CATCH PRACTICE BOT

Project Report

EKLAVYA MENTORSHIP PROGRAMME

At

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI

JUNE 2020

ACKNOWLEDGMENT

It was great working with fantastic seniors. We never imagined we could give something shape of which we had almost no idea of. But our collective effort and constant guidance from the seniors we could reach our milestone yet many more to go in the future. We are grateful towards you all and thank SRA for giving us this opportunity and igniting a lamp which could produce immense light.

Special Thanks to our Superb Mentors:

- **OMKAR SARGAR**
- **PREETI JAIN**
- **SAHARSH JAIN**
- **SHANTANU PANDE**

TEAM SUPERNOVA2.0:

DHAIRYA SHAH
dhairyashah110501@gmail.com
+919769016020

PRATHAMESH TAGORE
prathameshtagore@gmail.com
+918097579575

ATHARVA ALSHI
atharva2001alshi@gmail.com
+919082268675

MAYURESH MANE
mpm351@gmail.com
+917506034848

TABLE OF CONTENTS:

NO.	TITLE	PAGE NO.
1.	PROJECT OVERVIEW: 1.1 Description of Use Case of Project ……. 1.2 Technology Used……. 1.3 Brief Idea…….	 4 4-5 5
2.	INTRODUCTION: 2.1 General…….. 2.2 Basic Project Domains…….. 2.3 Theory……..	 6 6 6-7
3.	METHODS AND STAGES OF PROGRESS: 3.1 Approach…… 3.2 Person Detection ……... 3.3 Co-ordinate Extraction…….. 3.4 Equation of Trajectory……...	 7 7-9 9-13 13-14
4.	CONCLUSION AND FUTURE WORK: 4.1 Current Efficiency and Accuracy…….. 4.2 Achievements till now……... 4.3 Future Aspects……	 14-15 15-16 16-17
5.	REFERENCES: 5.1 Useful links & Research Papers / Helpful Courses ……...	 17

1. PROJECT OVERVIEW:

1.1 DESCRIPTION OF USE CASE AND PROJECT:

The launch mechanism of this project might get applications in:

1. Ball throwing machines like basket ball, cricket and various other games. It will have the same mechanism, i.e., to detect the required person and throw the ball in the required direction. This will also allow an efficient practice for the players.
2. This kind of launching mechanism could be also used in defence purposes. One such example is automated artillery firing, where the machine will be trained to detect the enemy troop and fire the shells accordingly.

1.2 TECHNOLOGY USED:

1. OpenCV library using Python:
   OpenCV is a library of programming functions mainly aimed at real-time computer vision. It is an open-source computer vision and machine learning software library. This library will allow us to capture the video in front of the camera and will also allow us to detect the required person in front of the screen when used with various classifiers.

2. HOG classifier Algorithm:
   The Histogram of oriented gradients, or HOG, is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. In our case, it intended to detect the person in front of the screen.

3. ROS:
   The Robot Operating System, or ROS, is a framework for writing robot software. It is a collection of tool, libraries and conventions that aim to specify

the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. It is used here to simulate the mechanism of the detecting the person and throwing the ball at an accessible range of the person.

4. SVM Classifier Algorithm:
   Support Vector Machines, or SVMs, are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It is used along with HOG classifier in order to get the person in front of the camera detected by the software.

5. NMS Algorithm:
   Non Maximum Suppression (NMS) is a technique used in many computer vision algorithms to select one entity out of many overlapping entities. It is generally used to reduce the large number of detected rectangles which is a defect in object detecting algorithm involving HOG and SVM classifiers.

1.3 BRIEF IDEA:

The aim of the project is to make a bot which is capable to detect a person in front of it and throw a ball towards him in order to give him a catching practice. At first, the bot will detect the person using OpenCV and person detecting algorithms like NMS or HOG and SVM.Then, it will identify the position of the person with respect to the bot and hence will evaluate the distance.Once the person is detected and the distance is evaluated, the bot will randomly decide a position in a range of 1m from the person where the ball will be thrown.

2. INTRODUCTION:

2.1 GENERAL:

Our main idea of the project is to build an automatic robot which can aim at the required target once it is identified. The robot will idetect the person nearby it and will shoot the ball towards that person. This can be achieved by adjusting the shooting components to the required positions for launching the projectile. For this project we have basically used different concepts like Computer Vision(Image Processing) , mechanics projectile equation and ROS simulation. Together summing up all these concepts and applying it we could build our automatic Catch-Practice-Robot.

2.2 BASIC PROJECT DOMAINS:

In our project while throwing the ball towards the target our first main target is to identify the person nearby the robot and choose the random point nearby that person and then launch the ball towards the random point. In the hardware part by using the cameras , two servos and the sensor helps to calculate the distance of the person from the robot. This automatic ball throwing robot can be widely use in sports field like basketball , for defence purposes automated detection of enemies and firing mechanism . This automatic robot doesn't require any manual help while launching the ball it will automatically launch by using cameras and servos. This requires knowledge of image processing for the person detection and the trajectory path equation for calculating the angle.

2.3 THEORY:

To detect the person nearby or in front of the robot it requires image processing knowledge for applying. By using open cv library it allows to get real time computer vision. Further by using Histogram of Oriented Gradients(HOG) classifier and Support Vector Machine(SVM) helped to detect the person in front of the robot. Further using python library random point near the person is selected. Then getting the camera and the world 3-d coordinates the camera and the servos adjusting their position launch the ball towards the randomly selected point. For lauchong

mechanism theoretical knowledge of projectile is required that is the trajectory formula for calculating the angle so the ball can be thrown at that same angle towards target.

Further the ROS Gazebo simulation for making the model of the robot by adding the depth camera , sensors to the robot and the person standing modelnearby the robot. By using this simulation of the is built for the project.

3. METHODS AND STAGES OF PROGRESS:

3.1 APPROACH:

The approach of the project is as described before. First of all its aim is to enable a person to catch the ball or object. Our proposed bot artillery comprises of servos, camera to get the 2D data of 2D image co-ordinates and a depth sensor eg. Lidar or the the whole to be replaced by a 3D camera as per availability. The person detection algorithm will detect the person and is written such that detecting the person it will create an area around the person . By using the random function in python we will select a random point in the area which is very much the point the launch mechanism is supposed to fire. After using co-ordinate extraction and detection hand in hand we will rotate the launching part in that plane using the servos to align the mechanism in accordance to the Equation of Trajectory studied in elementary mechanics. After doing so the object which is the ball in our case, will be fired.

3.2PERSON DETECTION:

The entire detailed process of training an object detector using Histogram of Oriented Gradients (yet), simply because each step can be fairly detailed. But I wanted to take a minute and detail the general algorithm for training an object detector using Histogram of Oriented Gradients. It goes a little something like this:

Step 1: Sample *P* positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples.

Step 2: Sample *N* negative samples from a *negative training set* that ***does not contain*** any of the objects you want to detect and extract HOG descriptors from these samples as well. In practice *N >> P.*

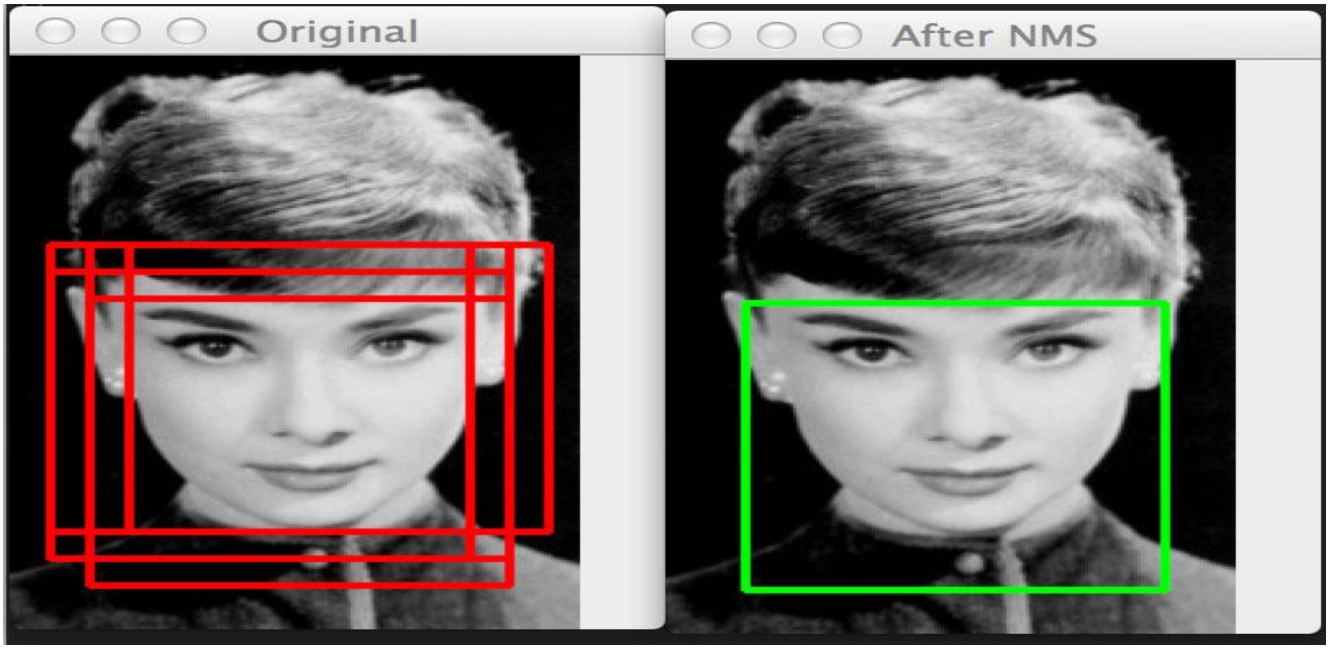
Step 3: Train a Linear Support Vector Machine on your positive and negative samples.

Step 4: **Apply hard-negative mining.** For each image and each possible scale of each image in your negative training set, apply the sliding window technique and slide your window across the image. At each window compute your HOG descriptors and apply your classifier. If your classifier (incorrectly) classifies a given window as an object (and it will, there will absolutely be false-positives), record the feature vector associated with the false-positive patch along with the probability of the classification. **This approach is called *hard-negative mining.***

Step 5: Take the false-positive samples found during the hard-negative mining stage, sort them by their confidence (i.e. probability) and re-train your classifier using these hard-negative samples. *(Note: You can iteratively apply steps 4-5, but in practice one stage of hard-negative mining usually [not not always] tends to be enough. The gains in accuracy on subsequent runs of hard-negative mining tend to be minimal.)*

Step 6: Your classifier is now trained and can be applied to your test dataset. Again, just like in Step 4, for each image in your test set, and for each scale of the image, apply the sliding window technique. At each window extract HOG descriptors and apply your classifier. If your classifier detects an object with sufficiently large probability, record the bounding box of the window. After you have finished scanning the image, apply non-maximum suppression to remove redundant and overlapping bounding boxes.

These are the bare minimum steps required, but by using this 6-step process you can train and build object detection classifiers of your own! Extensions to this approach include a **deformable parts model** and **Exemplar SVMs**, where you train a classifier for *each positive instance* rather than a *collection of them.*However, if you've ever worked with object detection in images you've likely ran into the problem of *detecting multiple bounding boxes around the object you want to detect in the image.*

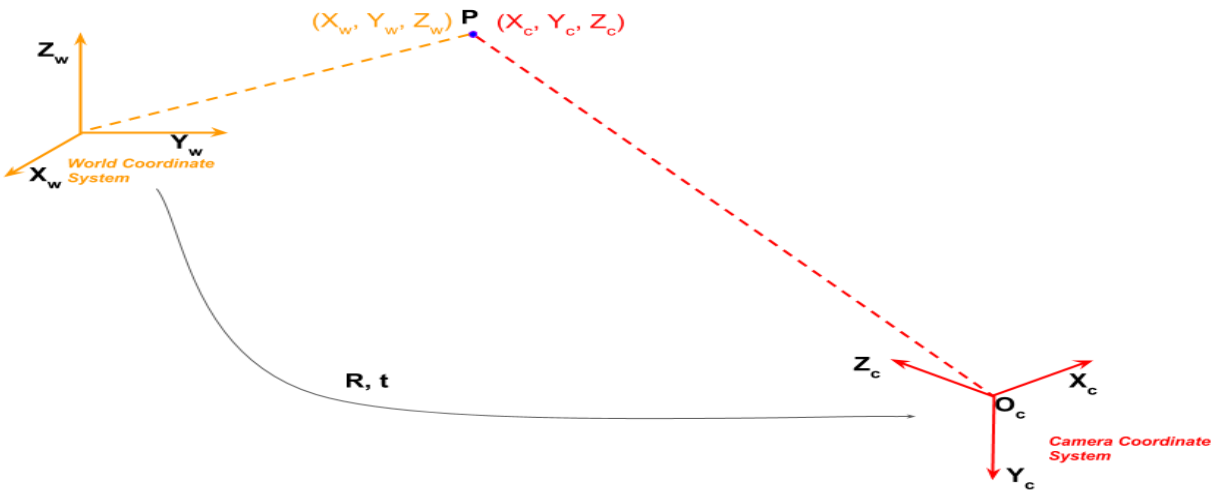


There are multiple ways to remedy this problem. Triggs et al. suggests to use the Mean-Shift algorithm to detect multiple modes in the bounding box space by utilizing the *(x, y)* coordinates of the bounding box as well as the logarithm of the current scale of the image. Thus Person detection is done and by using open cv we can draw rectangles and mark area aroung which we want the the random launch of the ball to be.

3.3 CO-ORDINATE EXTRACTION:

The World Coordinate System and the Camera Coordinate System are related by a Rotation and a translation. These six parameters (3 for rotation, and 3 for translation) are called the extrinsic parameters of a camera.

1. **Origin** : We can arbitrarily fix a corner of the room as the origin $(0, 0, 0)$.
2. **X, Y, Z axes** : We can also define the X and Y axis of the room along the two dimensions on the floor and the Z axis along the vertical wall.



Let's say our camera is located at some arbitrary location (t_X, t_Y, t_Z) in the room. In technical jargon, we can the camera coordinate is **translated** by (t_X, t_Y, t_Z) with respect to the world coordinates.

It is often convenient for mathematical manipulation to encode rotation as a 3×3 matrix. Now, you may be thinking that a 3×3 matrix has 9 elements and therefore 9 parameters but rotation has only 3 parameters. That's true, and that is exactly why any arbitrary 3×3 matrix is not a rotation matrix. Without going into the details, let us for now just know that a rotation matrix has only three degrees of freedom even though it has 9 elements.

Back to our original problem. The world coordinate and the camera coordinates are related by a rotation matrix $\mathbf{R}$ and a 3 element translation vector $\mathbf{t}$.

Figure 1: The World Coordinate System and the Camera Coordinate System are related by a Rotation and a translation. These six parameters (3 for rotation, and 3 for translation) are called the extrinsic parameters of a camera.

To define locations of points in the room we need to first define a coordinate system for this room. It requires two things

1. **Origin** : We can arbitrarily fix a corner of the room as the origin $(0, 0, 0)$.
2. **X, Y, Z axes** : We can also define the X and Y axis of the room along the two dimensions on the floor and the Z axis along the vertical wall.

Using the above, we can find the 3D coordinates of any point in this room by measuring its distance from the origin along the X, Y, and Z axes.

This coordinate system attached to the room is referred to as the **World Coordinate System**. In Figure 1, it is shown using orange colored axes. We will use bold font (e.g. $\mathbf{X_w}$) to show the axis, and regular font to show a coordinate of the point (e.g. X_w).

Let us consider a point P in this room. In the world coordinate system, the coordinates of P are given by (X_w, Y_w, Z_w). You can find X_w, Y_w, and Z_w coordinates of this point by simply measuring the distance of this point from the origin along the three axes.

Now, let's put a camera in this room.

The image of the room will be captured using this camera, and therefore, we are interested in a 3D coordinate system attached to this camera.

If we had put the camera at origin of the room, and align it such that its X, Y, and Z axes aligned with the $\mathbf{X_w}$, $\mathbf{Y_w}$, and $\mathbf{Z_w}$ axes of the room, the two coordinate systems would be the same.

However, that is an absurd restriction. We would want to put the camera anywhere in the room and it should be able to look anywhere. In such a case, we need to find the relationship between the 3D room (i.e. world) coordinates and the 3D camera coordinates.

Let's say our camera is located at some arbitrary location (t_X, t_Y, t_Z) in the room. In technical jargon, we can the camera coordinate is **translated** by (t_X, t_Y, t_Z) with respect to the world coordinates.

The camera may be also looking in some arbitrary direction. In other words, we can say the camera is **rotated** with respect to the world coordinate system.

Rotation in 3D is captured using three parameters —- you can think of the three parameters as yaw, pitch, and roll. You can also think of it as an axis in 3D (two parameters) and an angular rotation about that axis (one parameter).

However, it is often convenient for mathematical manipulation to encode rotation as a 3×3 matrix. Now, you may be thinking that a 3×3 matrix has 9 elements and therefore 9 parameters but rotation has only 3 parameters. That's true, and that is exactly why any arbitrary 3×3 matrix is not a rotation matrix. Without going into the details, let us for now just know that a rotation matrix has only three degrees of freedom even though it has 9 elements.

Back to our original problem. The world coordinate and the camera coordinates are related by a rotation matrix $\mathbf{R}$ and a 3 element translation vector $\mathbf{t}$

What does that mean?

It means that point P which had coordinate values (X_w, Y_w, Z_w) in the world coordinates will have different coordinate values (X_c, Y_c, Z_c) in the camera coordinate system. We are representing the camera coordinate system using red color.

The two coordinate values are related by the following equation

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$$

The 3×1 translation vector is appended as a column at the end of the 3×3 rotation matrix to obtain a 3×4 matrix called the **Extrinsic Matrix**.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

the extrinsic matrix $\mathbf{P}$ is given by $\mathbf{P} = [\mathbf{R}|\mathbf{t}]$

The image plane is placed at a distance f (focal length) from the optical center.

Using high school geometry (similar triangles), we can show the project image (x, y) of the 3D point (Xc, Yc, Zc) is given by.

$$x = f\frac{X_c}{Z_c}$$
$$y = f\frac{Y_c}{Z_c}$$

The above two equations can be rewritten in matrix form as follows

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Let's represent the image coordinates by (u, v).

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \qquad u = \frac{u'}{w'} \qquad v = \frac{v'}{w'}$$

And hence , the co-ordinates can be easily calculated knowing the intrinsic parameters.

3.4 EQUATION OF TRAJECTORY:

The basis of our calculation of angle of projection depends upon the equation of trajectory for a projectile. We shall know the velocity of our projectile coming out of the launch system. We will then feed this data to our micro-controller. Also, the

co-ordinates of our target will be fed to the micro-controller by image processing techniques using opencv. So now, our micro-controller knows v(velocity),x(x co-ordinate of target),y(y co-ordinate of target) but we still require the value of theta(angle of projection) for a successful strike according to the equation……

Trajectory formula is given by

$$Y = X\tan(\varphi) - gX^2/(2u^2\cos^2(\varphi))$$

Where……
X=x co-ordinate of target.
Y=y co-ordinate of target.
g=constant of gravitational acceleration.
φ =angle of projection/altitude/elevation

Thus, the above formula can be used by the micro-controller to calculate the required angle of projection for a successful strike. After calculating the required angle, we shall rotate our arm containing launch mechanism to that particular angle.

4. PROJECT CONCLUSION AND FUTURE WORK:

4.1 EFFICIENCY AND CURRENT ACCURACY:

i. PERSON DETECTION:-

Traditional haar cascade does not provide the desired level of accuracy for the project. We solved this problem by using histogram of oriented gradients method for person detection. On my personal computer, person detection on the concerned images takes approximately 0.1 seconds, this essentially means that I can analyze almost 10 images per second which is a lot better than harr cascade in terms of speed and accuracy. All of our test cases and related code can be found in the project github repository.

ii. CO-ORDINATE EXTRACTION:-

Accuracy related to this section depends on the quality of lidar sensor used for depth extraction which in turn depends on the 3d camera. Based on the assumption of accurate depth data and projection matrix of the given data, we were able to pinpoint the location of our concerned target in centimeters accurately up to 3 decimal places.

iii. ROS and GAZEBO:-

The dummy model of the bot is ready.The simulation of the whole process described was planned to run in gazebo using ros. The problem encountered was inavailability of required plugins which would give the practical ability to individual parts of the bot to function in unison . Also planned servo manipulation, control and coordination using ESP32 failed henceforth. Executing simulation and if possible making the bot practically is one of our future upcomings and we would try to improve it.

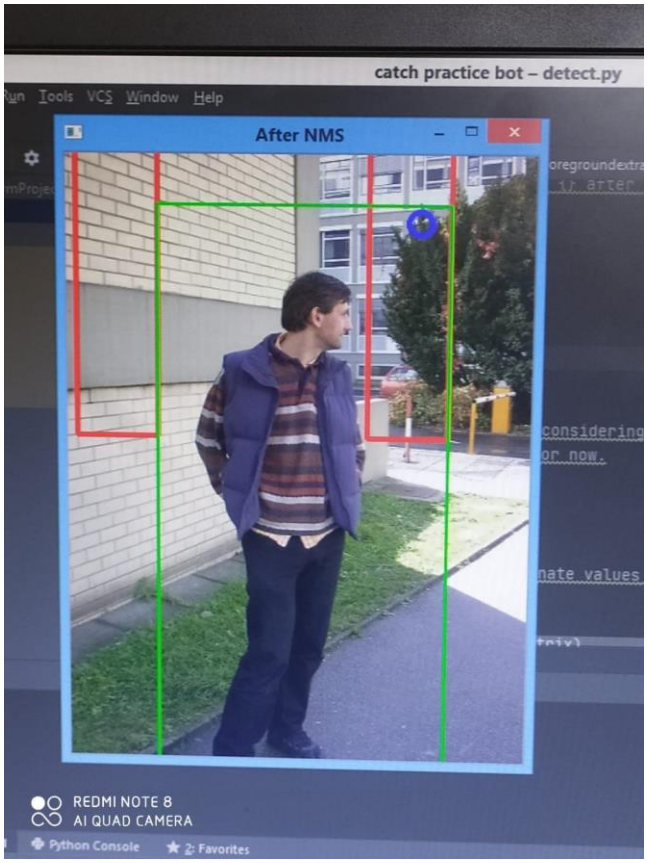
4.2 WHAT DID WE ACHIEVE? :

1. We learned basics of open cv/computer vision and applied that knowledge for the image co-ordinate extraction.

2. For person detection with accuracy, we learned histogram of oriented gradients method coupled with knowledge of linear SVM and non-maxima suppression.

3. For implementing additional features,(such as gesture recognition with hand detection) we learned different techniques such as graham scan for contour detection, contour formation based on color segmentation and motion detection, deploying haar cascades for hand detection etc.

With the help of knowledge gained as above, we were able to identify a person in an image, detect throwing region(region where the person may catch the projectile),pinpoint the exact location of the randomly chosen target point in terms of 3d co-ordinates inside the throwing region, calculate the angle to be rotated by launch mechanism so that it aligns itself perfectly with the target point etc.



4.3 FUTURE ASPECTS OF PROJECT:-

Catch machine throw mechanism has wide applications in the field of sports. Different types of robots such as basketball player, table-tennis player, catch practice machine for fielders in cricket, automatic balling machine for the batsman in cricket etc. can be viewed as extended applications of our main idea.

We are currently learning neural networks which will enable us to implement a robust gesture recognition model in our bot.

5. REFERENCES:

5.1 USEFUL LINKS & USEFUL COURSES:

- https://www.youtube.com/playlist?list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq
- https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/
- https://www.learnopencv.com/geometry-of-image-formation/
- https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/
- https://www.theconstructsim.com/rds-ros-development-studio/?utm_source=youtube&utm_medium=ros-projects&utm_campaign=mrm-p2
- http://wiki.ros.org/Documentation
- https://github.com/dhairyashah1/Eklavya20-CatchPracticeBot.git
- https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
- https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Application
- https://github.com/sashagaz/Hand_Detection
- https://medium.com/@soffritti.pierfrancesco/handy-hands-detection-with-opencv-ac6e9fb3cec1#:~:text=A%20brief%20presentation%20of%20a,written%20in%20C%2B%2B%20with%20OpenCV&text=Handy%20is%20a%20hand%20detection,the%20number%20of%20lifted%20fingers
- https://www.youtube.com/playlist?list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K