AUTONOMOUS FRUIT SORTING

[1]Fr. Fabian Barreto, [2]Vishal S. Jangale, [3]Rakesh B. Hande, [4]Tejas A. Kamble, [5]Waqaar I. Khatri

[1]Asst. Professor, [2]Student, [3]Student, [4]Student, [5]Student

[1]Department of Electronics and Telecommunication

[1]Xavier Institute of Engineering, Mahim, Mumbai.

Abstract: An autonomous system that can be used for automatic high-speed sorting is proposed. In this paper, we are using this system for sorting and picking fruits for ensuring faster production chain. The proposed approach takes into account different types of fruits. The main goal is to come up with a method for classifying these different types of fruits accurately and efficiently. We use Raspberry Pi, which is an open source Linux based board with OpenCV and Python for processing. Furthermore, we make use of camera module which captures the image of the object. Here we use different object detection modules provided in OpenCV, Google's object detection API and Tensor Flow libraries to detect fruits and sort respectively.

***IndexTerms*- Raspberry Pi, Fruit Sorting, Open CV, Tensor Flow, Object Detection API, Python.**

I. INTRODUCTION

[1]Technological advancements especially in Computer vision technology, are gradually finding its applications in the field of agriculture and food, as an answer to one of the biggest challenges facing humankind, namely, sufficing the food requirements of the increasing population. Efforts are being initiated towards the replacement of human operator or human beings looking after the plants and providing and taking care of necessities of plants with automated systems, as human operations are time consuming and are inconsistent.

Autonomous system performs every action which is required to control a process at utmost efficiency using instructions that have been programmed into it or as a response to some activities. Autonomous systems in most cases are faster, durable and more accurate. However, there are some basic architectures which necessarily go hand in hand with automation. Computer vision is a comparatively young discipline with its origin traced back to the 1960s. Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. [2]Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

Computer vision is different from the prevalent field of digital image processing. It works to extract three-dimensional structure from images with the goal of achieving full scene understanding. Studies in the 1970s formed the early foundations for many of the computer vision algorithms that exist today, including extraction of edges from images, labeling of lines, non-polyhedral and polyhedral modeling and representation of objects as interconnections of smaller structures, optical flow, and motion estimation.

[3]Computer vision systems are being used increasingly in the food and agricultural areas for quality inspection and evaluation purposes as they provide suitably rapid, economic, consistent and objective assessment. They have proved to be successful for the objective measurement and assessment of several agricultural products. [4]Over the past decade advances in hardware and software for digital image processing have motivated several studies on the development of these systems to evaluate the quality of diverse and processed foods. The majority of these studies focused on the application of computer vision to product quality inspection and grading. Traditionally, quality inspection of agricultural and food products has been performed by human graders. However, in most cases these manual inspections are time- consuming and labor-intensive. Moreover, the accuracy of the tests cannot be guaranteed. By contrast it has been found that computer vision inspection of food products, was more consistent, efficient and cost effective. Also, with the advantages of superior speed and accuracy, computer vision has attracted a significant amount of research aimed at replacing human inspection. Recent research has highlighted the possible application of vision systems in other areas of agriculture, including the analysis of animal behavior, applications in the implementation of precision farming and machine guidance, forestry and plant feature measurement and growth analysis.

II. OBJECTIVE

The main objective of the project is to detect a fruit, necessarily to distinguish between an Apple and a Banana. The computer program will capture the image from the camera and the image will be processed using image processing of computer vision technology and machine learning , based on a program, after which the computer will decide whether the object placed is a banana or an apple then robotic arm will pick the fruit and place it in the assigned position.

III. Problem Statement

In agricultural sector the efficiency and the accurate grading process is very essential to increase the productivity of produce. Everyday high-quality fruits are exported to other countries and generate a good income. That is why the grading process of the fruit is important to improve the quality of fruits. However, fruit grading by humans in agricultural industry is not sufficient, requires large number of labors and causes human errors. Automatic grading system not only speeds up the process but also gives accurate results. Therefore, there is a need for an efficient fruits grading or classification methods to be developed. Fruit’s color, size, weight, component texture, ripeness are important features for accurate classification and sorting of fruits such as orange’s, apple’s, mango’s etc. Objective of this paper is to emphasize on recent work reported on an automatic fruit detection system.

IV. Block Diagram

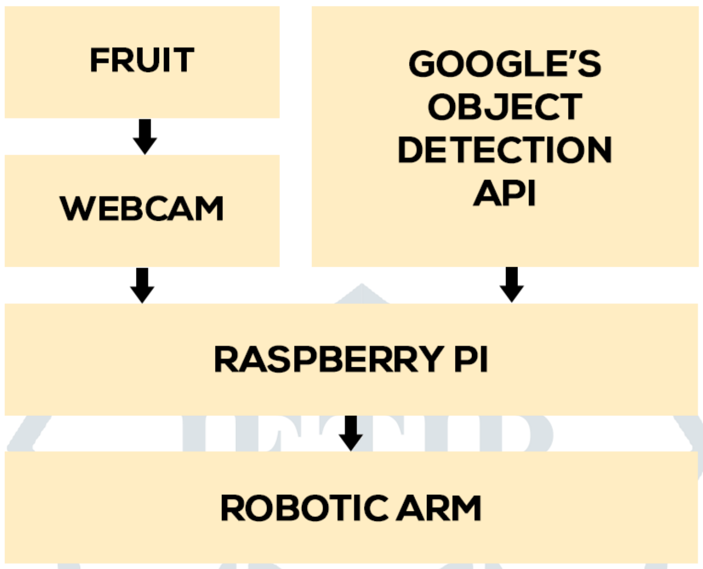

Fig.1 Block Diagram

V. Major Components And Work Flow

5.1 TensorFlow (Library):

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

5.2 OpenCV (Library):

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing.

5.3 Image Database:

TensorFlow needs hundreds of images of an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else, or only halfway in the picture.

For our apple/banana detection, we have two different objects we want to detect. We used our Logitech Camera and our Cell phones to take about 40 pictures of each fruit on its own, furthermore we clicked some pictures with various other non-desired objects in the pictures. Then, we took about another 100 pictures with multiple objects along with the fruit in the picture. We know we want to be able to detect the fruit when they're overlapping or being overlapped by some random object, so we made sure to have the fruit be overlapped by something or other in many images.

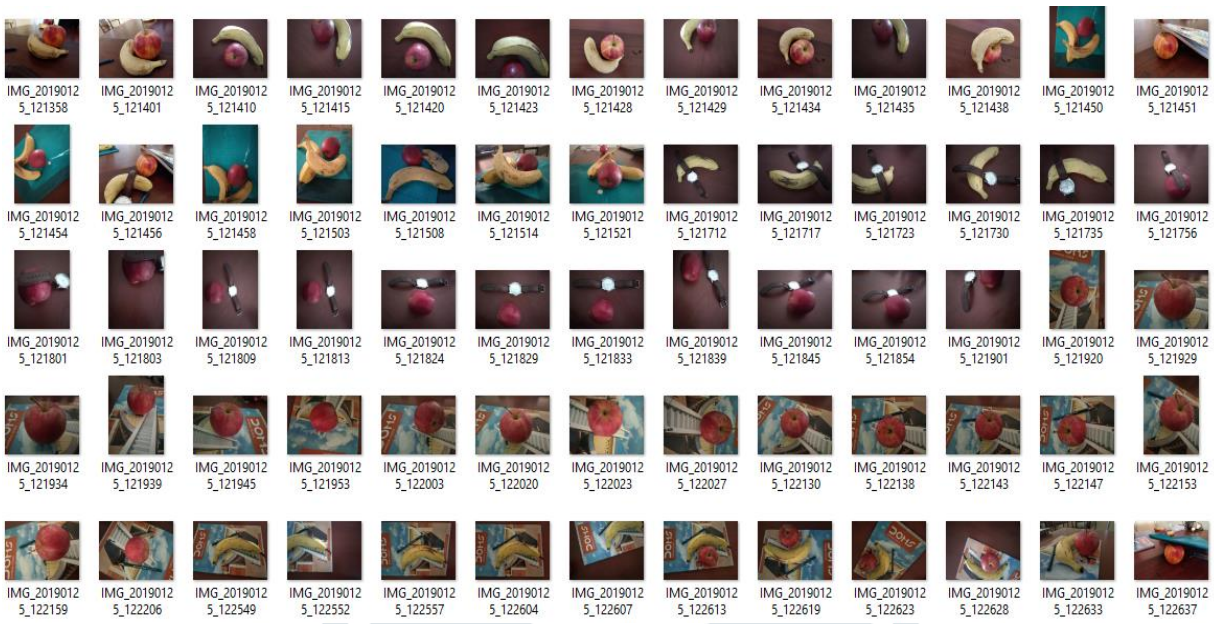

Fig.2 Database Images

We can use your phone to take pictures of the objects or download images of the objects from Google Image Search. It is recommended having at least 200 pictures overall. We used approx. 1024 pictures to train our card detector.

Making sure the images aren't too large. They should be less than 200KB each, and their resolution shouldn't be more than 720x1280. The larger the images are, the longer it will take to train the classifier. For Image size reduction we use Photoshop scripts to bulk reduce the size of the image.

After we have all the pictures you need, move 20% of them to the test directory, and 80% of them to the train directory. Make sure there are a variety of pictures in both the 'test' and 'train' directories. We use the 20/80 split-up in which 20% of the images were used for testing and 80% of the images were used for training.

5.4 Label Images:

With all the pictures gathered, it's time to label the desired objects in every picture. LabelImg is a great tool for labeling images. We download and install LabelImg, point it to our \images\train directory, and then draw a box around each fruit in each image. Repeat the process for all the images in the \images\test directory.

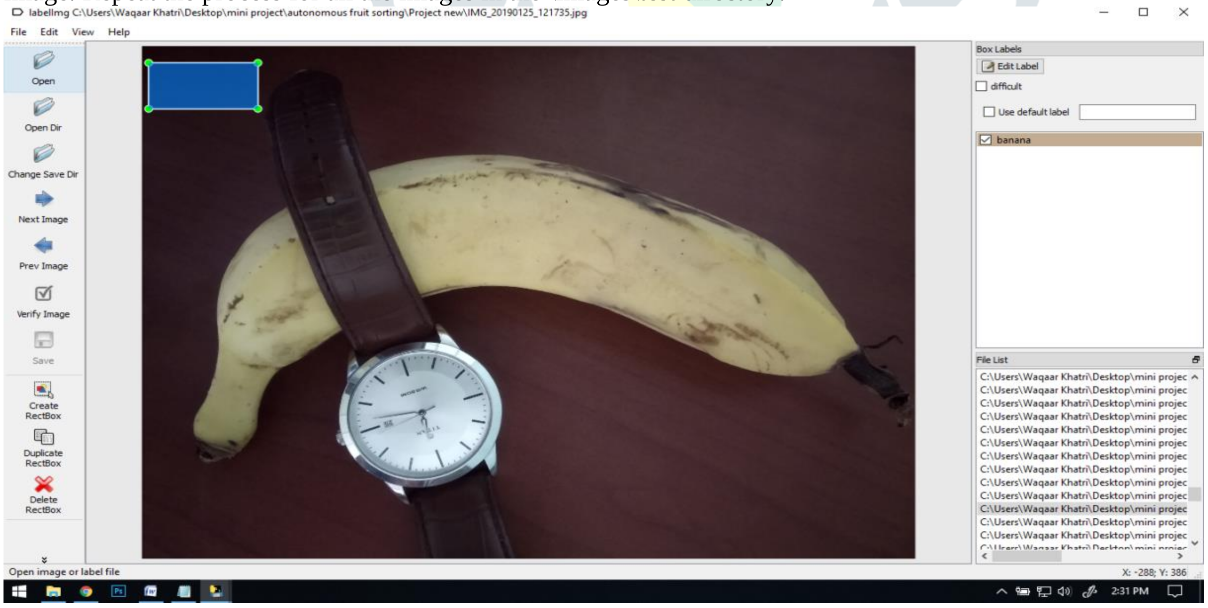

Fig.3 Labelimg Software Example

Steps for labeling using Labelimg Software.

- Click 'Change default saved annotation folder' in Menu/File
- Click 'Open Dir'
- Click 'Create RectBox'
- Click and release left mouse to select a region to annotate the rectangle box
- Use right mouse to drag the rectangle box to copy or move it
- The annotation will be saved to the folder specified.

When pressing space, the user can flag the image as verified, a green background will appear. This is used when creating a dataset automatically, the user can then through all the pictures and flag them instead of annotate them.

Table.1 Hotkeys for different operations in Labelimg.

Ctrl + u	Load all of the images from a directory
Ctrl +	Change the default annotation target directory
Ctrl +	Save
Ctrl +	Copy the current label and rect box
Space	Flag the current image as verified
w	Create a rect box
d	Next image
a	Previous image
del	Delete the selected rect box
Ctrl +	Zoom in
Ctrl +	Zoom out
↑→↓←	Keyboard arrows to move selected rect box

5.5 Generate Training Data:

With the images labeled, it's time to generate the TFRecords that serve as input data to the TensorFlow training model. First, the image .xml data will be used to create .csv files containing all the data for the train and test images. We use the following command

C:\tensorflow1\models\research\object_detection>

python xml_to_csv.py

This creates a train_labels.csv and test_labels.csv file in the \object_detection\images folder.

```
item {
  id: 1
  name: 'apple'
}
item {
  id: 2
  name: 'banana'
}
```

Then, generate the TFRecord files by issuing these commands from the \object_detection folder:

```
python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record

python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record
```

These generate a train.record and a test.record file in out folder. These will be used to train the new object detection classifier.

5.6 SSD_MOBILENET_V2_COCO model:

We used ssd_mobilenet_v2_coco detection model pre-trained on the COCO dataset, the Kitti dataset, the Open Images dataset, the AVA v2.1 dataset and the iNaturalist Species Detection Dataset. This model can be useful for out-of-the-box inference if you are interested in categories already in those datasets. They are also useful for initializing your models when training on novel datasets.

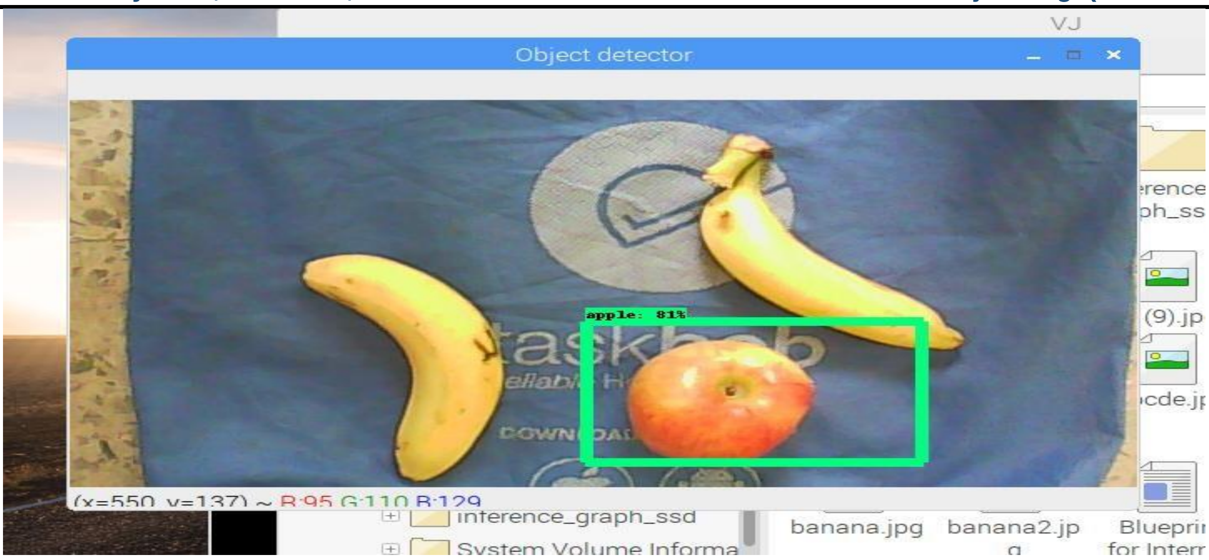

Fig.4 ssd_mobilenet_v2_coco object detection

5.7 Controlling Robotic Arm

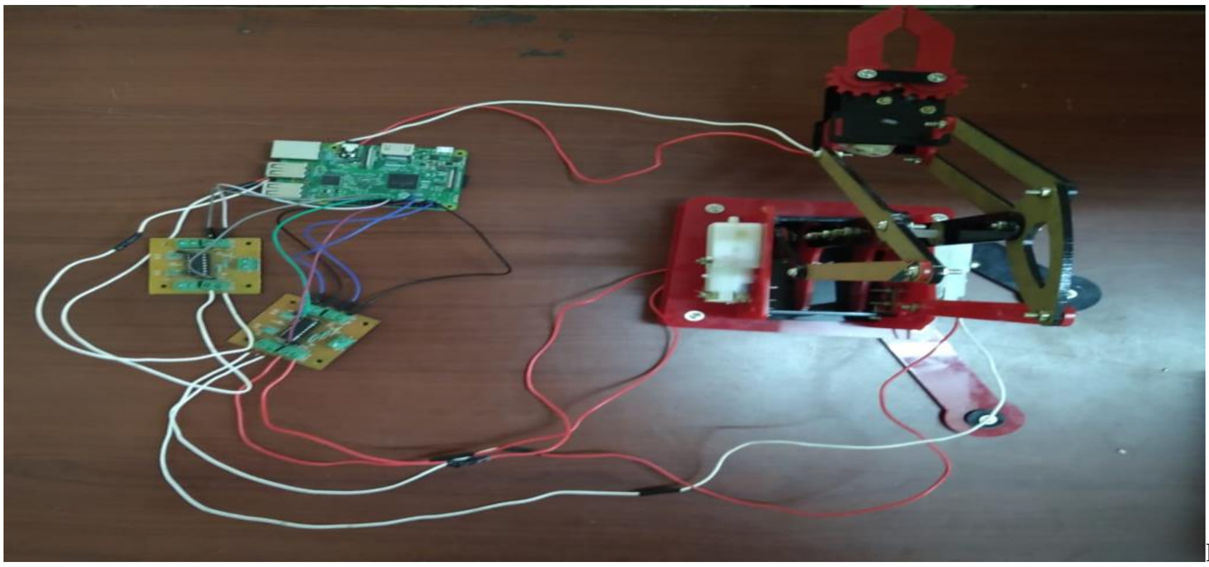

Fig.5 Robotic Arm

We have used and robotic arm with 4 DOF (Degree of freedom) out of which we have only used 3, i.e. base, shoulder, gripper. These motors are controlled by two L293D Motor Driver and a power supply giving 9V is used.

One L293D Motor Driver is with two motor, motor1(base) terminal is connected to L293D Motor Driver motor1 output whose motor1 input pin is connected to raspberry pi GPIO pins 2,3 and pin 4 is connected to the enable pin of L293D Motor Driver that is used to send PWM signal to control speed of the motor similarly motor2 terminal is connected to L293D Motor Driver motor2(shoulder) output whose motor2 input pin is connected to raspberry pi GPIO pins 17, 27 and pin 22 is connected to the enable pin of L293D Motor Driver and another L293D Motor Driver is used to connect the third motor3, motor3 terminal is connected to L293D Motor Driver motor3(gripper) output whose motor3 input pin is connected to raspberry pi GPIO pins 5,6 and pin 13 is
connected to the enable pin of L293D Motor Driver. And the python script is used to run a program that will control these motors via the GPIO pins.

5.8.1 Controlling Speed Of Motors

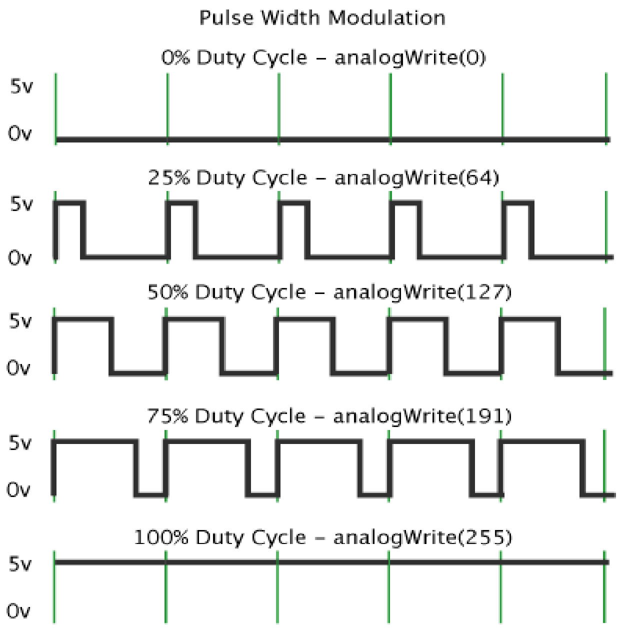

Fig.6 PWM Signals

The way we control the speed of the motor is by using a python module called PWM. That stands for Pulse Width Modulation. What PWM means is just controlling the amount of time a voltage is on by flipping between high and low for a set amount of time. The amount of time the voltage is high is called the 'duty' or 'duty cycle', and whatever percentage that is will be the percentage of power the motor runs on.

The L293D motor IC uses two pins referred to as inputs to sense the desired direction of the output, and another pin called Enable to sense on/Off. So, in our code, with the Enable pin On, if we want the motor to spin forward, we'll set input 1 to 'True' or 'HIGH', and input 2 to 'False' or 'LOW'. And if we want it to spin backwards, we'll set input 1 to 'False" or 'LOW' and input 2 to 'True' or "HIGH'. If both inputs are True or both are False, the motor will not run.

Since the IC has an Enable pin that controls it's on/Off state, we can leave both inputs set to run and just modulate the Enable pin, and the IC will only put out power according to the duty we set in the Enable pin. That way we keep the code simpler, and less things can go wrong.

VI. RESULTS

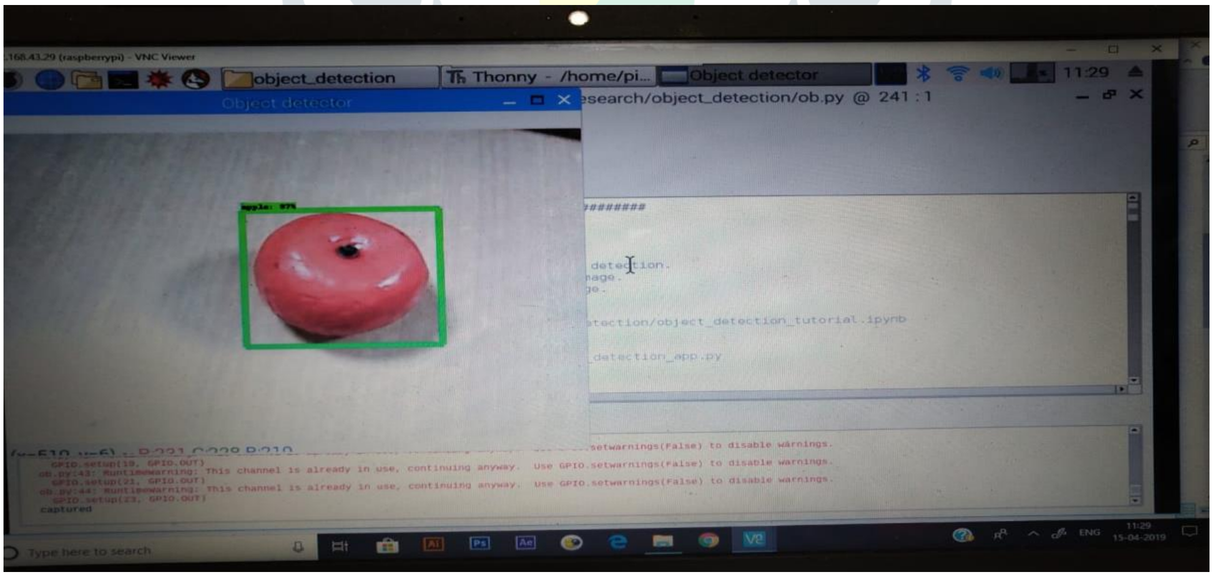

Fig.7 Object Detection Results

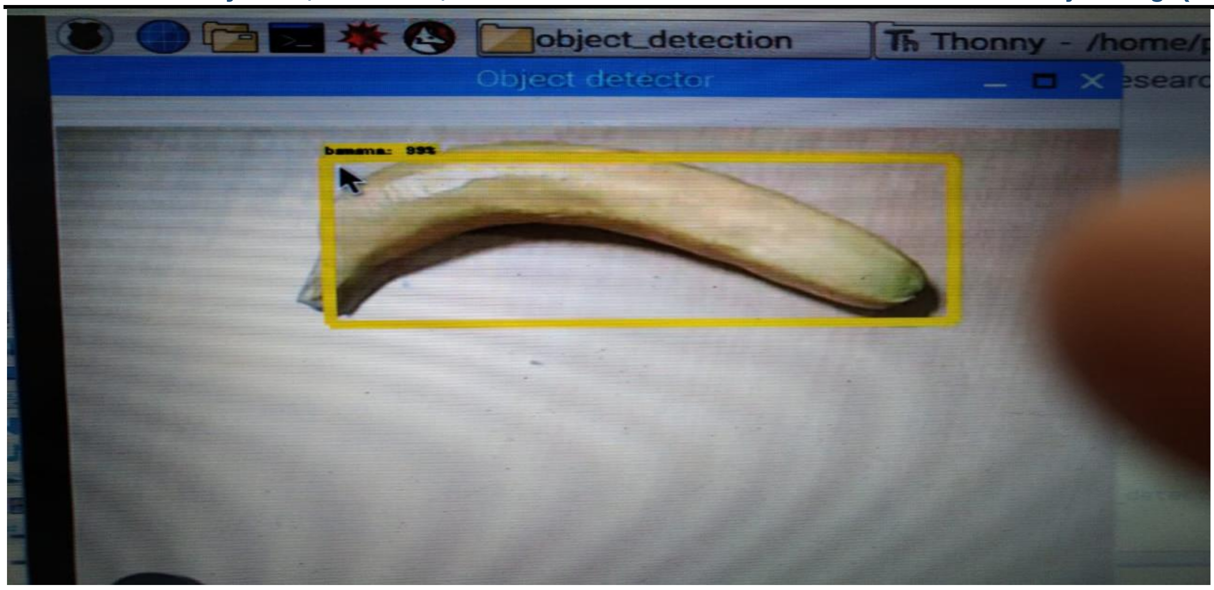

Fig.8 Object Detection Results

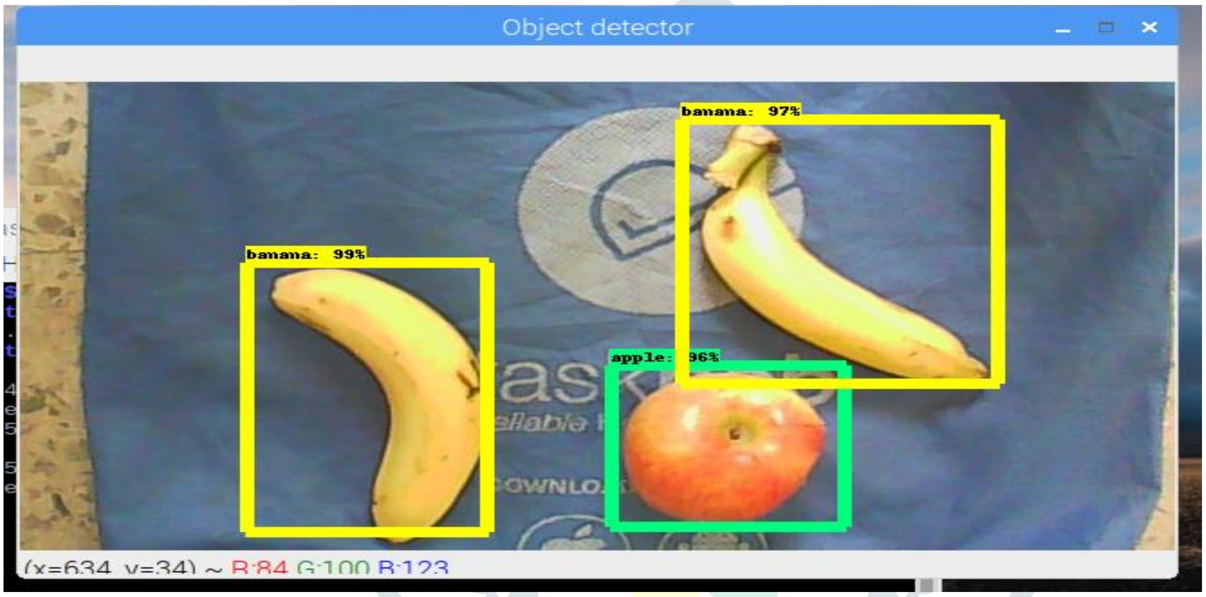

Fig.9 Robotic Arm Results

The detection of fruit is accurate it can detect multiple apple and banana the accuracy is approx. 95%.

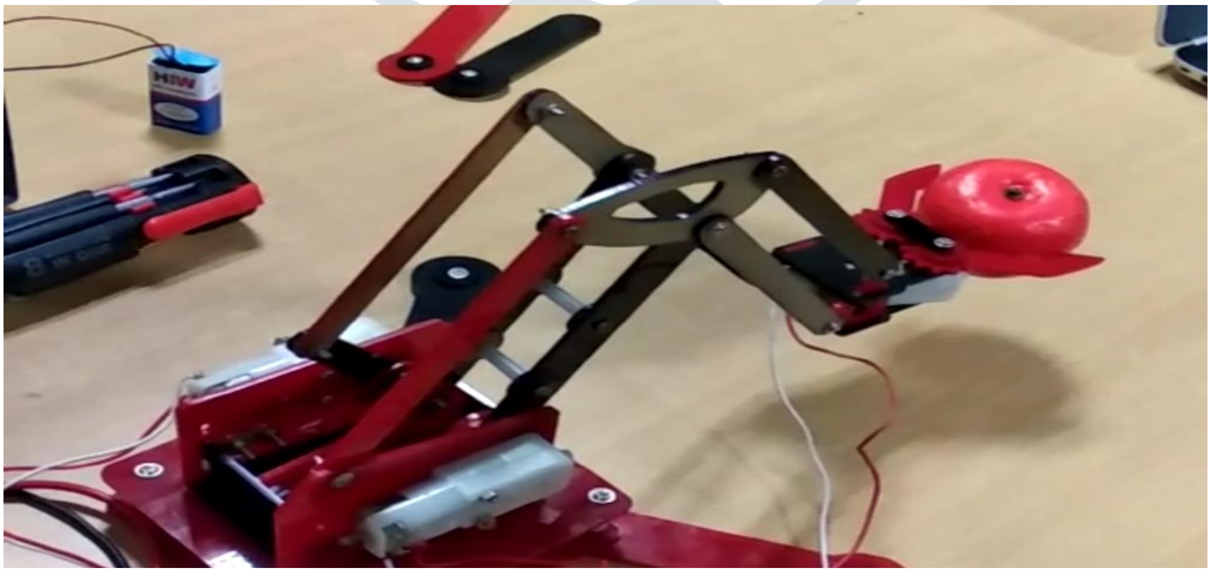
Fig.10 Robotic Arm Results

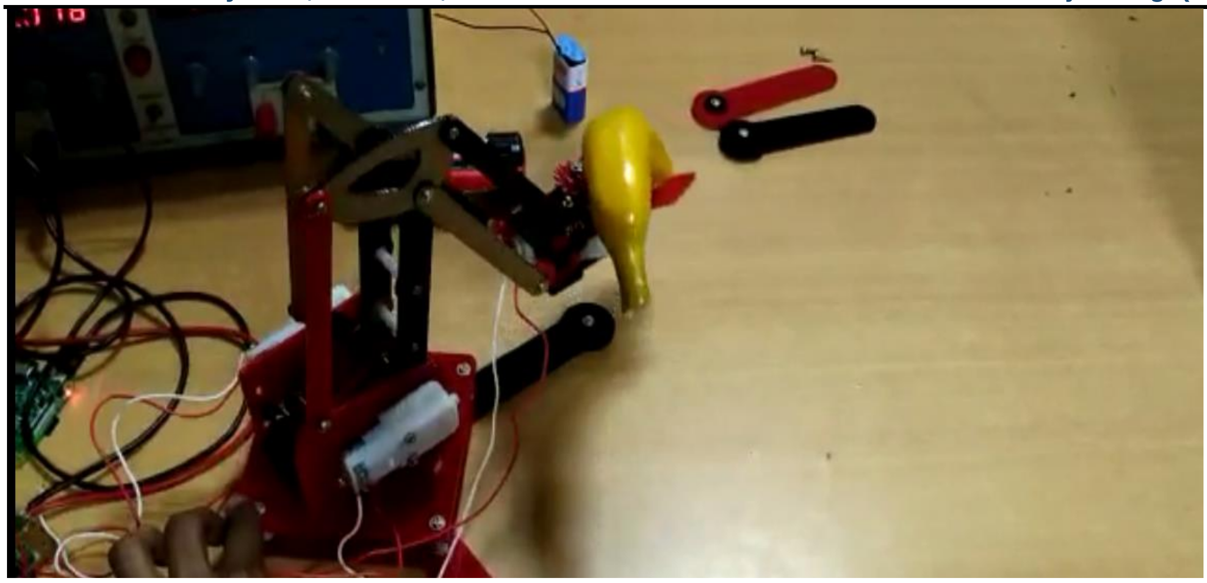
Fig.11 Robotic Arm Results

The robotic arm is able to pick plastic fruits as it is fragile, it puts the detected banana in left side and detected apple in right side.

VII. CONCLUSION

This fruit sorting system can be used to sort two fruit accurately and pick it up and place it in and fixed place this system. This system can also be used to detect quality of fruit and sort them accordingly if the database provided is a database containing a good quality fruit and bad quality fruit.

Database used in this system contained 1042 images of fruit apple and banana, and the accuracy with which it detected the fruit was 90% using ssd_mobilenet_v2_coco model which is deployed on raspberry pi. The system takes approx. 1 minute to detect the fruit as the ssd_mobilenet_v2_coco model on raspberry pi process at very less speed

Robotic arm is able to move fast enough and put the detected fruit to the pre-programed fixed location but the position is not so accurate because of the dc motors hence stepper motor with higher torque can be used with better 3d printed or metal chassis for robotic arm. This system is a low budget system is made using low budget components and uses the deep learning algorithm based on transfer learning. This system can be improved by using more power and faster deep learning model and better components.

VIII. ACKNOWLEDGMENT

This our privilege to express our sincerest regards to our project coordinator, Fr. Fabian Barreto, for her valuable inputs, able guidance, encouragement, wholehearted cooperation and constructive criticism throughout the duration of our project. We deeply express our sincere thanks to our Head of Department Dr Prof. Suprava Patnaik for encouraging and allowing us to present the project on the topic ―Autonomous Fruit Sorting at our department premises.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support.

REFERENCES

[1] Dr. Vilas D. Sadegaonkar, Kiran H. Wagh "Automatic Sorting Using Computer Vision & Image Processing For Improving Apple Quality,"IJIRD ISSN 2278-0211 Vol 4, Issue 1 Jan 2015
[2] Lekha Bhausaheb Kachare "Object Sorting Robot Using Image Processing," IJEECS ISSN 2348-117X Volume 5, Issue 7 July 2016
[3] Mathew George. "Multiple Fruit And Vegetable Sorting System Using Machine Vision," Int J Adv Technol 2015 ISSN: 0976-4860 IJOAT Volume 6 Issue 1,1000142
[4] Vishnu R.Kale, V. A. Kulkarni "Automation Of Object Sorting System Using Pick & Place Robotic Arm & Image Processing" IRAJ International Conference J.N.E.C Aurangabad January 2014
[5] Praveena K S et. Al. "Object sorting using image processing,"IJERT ISSN 2278-0181 NESC 2018
[6] Jyoti Jhawar "Orange Sorting by Applying Pattern Recognition on Colour Image,"2016 Elsevier B.V ISSN 1877-0509 [7] Viren Pereira, Vandyk Amsdem Fernandes and Junieta Sequeira, " Low cost object sorting robotic arm using raspberry pi", 2014 IEEE global humanitarian technology conference, Sept 27, 2014
[8] "Robotics and Computer Integrated manufacturing," Sarah Mansoor et. Al. 2014 Elviser Ltd.