Fruit Harvesting Robot Using Computer Vision

Sahil Mahendra Mangaonkar
Electronics and Telecommunication
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
sahilmmangaonkar@gmail.com

Ritvik Khandelwal
Electronics and Telecommunication
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
ritvik02.kh@gmail.com

Saquib Shaikh
Electronics and Telecommunication
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
saquibs12@gmail.com

Sameep Chandaliya
Electronics and Telecommunication
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
sammy.chandaliya@gmail.com

Shrenik Ganguli
Electronics and Telecommunication
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
shrenikganguli172@gmail.com

***Abstract*—Agriculture has conventionally been a labor-intensive occupation in India. However, in order to provide for the rapidly increasing population, in the face of rising labour costs, there is a need to explore autonomous alternatives in place of traditional methods. This paper proposes a prototype of an autonomous fruit harvesting robot consisting of a robotic arm erected on a mobile chassis. Our proposed design is capable of identifying fruits with the help of a camera module using image preprocessing supplemented with object detection algorithm (YOLO v3). We also qualitatively compared two models, one based only on image processing and the other based solely on object detection algorithm (YOLO), while taking into account the shape and colour of the fruits. When fruits are recognized, the robotic arm is engaged, and the fruit is picked and stored in the container attached to the robot's body. To pick and arrange the fruits, an end effector subsystem is used. We have also used sensors to collect important data like humidity, temperature, and rain for further processing.**

Keywords: Deep Learning, Object Detection, Image Processing, Robotic Arm, End Effector System, Fruit Harvesting Robot

I. Introduction

Agriculture is referred to as "the backbone of the Indian economy." Approximately 60% of India's population is employed in this profession, and a similar proportion of Indian land is classified as agricultural land. A majority of this population consists of small farmers with small tracts of land for which the process of harvesting fruits is very time consuming and labor-intensive. Manual fruit harvesting necessitates the proper quantity of employees at the proper time. Thus, to reduce dependence on seasonal labor, there is a need to mechanise and automate agriculture. Numerous attempts have already been made in this direction.

The project's major goal is to construct a wireless autonomous robot that can detect, pluck, and place fruits in a container linked to its body. Our robot is made up of a four Degrees-of-freedom (DoF) robotic arm with a claw-like end effector that is used to grab and pick the fruits from the plant, then place them in the container using three DC and two servo motors. The fruits are detected using a camera module that is integrated with a Raspberry Pi (microcomputer) that is placed above the arm. To detect the fruits, we deployed two methods. The first one is only image processing, and the second one is done using a deep learning based object detection algorithm. In order to facilitate the traversal of the arm on land, the arm is mounted on a travelling platform. We are using four motors for the robot to move in forward and backward directions and take turns as per requirement. We have to control the direction in which the robots will move as well as the speeds of the motors. This is achieved by the use of motor driver L293D. The Arduino Mega Microcontroller controls the whole mechanical aspect of the robot, including the movement of the arm, end-effector, and cart wheels. This microcontroller board has 54 digital GPIO pins (including 15 PWM pins) and hence fulfils our requirements for controlling the numerous DC motors used in the wheels and servo motors used in the arm. An Arduino is the brain that controls everything, and a 12V lithium polymer (li-po) battery was used to power the entire robot.

II. Related Works

For a better implementation and a more solid design of the robot, we looked into projects that were similar to the targeted use. We examined a number of research papers that addressed issues relating to robot design and arm actuation. On the basis of research in this sector [1, 3, 4], the elements needed to calculate the robot's motion, the end effector system's design, and the system's weight calculation were calculated.

Our group looked at previously established robotic systems that were used to pick fruits. Several articles and research have been written on the development of a three-dimensional robotic arm that would be used to harvest fruits growing in space. The end effector system uses a scissor to cut the branches on which the fruits grow, and the fruit is subsequently placed in the basket attached to the body. A camera module is employed to detect fruits on the farm. Color and shape detection are both done with the camera module. Application of image processing techniques [6, 13, 16] is utilised to detect the fruit and then move the robot to the location where they grow. The position of the fruits in the field was determined using a visual algorithm that was built. A fruit picking robot was guided by this information. One fruit at a time was collected by the robot. The robot's success rate was reported to be 60%.

In a very recent paper [17], the system was powered by a 12-volt Lithium-Ion (LiFePO4) battery. A two DoF robotic arm has been designed which can move in the forward and backward directions while closing and opening the claw. A gear system comprising of rack and pinion was used which was motored by the servo motor. This limited the access to the fruits in the farm making it difficult to pluck the fruit just by pulling it back. In our design, the robotic arm can move up and down and can rotate 180 degrees from the base so that fruits can be placed in the container. In order to pluck the fruits, we twist the claw five times.

In [3], the author has designed an arm that can move vertically and horizontally in one plane, which restricts the access of fruits in other planes. [4] employs a 4-DoF robotic arm with 0-180 degree base rotation, forward-backward movement, up-down motion, and 0-180 degree gripper opening. However, this makes picking the fruits more challenging. In our research, we suggest a robotic arm that twists the fruit five times to pluck it, making it easy to separate the fruit from the plant.

Object identification [9, 10] and tracking [12, 13] is a well-studied field with numerous practical applications. The primary purpose of object detection systems is to recognise the desired elements in a picture, as well as to determine their position in the current world and categorise them.

Machine learning algorithms [7, 8, 9] such as Scale Invariant Feature Transform (SIFT), Random Sample Consensus (RANSAC), and Convolutional Neural Network are used in vision-based software systems to increase detection and tracking accuracy (CNN). The use of neural networks necessitates sophisticated technology as well as access to a large database, resulting in a complex system. Such technologies are more difficult to operate remotely and to put into practice in the real world.

Another method is to use purely image processing [14] methods such as colour detection [6, 11] using various colour spaces such as RGB (Red, Green, and Blue), CMY (Cyan, Magenta, and Yellow), HSL (Hue, Saturation, and Luminance), HSV (Hue, Saturation, and Value), and shape detection using morphological transformation methods such as dilation and erosion. The benefit of utilising this strategy is the convenience of remotely processing in real-time on smaller devices, at the expense of some precision.

Inspired by [19], we have experimented and supplemented pre-processing techniques and also used sensors for remote monitoring [16].

III. Proposed System

A. *Robot:*

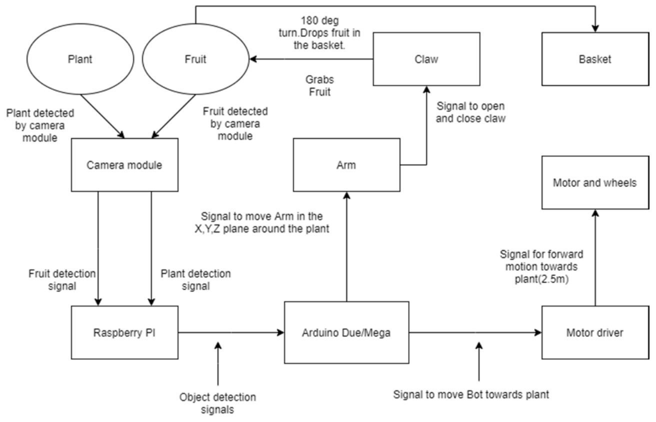

Plant
Fruit
180 deg turn.Drops fruit in the basket.
Claw
Basket
Grabs Fruit
Plant detected by camera module
Fruit detected by camera module
Signal to open and close claw
Camera module
Arm
Motor and wheels
Signal to move Arm in the X,Y,Z plane around the plant
Signal for forward motion towards plant(2.5m)
Fruit detection signal
Plant detection signal
Raspberry PI
Arduino Due/Mega
Motor driver
Object detection signals
Signal to move Bot towards plant

Fig 1: Block diagram of proposed system

The project comprises an Arduino based robotic arm for plucking up the fruits. The PCB consists of the Arduino Mega, motor drivers, optocouplers, servos, and power supply. The Arduino IDE is used to program the Arduino microcontroller. The heart of the robot, the Arduino, is the one which controls the motion of the robot and the movement of the arm and gripper with the help of servo motors. The wheels of the robot are attached to the DC motors.

The motors are driven using motor drivers, which control the speed and direction. Optocouplers are used to isolate the motor driver from the Arduino microcontroller, which saves it from the back emf induced due to mechanical movement of the shaft. The 4 degree of freedom robotic arm is designed with the help of servos and DC motors. The camera module is placed just above the claw. The gripper has two fingers to grab the fruits.

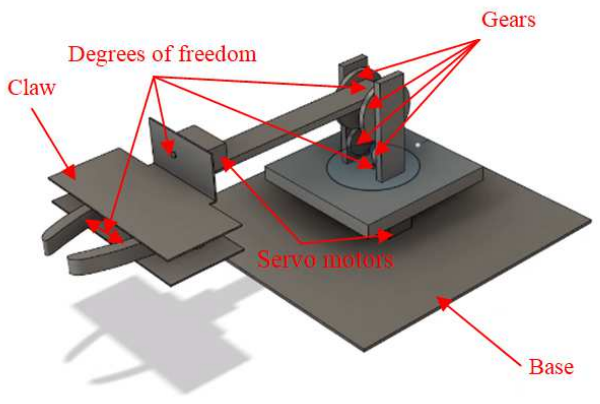


Fig 2: CAD of the robotic arm using Solidworks

The robot will be moving in the y-direction while the camera continuously searches for fruits. As soon as a fruit is detected using the image processing or object detection algorithm, a signal is sent to the robot to stop moving in the y-direction and the camera module is adjusted such that the fruit is in front of the camera. This will fix the camera and claw in along the y-axis. After this, the arm moves z' distance in z direction so that the claw is in front of the fruit. This fixes the position of the claw in the y-z plane. The robot then starts moving in the x direction until it touches the fruit. The contact sensor detects that the claw is touching the fruit and sends the appropriate signal, which closes the claw and plucks the fruit by twisting it. After that, the arm puts the fruit into the basket and comes back to its original position.

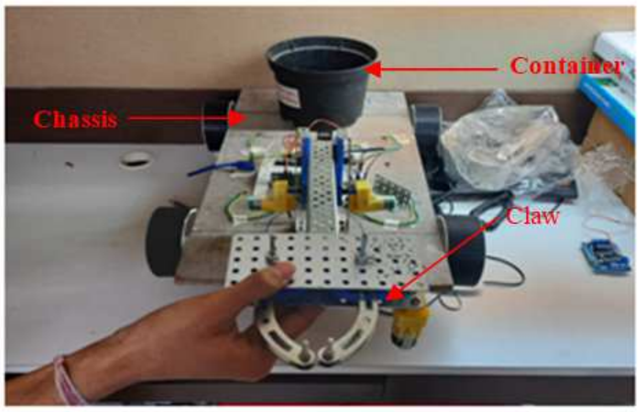


Fig 3: Front view of fully assembled arm

The whole system is powered by 12V LiPo 5500mAh and 3300mAh batteries. Owing to lower weight, increased capacity, and power delivery, LiPo batteries are the best fit for robot systems. We have used the Arduino Mega as the microcontroller pertaining to its varied GPIO pins and system interface. The voltage and current are supplied to the input of the L293D motor driver. DC motor drivers are necessary to drive the DC motors, which demand high current requirements for their operation. The motor driver circuit is separated from the Arduino Mega by the use of an optocoupler. We have used the PC817 IC for isolation purposes. Here, the input circuit is the Arduino, and the output circuit is the motor driver driving the DC motors. The isolation of the input from the output is a must as we do not want the input to be affected by the back EMF generated by the motors on the output side. In general, isolation methods are adopted to prevent any circuit from getting affected due to a surge in electronic factors.

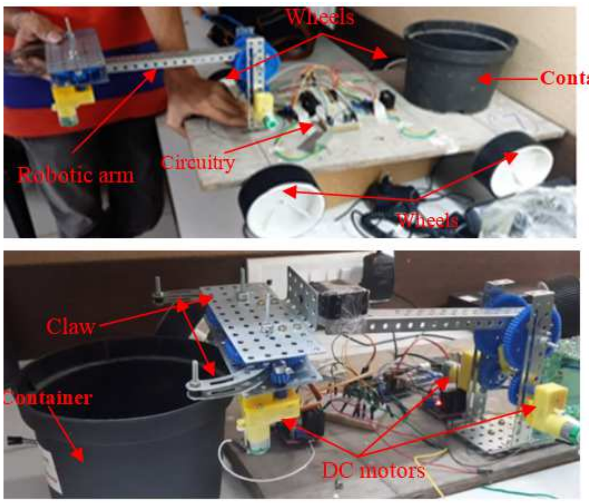


Fig 4: Side view of fully assembled arm

Firstly, the base of the robot starts to move along the row of plants in a plantation. The DC motors are attached to the wheels. The DC motors take 12V as input from the motor driver. As a result, the robot moves along the plant row. For this task, we have used four DC motors attached to the four wheels. On encountering a fruit in its path, the camera module captures the image and compares it with the algorithm built into the Raspberry Pi to carry out object detection.

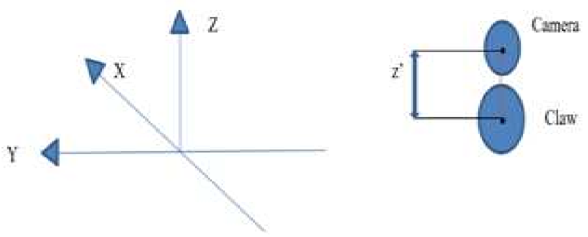


Fig 5: The motion of the base and the claw of the robot

*X direction into the screen. Z direction up and down. The Y axis runs along the left and right sides of the screen.

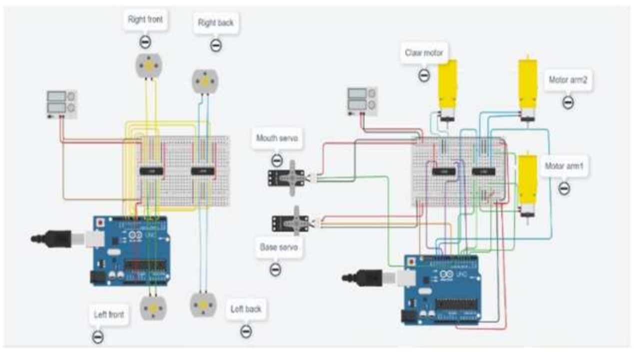


Fig 6: Circuit Simulation in Tinkercad

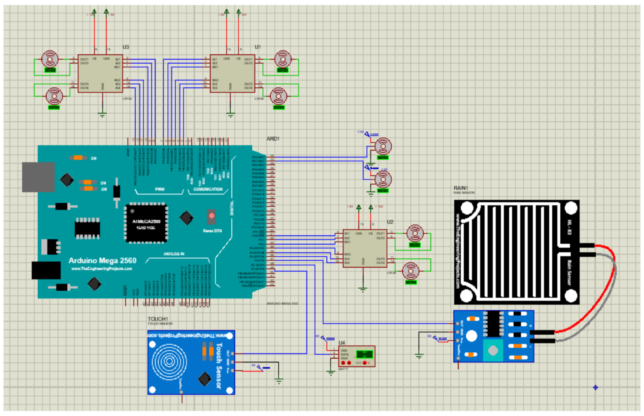


Fig 7: Final circuit including sensors and actuators

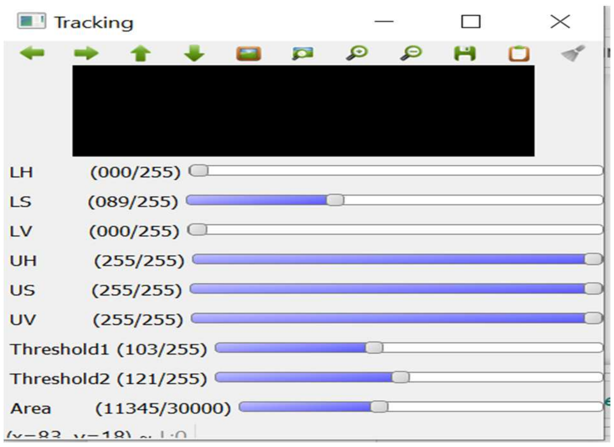
After recognising the desired fruit, the DC motors stop. Hence, the robot stops. Now, the command from the Arduino Mega prompts the servo motors to activate. Two servo motors were used. Of those, one was at the base of the arm, the other at the elbow. The servo motors provide precise angular motion. Servo motors typically take an input voltage of 7.4V. Since the position of the robot is fixed in the linear plane (X axis), the arm extends in the Y-Z plane, perpendicular to the linear plane. The range of angles of the servo motors is programmed for each part of the arm. On grabbing the fruit, the servo motor at the wrist rotates, the claw closes, and the fruit is plucked by the gripper. The gripper then drops the fruit into the basket behind it.

B. *Vision System*

The first method is the image processing method. Method 1 has further been divided into 1A and 1B. In 1A, we have used simple thresholding and dilation, while in 1B, we have used morphological transformations. As per our requirement, we will be extracting various features from the images in the dataset so that our model will learn to classify the fruits as per our requirements. The following was carried out to detect a fruit and an apple. We start by converting the RGB image to grayscale. Using a kernel of 7 by 7, we applied median blur, which is used to reduce salt and pepper noise. We tried using different blurring techniques and found median blur to be the most appropriate.

To detect the colour of the fruit, a mask is applied to the captured frame, highlighting only the desired colour, which in this case is the colour of the fruit. The software converts the RGB colour space into the HSV colour standard when the image is captured. This is effective for focusing attention on a single colour while blocking out all other colours in an image. After that, the mask will be divided into two types:

one for lower intensity levels and another for higher intensity levels, with the lower intensity level mask being utilised for lower intensity levels. To retrieve the desired value of the fruit's colour in HSV, we also used trackbar as an alternative method.



LH (000/255)
LS (089/255)
LV (000/255)
UH (255/255)
US (255/255)
UV (255/255)
Threshold1 (103/255)
Threshold2 (121/255)
Area (11345/30000)

Fig 8: Trackbar

The image is subjected to morphological manipulation to improve the accuracy of the contours. morphological operation was also used to close holes inside the fruit image. We specifically applied morphological operation opening to remove external image noise. Since morphological transformation had a certain time lag in real-time, we used simple thresholding along with image dilation. Dilation function is used to remove noise from the image. This along with canny edge detection gave an output with negligible time lag in real time. Other edge detection methods were also used, like Canny, SobelX, SobelY, and Scharr for edge detection. After comparing, we saw that Laplacian provided the best results.

With this method, we also use contours to recognise shapes, which is done with the help of the camera. Contours are all of the places along an item's boundary in an image that are the same colour or intensity as one another. The number of vertices used to identify the outlines of an object of interest is critical in establishing whether the object is square, rectangular, or circular in shape. OpenCV includes a function for detecting contours in images that can be used to identify objects in the image. In the memory is a collection of points along the boundary, which are referred to as contours. We can build contours by only storing the vertex points along the boundary rather than all of the pixels along the boundary; this procedure, known as contour approximation, is employed in practically all form identification applications and is widely used.

minEnclosingCircle is a function used to find coordinates of center and radius of circle made on contour. drawContours() function is used to draw contours. Contour only gets displayed if the area of contour is greater than or equal to threshold area, thus neglecting all small unnecessary contours in the image.

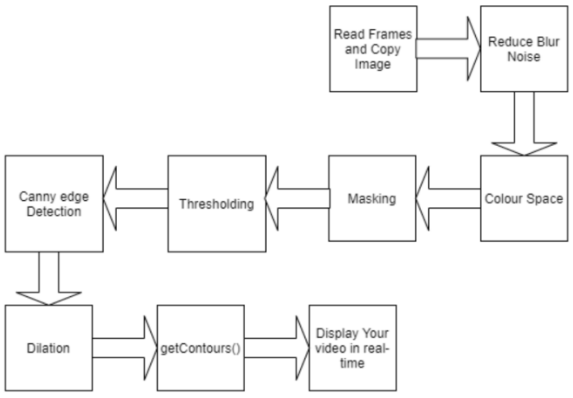


Fig 9: Flowchart of Method 1A

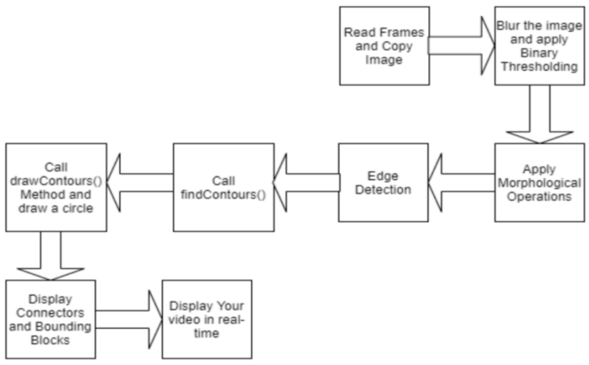

Read Frames and Copy Image
Blur the image and apply Binary Thresholding
Apply Morphological Operations
Edge Detection
Call findContours()
Call drawContours() Method and draw a circle
Display Connectors and Bounding Blocks
Display Your video in real-time

Fig 10: Flowchart of Method 1B

Now for our method 2 where we supplemented preprocessing techniques along with deep learning. For this, we used Yolo (You Only Look Once) V3.0.0, which is primarily a visual aid. We have used it as it has very high processing speed and high accuracy. We can achieve a certain balance between these two parameters by changing the model size. After the insertion of an image into the neural network, it creates the necessary parameters and predicts the possibilities of the frame object. The latest version has several scale forecasts for advanced classification. YOLO is much faster and friendlier compared to F-CNN, as it offers much lower levels of delay. To make the camera frames, we announced the Video Capture item and gave it a camera reference. After that, we created a temporary loop to continue getting the most updated frame on the camera. Using the study method, we got a picture and showed it using the imshow function. The next step is to load the Yolo V3 model. As trained on the coco dataset, we started collecting the names of our classes. This we imported into the coco.namesfile. Each model has two main features. One is architecture and the other is weight. For Yolo V3 we have both separate files. We therefore imported a structured configuration file and a weighting file containing the weights.First, we announced the variables of the method. We then downloaded our model using the readNetFromDarkNetfunction. We also set the backend to openCV and the target to CPU. We cannot send our camera image form directly to the network. It should be in a specific format called blob. The blobFromImage function, which produces a 4D blob from an image, was utilised. Resize as well as crop the image from the middle, subtract used values, scale values by scale factor, and swap the blue and red channels. All default values had been retained.

Now, depending on the image size we used when downloading cfg and weight files, we set the image size. Since Yolo architecture has 3 layers of extraction, we have to find their names so we can get their results. To find words, we have used the getLayerNames function. This returns all the names, but all we need are the names of the extraction layers. We therefore used the getUnconnectedOutLayers function, which returns the output layer indexes. We now use these references to find names in our layerNames list. Since we used 0 as the first element, we must subtract 1 from the indices we find by performing the function of getUnconnectedOutLayers. We will now be able to move forward and get network results. This will bring us back a list of 3 arrays of the next size

- 300 x 85
- 1200 x 85
- 4800 x 85

So now that we have the same members that contain all the details of the boxes, we filter out low confidence and build a list of appropriate boxes that contain items. We created a new job under the name findObjects. To keep the details of the appropriate boxes we will create 3 lists. One will contain the details of the points in the corner of the junction box, the other the class id with the highest confidence and the last one with the highest confidence level of the class. Now we will take out 3 output and get the boxes one by one. We called each box a det, short to find, because it contains more information than just a box link. Now that we have a confidence level we can refine. So we increased the confidence limit. So if self-confidence is a greater than this, it will only be worth the thing found. Then we got the pixel values of x, y, w, h. To find the pixel value we simply multiplied its width and height respectively. Note that we will use x, y which is the root base than cx, cy which is the center point. Eventually we added prices to the corresponding lists.

During this time we drew bounding boxes. But sometimes more than one box points to the same object. In this case, instead of one acquisition we will have 2 acquisitions, while we actually have only one acquisition. To combat this problem, we will use Non Max Suppression. In simple terms NMS removes scattered boxes. It finds scattered boxes and based on its self-confidence will select a high-confidence box and compress all the boxes without max. We therefore used the built-in NMSBoxes function. We entered the points of the combination box, their confidence values, confidence limit and threshold. The function returns the references after deletion. Now that everything is in order we simply pulled out the bounding boxes and showed the names of the items and the confidence values. Both the methods were quantized to fit into the raspberry pi and executed on a robotic arm with the main goal of fruit harvesting.

IV. Results

All the comparisons were carried out qualitatively. We observed that method 1B where we utilized the morphological operations resulted in better accuracy and smoothness compared to Method 1A. However, because morphological processes are computationally intensive, there was a large real-time lag. In real time, Method 1A, on the other hand, had a negligible time lag.

The image pre-processing supplemented with Yolo algorithm was observed to be much more accurate than the image processing methods i.e. Method 1. The neural network was quantized to fit into the raspberry pi and we were able to detect various fruits like orange, apple, banana, broccoli, carrot and a potted plant.

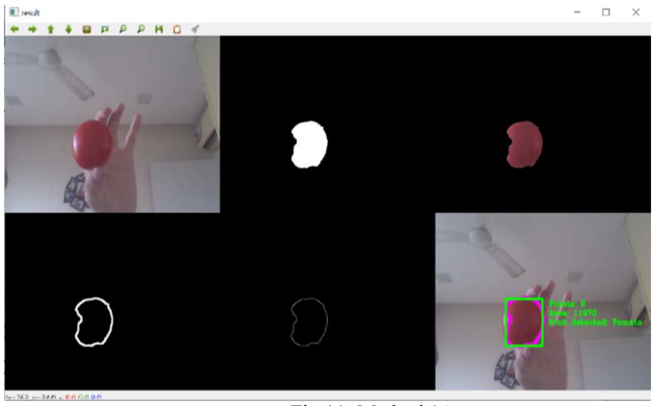


Fig 11: Method 1A

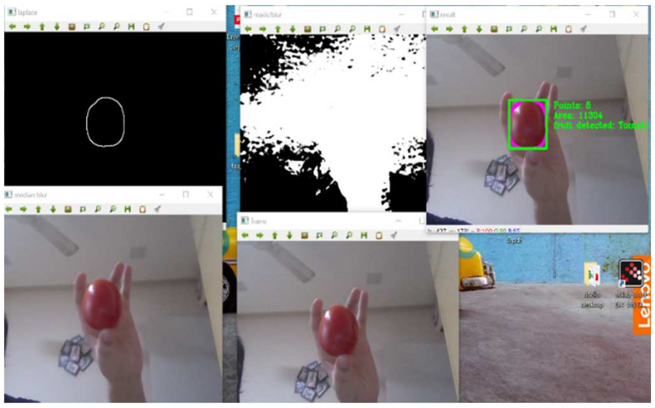


Fig 12: Method 1B

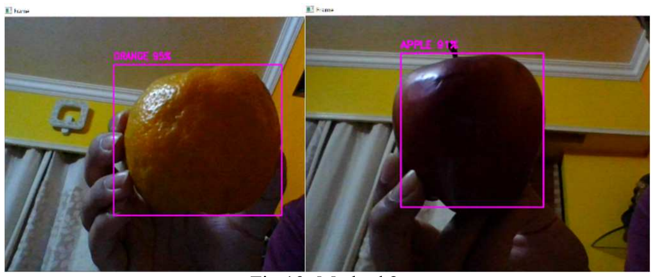


Fig 13: Method 2

V. Sensors

A. Camera Module:

Sony IMX219 which is of 8-megapixel sensor is used in the v2 camera module. This module can record video of high definition while taking still photos. It comprises of different video modes and we utilized the 1080p30 mode. This camera module is used to capture images to detect fruits using image processing.

B. Digital Sensor TTP223B Module Capacitive Touch Switch:

A capacitive touch sensor may detect a change in capacitance when a solid that can conduct electricity comes into contact with one of its electrodes (Fruit, a human body, etc.). TTP223B is the basis for the Digital Capacitive Touch

Switch Module. It normally produces little output and maintains a low power state. It produces high and changes to the quick reaction state when a touch is detected on the circular indicated region. It returns to low power mode after 12 seconds of not being touched. The claw linked to the robotic arm will close as soon as the touch is recognized.

C. *DHT22: Digital-output relative humidity & temperature sensor/module:*

The DHT22 generates a calibrated digital signal. To maintain dependability and stability, it uses a patented digital signal collection technique as well as humidity monitoring technology. The sensing devices is inter-connected with an 8-bit single-chip CPU. This sensor is used to keep track of the temperature and humidity inside the farm for a specific amount of time.

VI. Conclusion

The project is designed to be cost-effective for farmers while still being responsive to their needs. We designed a prototype within a very reasonable budget. This robotic arm consists of 4 degree of freedom (DOF).The robotic arm was controlled by the Arduino Uno microcontroller which was interfaced with DC motors and server motors for rotatory motion of the arm. The fruit was detected by the camera module by using the computer vision algorithm developed by us. On detecting the fruit, a signal is sent to the robotic arm via Arduino microcontroller to pluck the fruit. We aimed to keep the system basic but strong so that it may be scaled to use in a variety of real-world scenarios.

As a future scope of the project, with appropriate funding, this prototype can be scaled to made farm ready and have higher degrees of freedoms. Data analysis can also be carried out from the data collected from different sensors like temperature, humidity, rain, etc. This analysis can aid farmers to take informed actions. From the computer vision front, post processing techniques (Prior knowledge inference) can also be supplemented with the existing methods and a detailed analysis can be carried out quantitively by using evaluation metrics.

VII. References

[1] Peter P. Ling, Reza Ehsani, K.C. Ting, Yu-Tseh Chi, Nagarajan Ramalingam, Michael H. Klingman, and Craig Draper, "Sensing and End-Effector for a Robotic Tomato Harvester," 2004, Ottawa, Canada August 1 - 4, 2004, 2004.

[2] C. de la Peña, "Thinking Through the Tomato Harvester," Boom, vol. 3, no. 1, pp. 34–40, 2013.

[3] F. Taqi, F. Al-Langawi, H. Abdulraheem, and M. El-Abd, "A cherry-tomato harvesting robot," 2017 18th International Conference on Advanced Robotics (ICAR), Jul. 2017.

[4] R. Yenorkar and U. M. Chaskar, "GUI Based Pick and Place Robotic Arm for Multipurpose Industrial Applications," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Jun. 2018.

[5] Coding Raspberry PI &Python: Step by Step Guide from Beginner to Advance by Leonard Eddison.

[6] I. Benkhaled, I. Marc, and D. Lafon, "Colour contrast detection in chromaticity diagram: A new computerized colour vision test," 2017 IEEE Western New York Image and Signal Processing Workshop (WNYISPW), Nov. 2017.

[7] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, "Object Detection Algorithms for Video Surveillance Applications," 2018 International Conference on Communication and Signal Processing (ICCSP), Apr. 2018.

[8] C. H. Low, M. K. Lee, and S. W. Khor, "Frame Based Object Detection--An Application for Traffic Monitoring," 2010 Second International Conference on Computer Research and Development, 2010.

[9] X. Liu, B. Dai, and H. He, "Real-time object segmentation for visual object detection in dynamic scenes," 2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR), Oct. 2011.

[10] B. Gupta, A. Chaube, A. Negi, and U. Goel, "Study on Object Detection using Open CV - Python," International Journal of Computer Applications, vol. 162, no. 8, pp. 17–21, Mar. 2017.

[11] Shape Size Colour Estimation of Object using OpenCV by Aditya Bhardwaj

[12] Colour Object Tracking on Embedded Platform Using Open CV by Krutika a Veerapur, Ganesh V. Bhat.

[13] G. Chandan, A. Jain, H. Jain, and Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV," 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Jul. 2018.

[14] M. E. Celebi and B. Smolka, Eds., "Advances in Low-Level Color Image Processing," Lecture Notes in Computational Vision and Biomechanics, 2014.

[15] S. Velasco-Forero and J. Angulo, "Vector Ordering and Multispectral Morphological Image Processing," Advances in Low-Level Color Image Processing, pp. 223–239, Dec. 2013.

[16] R. Oommen Thomas and K. Rajasekaran, "Remote Monitoring and Control of Robotic Arm with Visual Feedback using Raspberry Pi," International Journal of Computer Applications, vol. 92, no. 9, pp. 29–32, Apr. 2014.

[17] S. Salvi, V. Sahu, R. Kalkundre, and O. Malwade, "Autonomous Tomato Harvester Using Robotic Arm and Computer Vision," Intelligent Communication, Control and Devices, pp. 75–90, 2021.

[18] J. Baeten, K. Donné, S. Boedrij, W. Beckers, and E. Claesen, "Autonomous Fruit Picking Machine: A Robotic Apple Harvester," Field and Service Robotics, pp. 531–539.

[19] A. Kuznetsova, T. Maleva, and V. Soloviev, "Using YOLOv3 Algorithm with Pre- and Post-Processing for Apple Detection in Fruit-Harvesting Robot," Agronomy, vol. 10, no. 7, p. 1016, Jul. 2020.