision. We are utilizing SSD light model for our location part. We made a folder straightforwardly in C: and name it "tensorflow1". This archive which we have downloaded contains the pictures, explanation information, .csv records, and TFRecords expected to prepare an object finder. We utilized these pictures and information to work on making your own Article Locator. It additionally contains Python contents that are utilized to create the preparation information. It has contents to try out the article location classifier on pictures, recordings, or a webcam feed. After this we proceeded with the necessary steps for configuration and setting up virtual environment variables for python for TensorFlow Object Detection API.

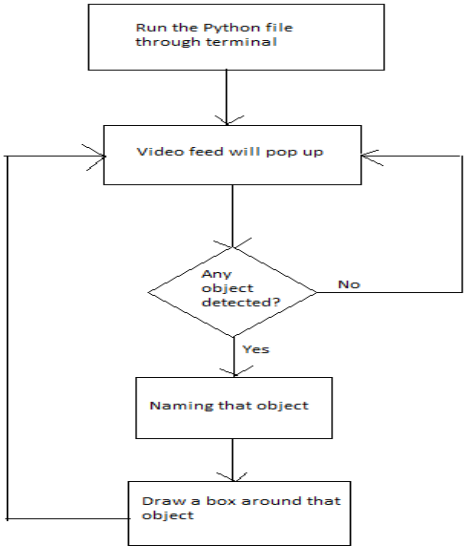


Fig. 3. Flowchart for object detection

2.4 Creating, labelling and training of dataset

TensorFlow requires many pictures of an item to prepare a decent identification classifier. To prepare a powerful classifier, the preparation images ought to have irregular items in the picture alongside the ideal object and ought to have a different background and lighting conditions. There ought to be a few pictures where the ideal object is mostly darkened, covered with something different, or just most of the way in the image. For this we made the dataset of two

classes viz. KJSCE_BALL and KJSCE_BOTTLE. We utilized our versatile to take around 40 photos of each item all alone, with different other non-wanted articles in the photos. One can likewise download the pictures of the pictures from Google Picture Search.

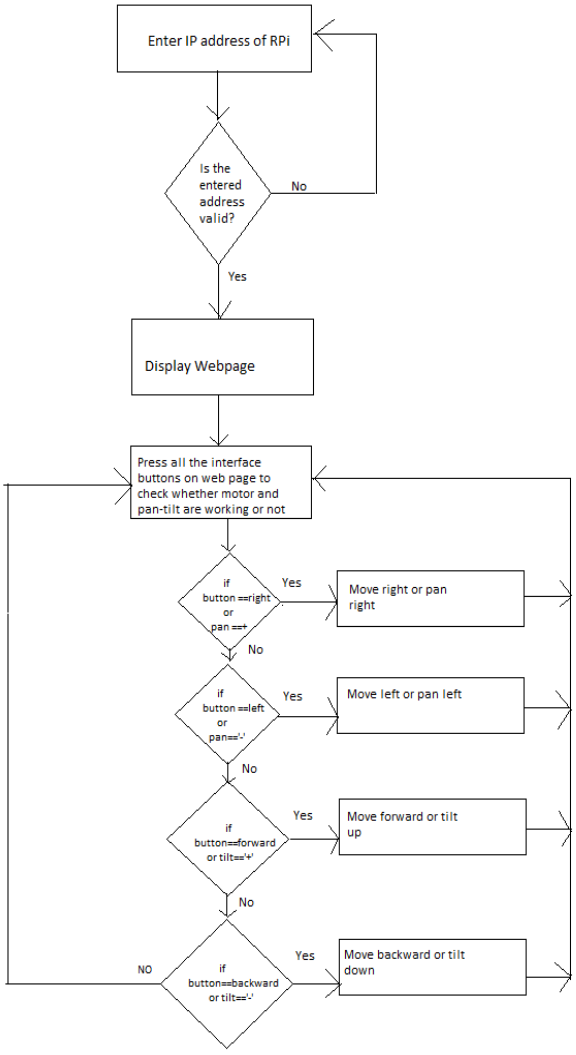


Fig. 4. Flowchart of Bot control

We should ensure the pictures are not very huge. They ought to be under 200KB each, and their goals ought not be more than 720x1280. Sample dataset images are shown in figure 6. The bigger the pictures are, the more it will take to prepare the classifier. In the wake of having the photos, we moved 20% of them to the \object_detection\images\test registry, and 80% of them to the \object_detection\images\train index.

For this we installed an image annotations tool called as LabelImg as can be seen from figure 7. After installing LabelImg, we should cross through \images\train registry, and afterward attract a box around each object each picture. LabelImg spares a .xml record containing the name information for each picture. These .xml documents will be utilized to create TFRecords, which are one of the contributions to the TensorFlow coach.

Fig. 5. Webpage control for bot

Fig. 6 Dataset

With the images labeled, we must generate the TF Records that serve as input data to the TensorFlow training model. Here we used a python script to convert all the .XML files to .CSV files which is present in our object detection folder. This will create an excel sheet of both test.csv and train.csv files in image folder.

With the images marked, we need to produce the TF Records that fill in as info information to the TensorFlow preparing model. Here we utilized a python content to change over all the .XML records to .CSV documents which is available in our article identification organizer. This will make an exceed expectations sheet of both test.csv and train.csv documents in picture organizer. The label map mentions to the mentor what each article is by characterizing a mapping of class names to class ID numbers. When the label map document is made in the wake of appointing explicit id numbers the object identification preparing pipeline must be designed. It characterizes which model and what parameters will be utilized for preparing which is the last advance before running preparing.

3. Results and discussion

Prior to beginning the principle programming, we tried every one of our segments interfacing with Raspberry-pi with

test sketches to confirm everything works right. Our main objective was to perform object detection and then perform its tracking. Firstly, we tested our bot with Pan-Tilt control for object detection based on Open-CV and its tracking by the camera based entirely on the color. One thing that we attempted to achieve in this was the identification and following of a specific shading object. For that, we should comprehend somewhat increasingly about how OpenCV decipher hues.

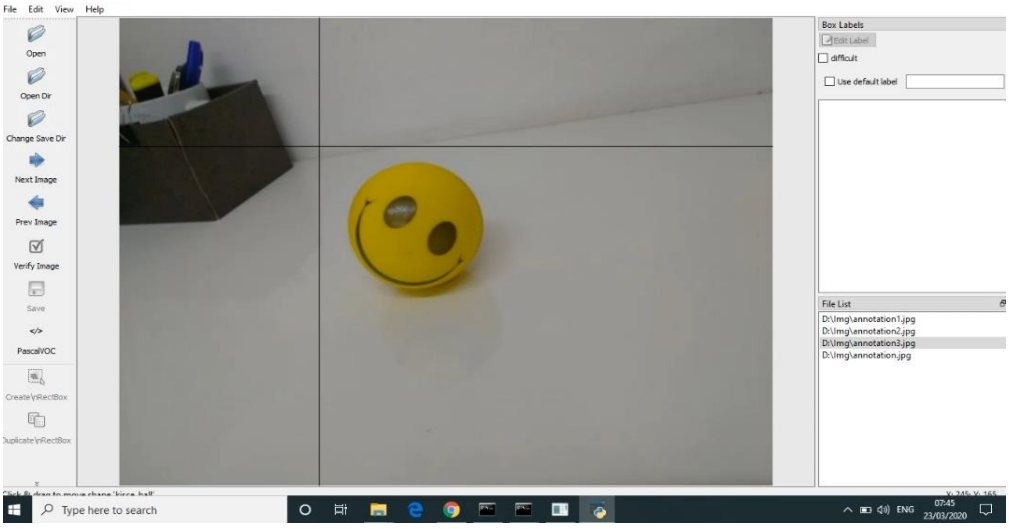
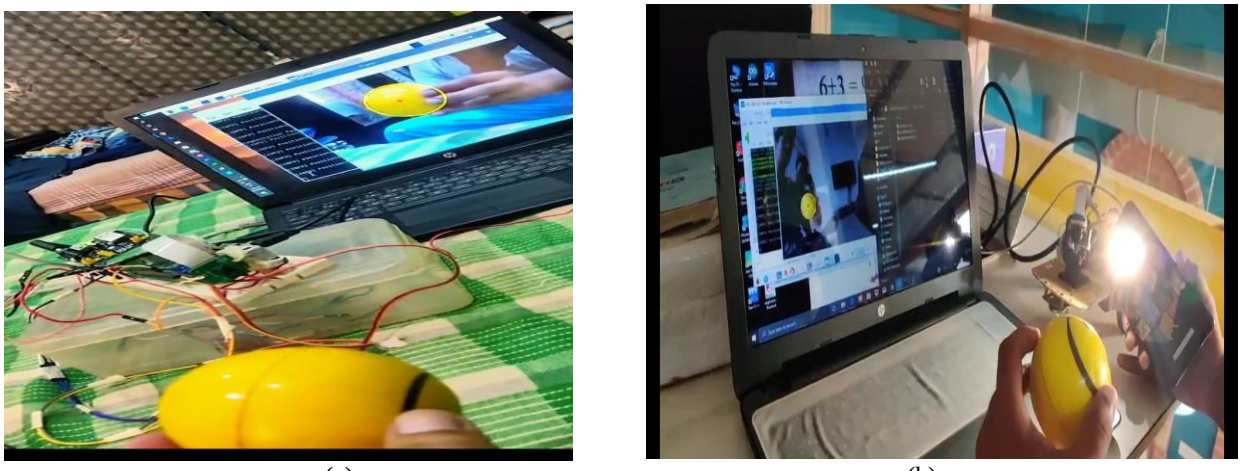


Fig. 7. Labelling the data

As a rule, our camera works with RGB shading mode, which can be comprehended by considering everything potential hues that can be produced using three hued lights for red, green, and blue. Regularly a pixel is spoken to by 3 parameters, blue, green, and red. Every parameter as a rule has an incentive from 0 – 255 (or O to FF in hexadecimal). For instance, an unadulterated blue pixel on your PC screen would have a B estimation of 255, a G estimation of 0, and a R estimation of 0. OpenCV works with HSV (Hue, Saturation, Value) shading model, that it is an elective portrayal of the RGB shading model, structured during the 1970s by PC illustrations specialists to all the more intently line up with the manner in which human vision sees shading making qualities. After the conversion of bgr to hsv model we wrote a code accordingly and tried to identify our object which is nothing but a ball of yellow color. After testing the detection part, we did the testing for tracking part through Open CV itself. The python content we made had the alternative to 1 perceive the proximity of the shaded ball 2 track and draw the circumstance of the ball as it moved around the screen Then after this the idea was to situate the item in the center of the screen utilizing the Pan/Tilt component, we made a content on it as needs be in which we can discover the object and draw a hover on it with a red speck in its inside. Subsequent to mapping the object position we took a shot at the (x, y) directions of our object as for pan-tilt system.

Once knowing the starting coordinates for pan-tilt tracking it became easier for us to have a certain co-ordinate positioning system and also we wanted to have our object to the center of our system in order to detect and track the center which we had made it earlier (red dot). Once this all was done, we then created a function for pan-tilt mapping with respect to object within the (x, y) boundaries by predefining some values. And then tested it as shown in figure 8(a) and (b).



(a) (b)

Fig. 8(a) Object co-ordinate positioning, (b) Object Detection with OpenCV

The above task we did just to get a clear idea of tracking with respect to our pan-tilt mechanism. On testing it we got to understand the behavior of servo control mechanism which was not as per our expectations. While tracking we got to face some issues in our bot such as jitters (continuously wobble) in our pan-tilt control which was not favorable. So, we decided to add a control part to our pan-tilt mechanism to make it steady.

For this we included the concept of control system-PID by tuning its parameters to certain values to avoid the jitter issue. We set the parameter values of Kp, Ki and Kd accordingly while running the object tracking function simultaneously. As per figure 9, the PID controller figures a mistake term (the contrast between wanted set point and sensor perusing) and has an objective of making up for the blunder. To put it plainly, we can sum up it as P – relative, present (large adjustments) I – integral, "in the previous" (authentic); D – derivative, hosing (envisions what's to come).After solving the issue of jitter, we were able to get quite good results from it.

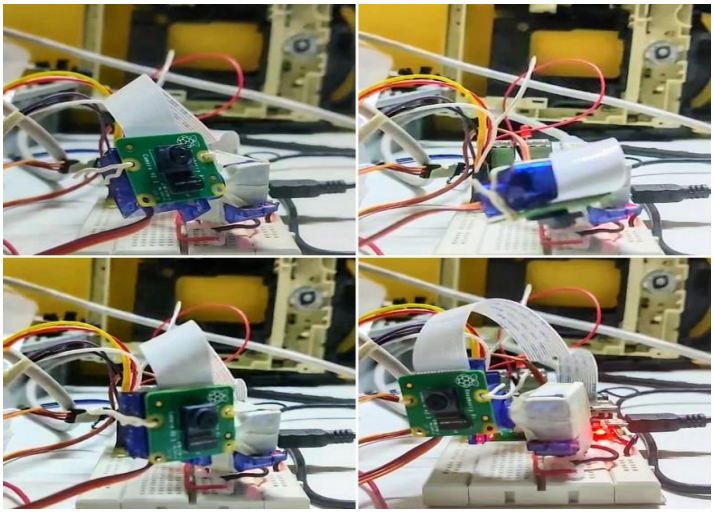


Fig. 9. Testing of PID loop

Lettering and symbols should be clearly defined either in the caption or in a legend provided as part of the figure.

Whenever possible, figures should be placed at the top or bottom of a page, as close as possible to the first reference to them in the paper.

3.1 Training with dataset

TensorFlow will introduce the preparation. The instatement can take as long as 30 seconds before the real preparing starts. Each progression of preparing reports the misfortune. It will begin high and get lower and lower as preparing advances. The preparation routine intermittently spares checkpoints about like clockwork. We can end the preparation by squeezing Ctrl+C while in the order brief window.
When preparing starts, it will resemble this: Each progression of preparing reports the loss as shown in figure 10. It will begin high and get lower and lower as preparing advances. The loss numbers will be unique if an alternate model is utilized, also the number of steps and time required for training will be dependent upon how powerful our CPU and GPU is.

```
INFO:tensorflow:Starting Session.
INFO:tensorflow:Saving checkpoint to path training/model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 2.6708 (5.383 sec/step)
INFO:tensorflow:global step 2: loss = 3.0352 (0.251 sec/step)
INFO:tensorflow:global step 3: loss = 3.4884 (0.204 sec/step)
INFO:tensorflow:global step 4: loss = 2.9733 (0.193 sec/step)
INFO:tensorflow:global step 5: loss = 2.2184 (0.191 sec/step)
INFO:tensorflow:global step 6: loss = 2.0321 (0.554 sec/step)
INFO:tensorflow:global step 7: loss = 2.0424 (0.211 sec/step)
INFO:tensorflow:global step 8: loss = 2.0252 (0.208 sec/step)
INFO:tensorflow:global step 9: loss = 2.0053 (0.194 sec/step)
INFO:tensorflow:global step 10: loss = 1.3622 (0.193 sec/step)
INFO:tensorflow:global step 11: loss = 1.8027 (0.197 sec/step)
INFO:tensorflow:global step 12: loss = 1.2485 (0.196 sec/step)
INFO:tensorflow:global step 13: loss = 1.0712 (0.193 sec/step)
INFO:tensorflow:global step 14: loss = 1.6604 (0.189 sec/step)
INFO:tensorflow:global step 15: loss = 1.2657 (0.192 sec/step)
INFO:tensorflow:global step 16: loss = 1.4351 (0.193 sec/step)
INFO:tensorflow:global step 17: loss = 1.2152 (0.192 sec/step)
```

Fig. 10. Estimating loss

3.2 Inference graph

Once the training is complete, the last step is to come up with the inference graph. This inference graph is used for our detection part through the cloud storage part which we have created between the Raspberry-Pi, Webpage, and pc.

Once the inference graph was generated, we used Dropbox platform as a part of our cloud storage which is used to send the data to our bot via webpage. We also had to generate an access key provided by Dropbox API to access that account which we have created. Two scripts were written for the purpose. One script was to upload a file from a local computer to cloud and another script to download the uploaded file from cloud to Raspberry Pi. Figure 11 shows detection of two objects, ball and bottle.

The webpage control we created is shown in figure 12. Here, there is an attempt from us to have a user-friendly experience and that the user can remotely access it from anywhere provided having a good internet connectivity. As we can see in the above figure, we tried incorporating each and every module which was tested earlier such as camera control, bot control and object detection & tracking into one by merging it through using HTML, CSS, python-flask interface. Camera control and bot control are easy to understand as it will involve manual handling of bot and camera movement. Object Detection and tracking control works entirely on the algorithm as explained before.

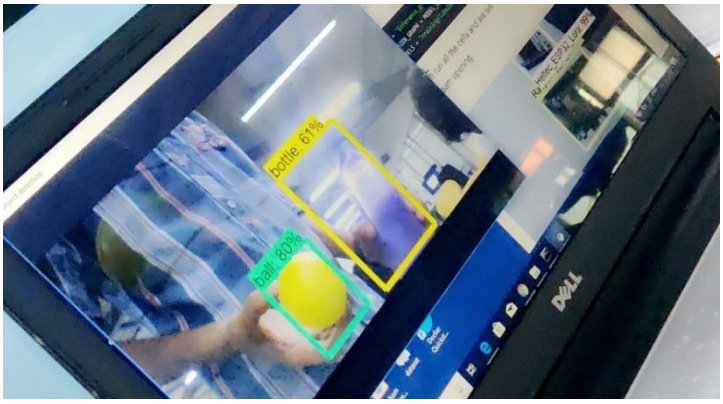


Fig. 1l. Object Detection of our two objects ball and bottle

3.3 *Webpage control*

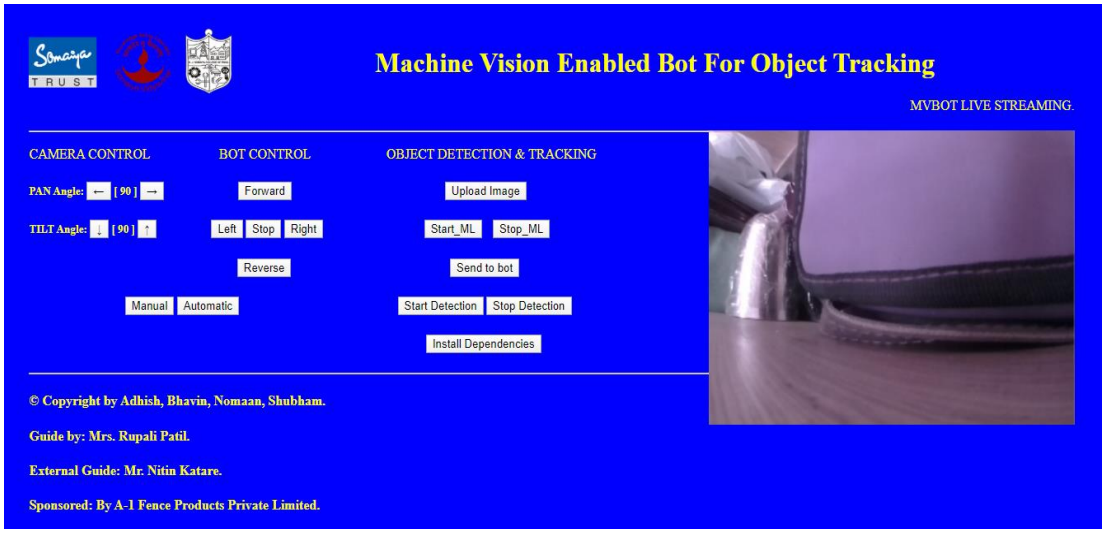


Fig. 12. Complete Controlling interface with live stream

We also added a part of user control for either manual or automatic control which functions in an independent manner. It works like if we click on manual button then the buttons related to ML part would be disabled else if we click on automatic button, rest buttons other than ML part would be disabled.

3.4 Hardware implementation

As we can see that once recognizing the item it attempts to tail it likewise subsequently, we can say that we have accomplished the objective as shown in figure 13. But we can still work on its accuracy for detection by testing it with another model. Also in figure 14 the green box created around the object is nothing but our object detection algorithm part and the red dot for its tracking part.

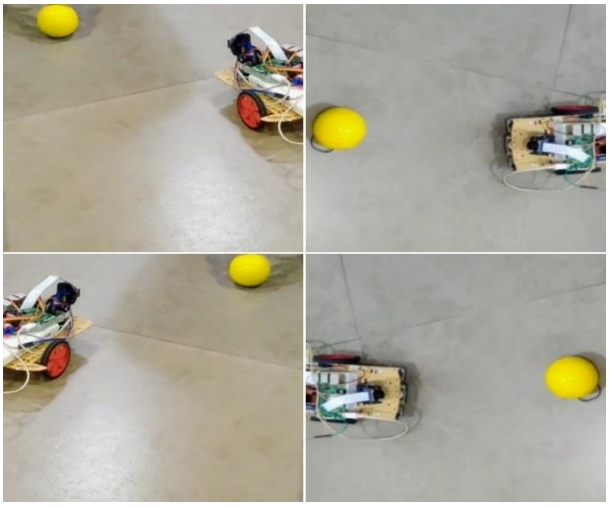


Fig. 13. Object tracking after detection

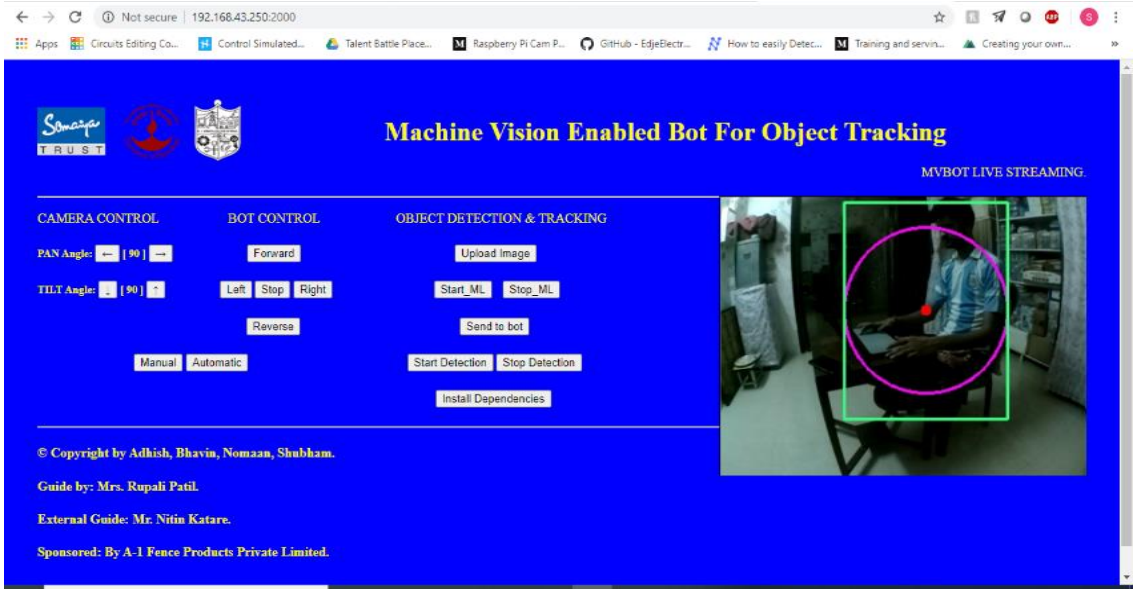


Fig. 14. Live Object Detection & Tracking on webpage

4. Conclusions

Recognizing and following individuals is getting progressively significant in automated applications due to the expanding interest for community work in which individuals associate intimately with and in a similar workspace as robots. Since our project is related to robotics vision processing, we got to learn an ample amount of stuff of how the technology is booming such as robots see, analyze, and make decisions more like humans every day, thanks to advances in converging technologies like Machine Learning(ML), and Computer Vision(CV).

We learnt that developing such visual analysis logic not only involves implementing solutions that can determine the

orientation of objects, deal with moving objects, and perform navigation but also includes preprocessing data collected from the real world via sensors, motor, etc. to get it into a more usable state by various subsystems and performing feature detection to extract visual features from the data such as corners, edges, etc. When these frameworks are set up we can proceed onward to more elevated level mechanical vision usefulness specifically object identification and arrangement and object following. Addition to this working on the webpage side gave us an idea of how to make a user-friendly setup which can be access from anywhere.

References

Lopes, M., & Santos-Victor, J. (2005). Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *35*(3), 438-449.

Ju, Z., Yang, C., Li, Z., Cheng, L., & Ma, H. (2014, September). Teleoperation of humanoid baxter robot using haptic feedback. In *2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)* (pp. 1-6). IEEE.

Konidaris, G., Kuindersma, S., Grupen, R., & Barto, A. (2012). Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, *31*(3), 360-375.

Kober, J., & Peters, J. (2010). Imitation and reinforcement learning. *IEEE Robotics & Automation Magazine*, *17*(2), 55-62.