**ACKNOWLEDGEMENT**

The euphoria and the sense of achievement that accompany the successful completion of our minor project would be incomplete without acknowledging the crucial contribution of those who made it possible and whose guidance and encouragement made our effort successful.

We are grateful to our guide **Prof.** **Dr. Hemalata Bhujale,** and thank her immensely for her help, support and guidance throughout our work.

We are thankful to **Dr. Gopal A. Bidkar,** HOD of electronics and communication engineering, for encouraging us and providing us with all the facilities required during our project.

We are grateful to our principal **Dr. S. B. Vanakudre**, for providing academic environment during our work.

We are also thankful to all the teaching and non-teaching staff of electronics and communication department.

**ABSTRACT**

The growth of the technologies requested higher performances to fulfill the human needs and market. This project is implemented to make human work easier and can reduce the use of human power because of its potential applications. The project “Fruit Sorting Using Image Processing” is used to detect and distinguish between different kinds of fruits. This report puts forth all the detailed explanation about the components used and its working. Whenever a fruit is placed under camera, the code extracts the features and distinguishes the fruits based on their color.

**CONTENTS**

**Chapter No. Chapter Name Page No.**

**1 INTRODUCTION 5**

1.1 Motivation 5

1.2 Problem Definition 5

1.3 Introduction to Image processing 6

1.4 Area of Utility 6

**2 COMPONENTS REQUIRED 7**

2.1 Software Tools 7

2.1.1 Python 2.7 7

2.1.2 OpenCV 7

2.1.3 Modules and Libraries 8

2.2 Hardware Tools 9

2.2.1 Raspberry Pi 3 9

2.2.2 Raspberry Pi Camera10

**3 WORKING AND OPERATION 15**

3.1 Methodology 15

3.2 Block Diagram 15

3.2.1 Image Analysis 11

3.4.1 Image Acquisition 11

3.4.2 Image Pre-Processing 12

3.4.3 Color Detection 12

3.4.4 Detecting the Fruit and 12

Displaying the Results

3.3 Flowchart 13

3.4 Image Processing Concepts used in Project 14

3.4.1 Gaussian Filter 14

3.4.2 Kernel 14

3.4.3 Morphological Transformations 15

3.4.3.1 Erosion 15

3.4.3.2 Dilation 16

3.4.3.3 Opening 16

3.4.3.4 Closing 16

3.4.4 Contours 17

3.5 Code 18

**4 APPLICATIONS 20**

4.1 Applications 20

4.2 Conclusion20

**REFERENCES 21**

**PICTURE GALLERY 22**

**LIST OF FIGURES**

**Sl. no Figure name Page No.**

2.1 Raspberry pi 3 9

2.2 Raspberry pi camera 10

3.1 Block Diagram 11

3.2 Flow Chart 13

3.3 Gaussian Filter 14

3.4 Kernel 14

3.5 Original Image 15

3.6 Image after Erosion 15

3.7 Image after Dilation 16

3.8 Image after Opening 16

3.9 Image after Closing 16

3.10 Extreme Point Contours 17

**CHAPTER 1**

**INTRODUCTION**

**1.1 Motivation**

Agricultural industry means industry, which relates to agriculture. These industries focus the post-harvest process such as processing the agricultural products after harvest and storing the products for domestic applications. This process also includes cleaning, sorting, grading and packaging. Sorting and grading is one of the post-harvest process which classifies the products based on appearance, size and shape which determines the quality of food products. Sorting is also done by human experts, but is more tedious, time taking process.

The above-mentioned disadvantages can be overcome by automatic sorting technique through machine vision which is fast, accurate and cost effective. Machine vision includes

capturing the images, analysis and processing of images, making easy to achieve the region of interest and make easy to determine visual quality characteristics in food products. In recent years, many of the agricultural and food industries which include sorting and grading fields of fruits use the image processing and machine vision techniques. The quality attributes such as shape, size, color and other external features are analyzed using machine vision techniques. Computer Vision is used to capture images from the real world and gather from these. It includes image acquisition, preprocessing, analyzing and understanding the sample images to gather the information in symbolic form or numerical value. The main aim of computer vision is to reproduce the effect of human vision by electronically perceiving and understanding the images.

**1.2 Problem Definition**

To develop a software which sorts different fruits based on its color.

**1.3 Introduction to Image Processing**

* Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it.
* It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.
* Image processing basically includes the following three steps:

1. Importing the image via image acquisition tools;

2. Analyzing and manipulating the image;

3. Output in which result can be altered image or report that is based on image analysis

**1.4 Area of Utility**

* Jam Industry
* Fruit packaging industries
* Horticulture

**CHAPTER 2**

**COMPONENTS REQUIRED**

**2.1 Software Tools**

**2.1.1 Python 2.7**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

**2.1.3 OpenCV**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](https://opencv.org/platforms/android/) and Mac OS.

**2.1.4 Modules and libraries**

* **time:**

This module provides various time-related functions. time.sleep(secs). One of the core functions of the **time** module is **time ()**, which returns the number of seconds since the start of the epoch as a floating-point value.

* **numpy:**

numpy is the fundamental package for scientific computing with Python. It contains a powerful N-dimensional array object useful for linear algebra, Fourier transform, and random number capabilities

* **imutils:**

This package includes a series of OpenCV + convenience functions that perform basics tasks such as translation, rotation, resizing, and skeletonization.

* **cv2:**

This library enables us to use tools of **OpenCV library.**

**2.2 Hardware Tools**

**2.2.1 Raspberry Pi 3**

The Raspberry Pi 3 Model B is the third generation Raspberry Pi as shown in fig 2.1. This powerful credit card sized single board computer can be used for many applications and supersedes the original Raspberry Pi 3 model B+ and Raspberry Pi 2 model B. Whilst maintaining the popular board format the Raspberry Pi 3 model B brings you a more powerful processes, 10x faster than the first generation Raspberry Pi. Additionally, it adds wireless LAN & Bluetooth connectivity making it ideal solution for connected designs.

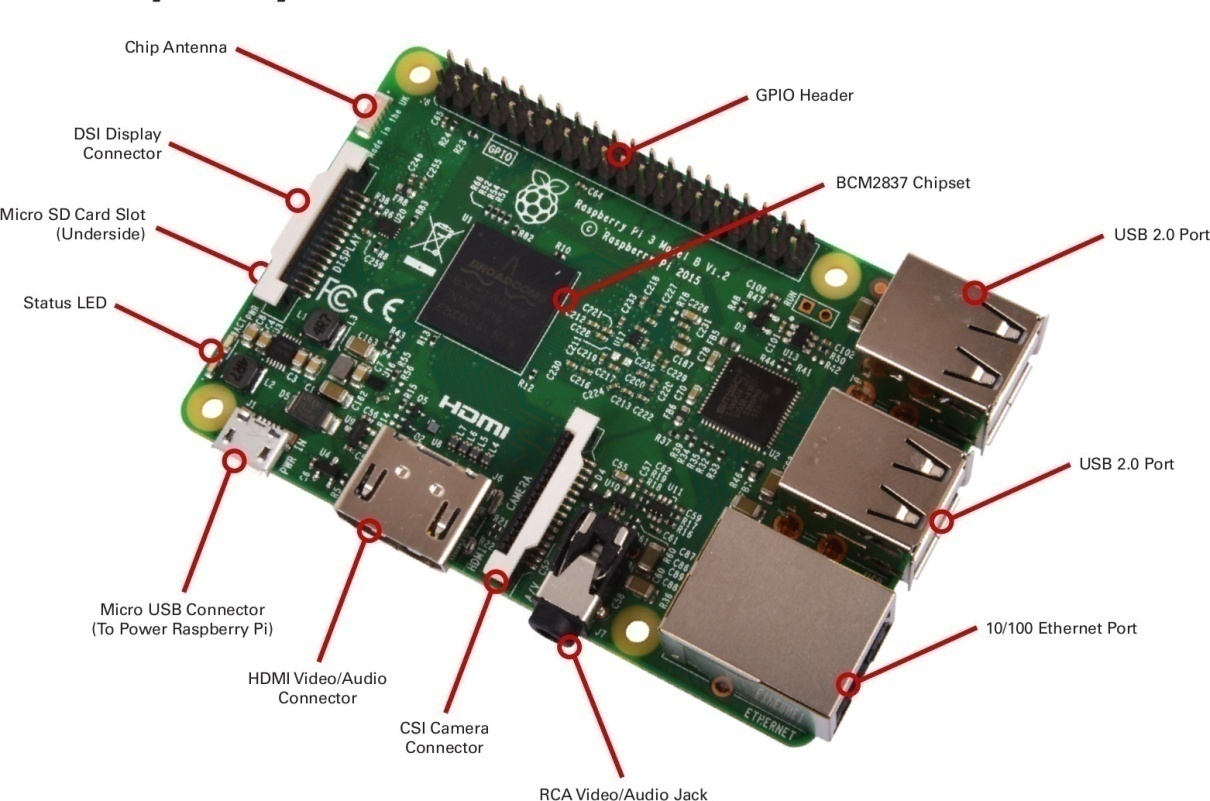


Fig 2.1: Raspberry pi 3

**2.2.2 Raspberry Pi Camera**

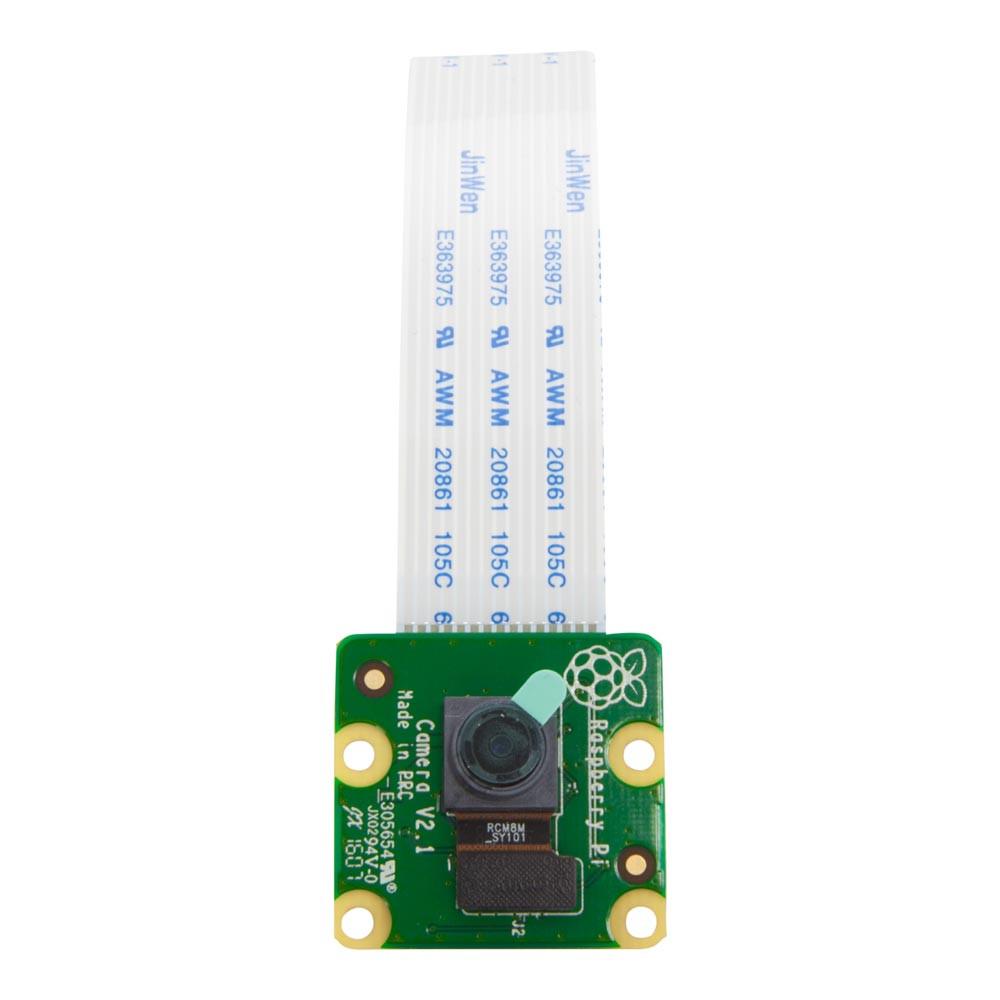
****

Fig 2.2: Raspberry pi 3 camera

* RPI CAMERA BOARD as shown in fig 2.3 plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps with latest v1.3. Board features a 5MP (2592 × 1944 pixels) Omnivision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 pin Ribbon Cable, to the dedicated 15 pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor.
* Compatible all models of Raspberry Pi 1, 2 and 3
* 5MP omnivision 5647 camera module
* Still picture resolution of 2592 x 1944
* Video supports 1080p at 30fps, 720p at 60fps and 640x480p 60/90 recording
* 15 pin MIPI camera serial interface plugs directly into the Raspberry Pi Board
* Size is 20mm x 25mm x 9mm
* Weight of 3g
* Fully compatible with the ModMyPi Raspberry Pi Case

**CHAPTER 3**

**WORKING AND OPERATION**

**3.1 Methodology**

The methodology of this project can be divided as follows: -

1. Image Acquisition

2. Image Pre-Processing

3. Color Detection

4. Detecting the Fruit

5. Displaying the Results

**3.2 Block Diagram**

The below block diagram fig 3.1 shows the working principle of the Fruit Sorting Machine Using Image Processing. First the pi camera captures the image of fruits, then the Raspberry Pi 3 does the Image analysis of the captured image and a decision is made based on color and the results are displayed.

RASPBERRY PI 3

DISPLAY DETECTION RESULTS

CAMERA

DECISION BASED ON COLOR AND SHAPE

IMAGE ANALYSIS

Fig 3.1 Block Diagram

**3.2.1 Image Analysis**

**3.2.1.1 Image Acquisition**

To acquire the image, first we must set up a proper dark or black background for the fruit. The fruit is kept directly under the Raspberry Pi camera with a black background. Proper and bright lighting is a must to capture a proper image. After starting the live video stream from the camera, we return the frames of video at some constant time. These returned images are then processed by the Raspberry Pi 3 to get the results

**3.2.1.2 Image Pre-Processing**

After receiving the returned image, we first resize the frame of image according to our needs. Next step is to smoothen the image using a filter. It is useful for removing high frequency content (e.g. noises and edges) from the image. In our project we are using Gaussian Blur kernel. After blurring the image, we convert it from RGB to HSV model. After this we take the section of image which has values in the range of the values specified by us.

Even after these processes, if we still have some more noise we clear it using morphological operations. For perform such operation, we first need a matrix of ones (i.e. a kernel). We use the kernel to do Erosion and Dilation operations on the images. Both the operations are done in sequence. If dilation follows erosion, it removes the background noise. While the vice-versa will remove foreground noise.

**3.2.1.3 Color Detection**

We define appropriate values of thresholds for the colors of fruits in the dictionary. the values are in HSV values. The values are given along with the color keys. The loop of processing moves forward only in the frame contains any values between the upper and lower threshold values of the keys.

**3.2.1.4 Detecting the fruit and Displaying the Results**

Based on the key and contour, the fruit is detected. Next, we draw a circle around the fruit with the centre coordinates and the radius measured from above steps. A text is displayed showing the name of the fruit. In this way, the fruits are sorted.

**3.3 Flowchart**

START

START THE CAMERA AND CAPTURE FRAMES

APPLY THE IMAGE PROCESSING ALGORITHMS ON THE CAPTURED IMAGE

CHECK IF THE VALUES LIE WITHN THE REQUIRED VALUES

NO

YES

DISPLAY THE DETECTED FRUIT

CHECK FOR KEYBOARD INTERRUPT

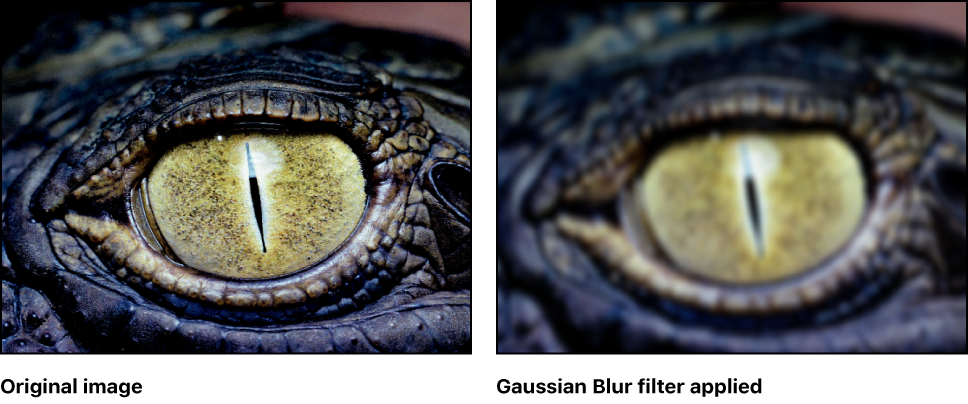
END

Fig 3.2: Flow chart

**3.4 IMAGE PROCESSING CONCEPTS USED IN THE PROJECT**

**3.4.1 Gaussian Filter**

* Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array.
* It is a Low Pass Filter.
* It is useful for removing high frequency content (e.g. noises and edges) from the image.



(a) (b)

Fig 3.3: (a) Original Image, (b) Blurred Image

**3.4.2 Kernel**

* A kernel is a small matrix that slides from left-to-right and top-to-bottom across a larger image. At each pixel in the input image, the neighborhood of the image is convolved with the kernel and the output stored.

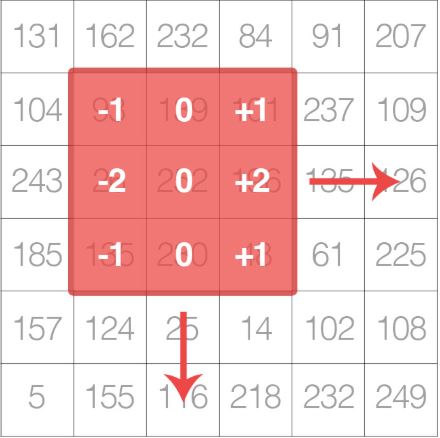


Fig 3.4: A kernel matrix

**3.4.3 Morphological Transformations**

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are **Erosion** and **Dilation.** Then its variant forms like **Opening**, **Closing** etc. also comes into play.



Fig 3.4: Original image

**3.4.3.1 Erosion**

* The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).
* All the pixels near boundary will be discarded depending upon the size of kernel.
* So, the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises



Fig 3.5: Image after Erosion

**3.4.3.2 Dilation**

* It is just opposite of erosion. Here, a pixel element is ‘1’ if at least one pixel under the kernel is ‘1’.
* So, it increases the white region in the image or size of foreground object increases and It is also useful in joining broken parts of an object.



Fig 3.6: Image after Dilation

**3.4.3.3 Opening**

* Opening is just another name of erosion followed by dilation.
* Here we use the function, **cv2.morphologyEx()**



Fig 3.7: Image after opening

**3.4.3.4 Closing**

* Closing is reverse of Opening, Dilation followed by Erosion.



Fig 3.8: Image after Closing

**3.4.4 Contours**

* Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
* We use **cv2.findContours()** function to find contours. It stores the (x,y) coordinates of the boundary of a shape.

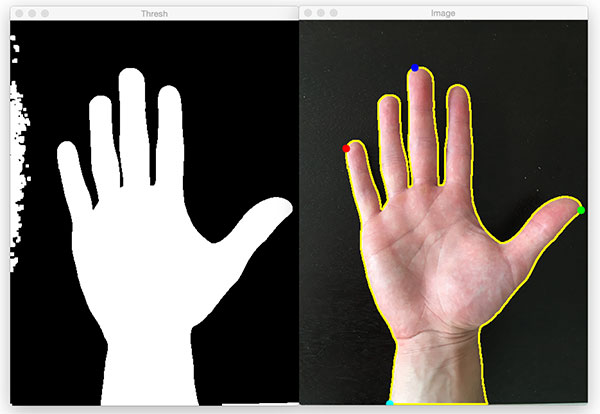


Fig 3.9: Extreme point Contours

**3.4.5 COLOR DETECTION**

* We set the upper and lower threshold values (in HSV format) for the specific color we are interested in.
* Then we use the cv2.inRange() function to get the HSV values between the range of Upper and Lower values.

**3.5 Code:**

import time

import numpy as np

import imutils

import cv2

from picamera import PiCamera

from picamera.array import PiRGBArray

from pivideostream import PiVideoStream

lower = {'red':(166, 84, 141), 'green':(66, 122, 129), 'blue':(97, 100, 117), 'yellow':(23, 59, 119), 'orange':(0, 50, 80)}

upper = {'red':(186,255,255), 'green':(86,255,255), 'blue':(117,255,255), 'yellow':(54,255,255), 'orange':(20,255,255)}

colors = {'red':(0,0,255), 'green':(0,255,0), 'blue':(255,0,0), 'yellow':(0, 255, 217), 'orange':(0,140,255)}

vs=PiVideoStream()

vs.start()

time.sleep(2.0)

while True:

frame = vs.read()

frame = imutils.resize(frame, width=800, height=None, inter=cv2.INTER\_AREA)

blurred = cv2.GaussianBlur(frame, (11, 11), 0)

hsv = cv2.cvtColor(blurred, cv2.COLOR\_BGR2HSV)

for key, value in upper.items():

kernel = np.ones((9,9),np.uint8)

mask = cv2.inRange(hsv, lower[key], upper[key])

mask = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel)

mask = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernel)

cnts = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)[-2]

centre = None

if len(cnts) > 0:

c = max(cnts, key=cv2.contourArea)

((x, y), radius) = cv2.minEnclosingCircle(c)

M = cv2.moments(c)

centre = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

if radius > 0.5:

cv2.circle(frame, (int(x), int(y)), int(radius), colors[key], 2)

if key == 'red':

cv2.putText(frame,"Apple", (int(x-radius),int(y-radius)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.6,colors[key],2)

time.sleep(3)

elif key == ‘orange':

cv2.putText(frame,"Orange", (int(x-radius),int(y-radius)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.6,colors[key],2)

time.sleep(3)

cv2.imshow("Frame", frame)

time.sleep(3)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

camera.release()

cv2.destroyAllWindows()

**CHAPTER 4**

**APPLICATIONS**

**4.1 Applications**

This project can be used in the industry where fruits need to be properly organized, counted and sorted.

1. It can be used in Jam Industry

2. It can be used for Grading fruits

3. It can be applied to any sorting mechanism

4. Quality determination centers.

**4.2 Conclusion**

* We can perfectly define the quality of the fruits based on its color
* We can define or have multiple grading sections in the fruit sorting mechanisms thus being more precise about the quality

**REFERENCES**

1. Youtube.com – sentdex tutorials
2. OpenCV Documentation, Tutorials and Modules
3. Hongshe Dang, Jinguo Song, Qin Guo, “A Fruit Size Detecting and Grading System Based on Image Processing,” 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, pp 83-86.
4. Harshavardhan G. Naganur, Sanjeev S. Sannakki, Vijay S Rajpurohit, Arunkumar R, “Fruits Sorting and Grading using Fuzzy Logic,” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 6, August 2012, pp 117-122.
5. John B. Njoroge. Kazunori Ninomiya. Naoshi Kondo and Hideki Toita, “Automated Fruit Grading System using Image Processing,” The Society of Instrument and Control Engineers(SICE2002), Osaka, Japan, August 2002, pp 1346-1351.
6. J. V. Frances, J. Calpe, E. Soria, M. Martinez, A. Rosado, A. J. Serrano, J. Calleja, M. Diaz, “Application of ARMA modeling to the improvement of weight estimations in fruit sorting and grading machinery,” IEEE 2000, pp 3666-3669
7. Wong Bing Yit, Nur Badariah Ahmad Mustafa, Zaipatimah Ali, Syed Khaleel Ahmed, Zainul Abidin Md Sharrif, “Design and Development of a Fully Automated Consumer-based Wireless Communication System For Fruit Grading”, ISCIT 2009, pp 364- 369.
8. Naoshi Kondo, “Fruit Grading Robot”, Proceedings of the2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), pp 1366-1371.
9. R. C. Gonzalez, R. E. Woods," Digital Image Processing", Pearson Education.IIEd.,2002

**PICTURE GALLERY**

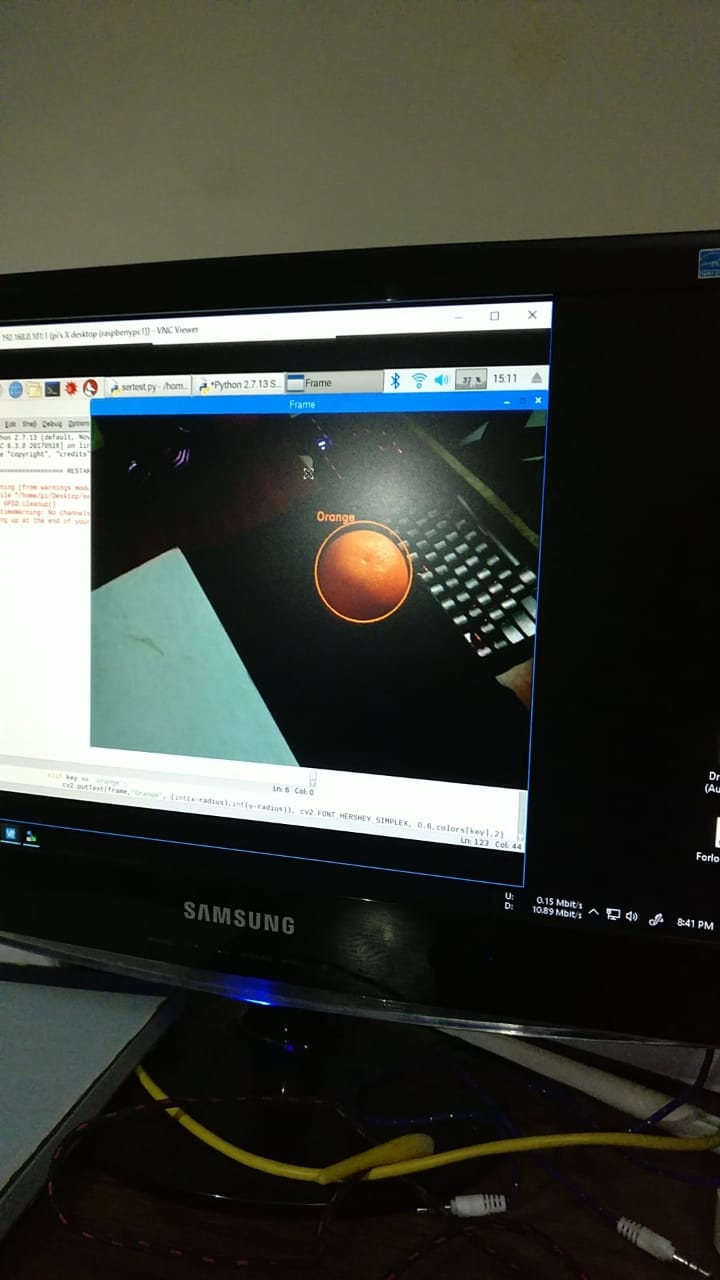
****

Fig: Detection of orange

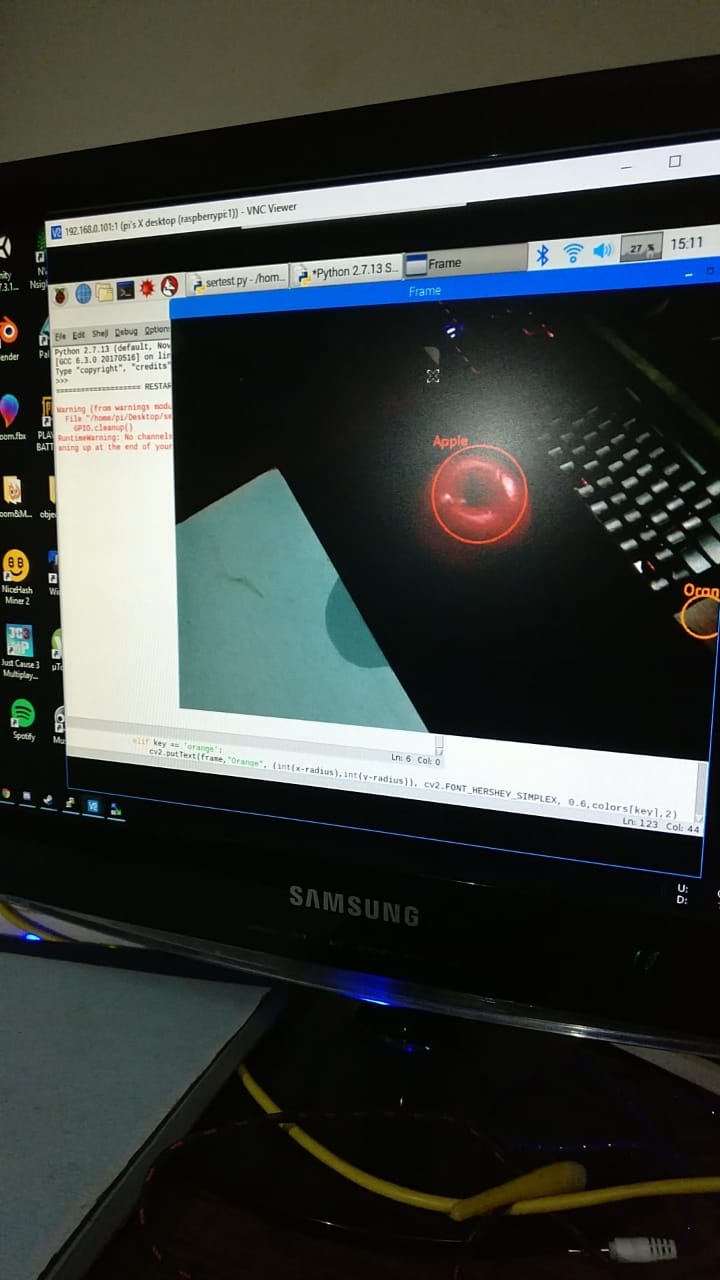
****

Fig: Detection of Apple