# “F RUIT RIPENESS DETECTOR”

Submitted in partial fulfillment of the requirement of CMR College of engineering and technology

(Autonomous)
**(Information Technology)**

**By**

| | |
|---|---|
| **1)K.ANURAG** | **“ID No: 17414”** |
| **2)R. VENU** | **“ID No: 17436”** |
| **3)K. CHANDRIKA** | **“ID No:17437”** |
| **4)A.SANDHYA GAYATRI** | **“ID No:17141”** |
| **5)KASHISH SINGHAL** | **“ID No:17447”** |
| **6)V.SHASHIKANTH** | **“ID No:17444”** |

**Department of Computer Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

(NAAC Accredited with ‘A+’ Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)

KANDLAKOYA, MEDCHAL ROAD HYDERABAD-501401

**2022-2023**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**
**(Autonomous)**
(NAAC Accredited with 'A+' Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)
KANDLAKOYA, MEDCHAL ROAD HYDERABAD-501401
**2022-2023**

# CERTIFICATE

*This is to certify that*

| | |
|---|---|
| **1)K.ANURAG** | **"ID No: 17414"** |
| **2)R. VENU** | **"ID No: 17436"** |
| **3)K. CHANDRIKA** | **"ID No:17437"** |
| **4)A.SANDHYA GAYATRI** | **"ID No:17141"** |
| **5)KASHISH SINGHAL** | **"ID No:17447"** |
| **6)V.SHASHIKANTH** | **"ID No:17444"** |

# INDEX

# Introduction

In order to combat global warming, plants have emerged as a major source of energy. Automation now has a significant impact across many industries. population in the majority in India's economy is mostly dependent on agriculture, and its basic source of income. India's local fresh fruit Exports have grown. Agriculture is regarded as an art form. Agriculture is also simpler and more productive thanks to science. Enable farmers to produce more during the last stage of time for growth. As both worldwide and domestic competition has intensified. It is necessary to expand domestic fruit markets. Techniques for assessing ripeness to cut down on loss occurred to the grower and packer as a result of fruit rotting.

Fruits can currently be found both in their natural and artificially processed forms. Among the processed forms are jam, paste, and juice. The export of those processed meals increases the nation's revenue.

People only prefer fresh, high-quality fruit because they are so concerned about their health. Fruit quality must be good in order to produce processed goods of high grade.

The primary obstacle is physically separating fruits of excellent and bad quality in industries since it takes a lot of time and expensive labor. Therefore, the need to identify the quality of fruits is given significant significance in the fruit industry.

Technology-based solutions that boost output and benefits while cutting costs and time are in high demand in the agriculture sector. All stages of the industrial process—from pre-harvest through harvest and post-harvest—have been enhanced by automation of agricultural jobs, but fruit plucking and gathering is still the most crucial activity.

Using different methods, we can use OpenCV software to determine the fruit's quality. Lemons can be mature green, green, or yellow at different stages of ripening. The majority of them were based on color characteristics because fruit quality is significantly influenced by the fruit's hue. The majority of factors used to determine fruit maturity may be used to calculate the fruit's surface color.

The goal of the current effort is to analyze the fruit's age factor using the method indicated, which aims to reduce labor requirements while maintaining the highest level of accuracy. The method is also suggested since it allows the process to move more quickly than the manual approach.

## PROBLEM STATEMENT

Fruit Ripeness Detection is developed in Python using OpenCV library. This project meets the results by the means of Image processing. In computer vision, OpenCV is a sizable open-source library for image processing, machine learning, and computer vision. It now plays a significant part in real-time operation, which is crucial in modern systems. Using it, one may analyze pictures and movies to find faces, objects, and even human handwriting.

### Proposed System:

The proposed system includes Automatic detection of the ripeness of fruits by just seeing the color of the fruits. Therefore, making it easier as well as cheaper. This uses OpenCV to process the image.

We use OpenCV, a free to use yet powerful Computer Vision Library which supports Python.
Based on the findings, the RGY ratio threshold for ripe fruit is fixed as greater than 9000, and that for unripe fruit is less than 9000. The majority of unripe fruits are typically green in color. Depending on the fruit we choose, we can alter the RGY value. As a result, in the programmed below, when we show the fruit through the webcam, the fruit in green is presented as unripe while the fruit in all other colors is displayed as ripe.

The Canny edge detector is an edge detection operator that uses a multi stage algorithm to detect a wide range of edges in images.

### Recommended System Requirements:

- Operating System:
    - Windows 10
    - Mac OS 10.14 and above
    - Ubuntu 18 and above
- Intel Core i5 or higher Processor,
- 8 GB of RAM or higher
- Python compiler with needed libraries,
- Text Editor.

### Technologies and Tools Used:

- VS Code,
- Python,
- OpenCV.

# IMPLEMENTATION

```
import io

import os

import random

import cv2

import numpy as np

import time

from copy import deepcopy

kernelOpen=np.ones((5,5))

kernelClose=np.ones((20,20))

# The name of the image file to annotate

i=time.strftime("%d-%m-%y_%H-%M-%S")

# capture image

camera = cv2.VideoCapture(0)

return_value, image = camera.read()

cv2.imwrite(i+'.jpeg', image)

del(camera)

frame=image

edge_img=deepcopy(image)

# finds edges in the input image image and

# marks them in the output map edges

edged = cv2.Canny(edge_img,50,100)

edged = cv2.dilate(edged, None, iterations=1)

edged = cv2.erode(edged, None, iterations=1)

# find contours in the edge map

#_,cnts, h = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

cnts, h = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
max_contA=cv2.contourArea(cnts[0])

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

for i in range(len(cnts)):

    x,y,w,h=cv2.boundingRect(max_cont)

    cv2.rectangle(edge_img,(x,y),(x+w,y+h),(0,0,255), 2)

croppedk=frame[y:y+h,x:x+w]

# Display the fruit

cv2.imshow('Edges',edge_img)

frame=edge_img

# converting BGR to HSV

hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# define range of red color in HSV

lower_red = np.array([0,50,50])

upper_red = np.array([10,255,255])

# create a red HSV colour boundary and

# threshold HSV image

redmask1 = cv2.inRange(hsv, lower_red, upper_red)

# define range of red color in HSV

lower_red = np.array([170,50,50])

upper_red = np.array([180,255,255])

# create a red HSV colour boundary and

# threshold HSV image

redmask2 = cv2.inRange(hsv, lower_red, upper_red)

redmask=redmask1+redmask2

maskOpen=cv2.morphologyEx(redmask,cv2.MORPH_OPEN,kernelOpen)

maskClose=cv2.morphologyEx(maskOpen,cv2.MORPH_CLOSE,kernelClose)

maskFinal=maskClose
```

```
cv2.imshow('Red_Mask:',maskFinal)

cnt_r=0

for r in redmask:

    cnt_r=cnt_r+list(r).count(255)

print ("Redness ",cnt_r)

lower_green=np.array([50,50,50])

upper_green=np.array([70,255,255])

greenmask = cv2.inRange(hsv, lower_green, upper_green)

cv2.imshow('Green_Mask:',greenmask)

cnt_g=0

for g in greenmask:

    cnt_g=cnt_g+list(g).count(255)

print ("Greenness ",cnt_g)

lower_yellow=np.array([20,50,50])

upper_yellow=np.array([30,255,255])

yellowmask = cv2.inRange(hsv, lower_yellow, upper_yellow)

cv2.imshow('Yellow_Mask:',yellowmask)

cnt_y=0

for y in yellowmask:

    cnt_y=cnt_y+list(y).count(255)

print ("Yellowness ",cnt_y)

#Calculate ripeness

tot_area=cnt_r+cnt_y+cnt_g

rperc=cnt_r/tot_area

yperc=cnt_y/tot_area

gperc=cnt_g/tot_area

#Adjust the limits for your fruit
```

```
glimit=0.5
ylimit=0.8
if cnt_g>9000:
    print("unripen")
else:
    print("ripen")
while True:
    k = cv2.waitKey(5) & 0xFF
    if k==27:
            break
# De-allocate any associated memory usage
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## By using Android

**JAVA CODE:**

```
package com.example.ripenessditector;
import android.Manifest;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;


public class MainActivity extends Activity {

    private static final int CAMERA_PERMISSION_REQUEST_CODE = 100;

    private static final int CAMERA_REQUEST = 1888;

    private ImageView imageView;

    private TextView rgbValuesTextView;

    String p;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Button btnOpenCamera = findViewById(R.id.btnOpenCamera);

        imageView = findViewById(R.id.imageView);

        rgbValuesTextView = findViewById(R.id.rgbValuesTextView);

        btnOpenCamera.setOnClickListener(new View.OnClickListener() {

            @Override

            public void onClick(View v) {

                // Check for camera permission before opening the camera

                if (checkCameraPermission()) {

                    openCamera();

                } else {

                    requestCameraPermission();
                }
            }
        });
    }

    private boolean checkCameraPermission() {
        return ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED;
    }
    private void requestCameraPermission() {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
CAMERA_PERMISSION_REQUEST_CODE);
    }
    private void openCamera() {
        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(cameraIntent, CAMERA_REQUEST);
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                openCamera();
            } else {
                // Permission denied, handle accordingly (e.g., show a message)
            }
        }
```

```
}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {

        Bitmap photo = (Bitmap) data.getExtras().get("data");

        imageView.setImageBitmap(photo);

        // Define the coordinates of the region you want to analyze (example: a square region at the center)

        int startX = photo.getWidth() / 4;

        int startY = photo.getHeight() / 4;

        int endX = photo.getWidth() * 3 / 4;

        int endY = photo.getHeight() * 3 / 4;

        // Variables to count the number of greenish and yellowish pixels

        int greenishPixelCount = 0;

        int yellowishPixelCount = 0;

        // Iterate through the selected region

        for (int x = startX; x < endX; x++) {

            for (int y = startY; y < endY; y++) {

                int pixelColor = photo.getPixel(x, y);

                int red = Color.red(pixelColor);

                int green = Color.green(pixelColor);

                int blue = Color.blue(pixelColor);

                // Check if the pixel appears greenish

                if (green > red + blue) {

                    greenishPixelCount++;

                }

                // Check if the pixel appears yellowish
                if (red > green && green > blue) {
                    yellowishPixelCount++;
                }
            }
        }
        int greenishThreshold = 100; // Tweak this value based on your requirements
        int yellowishThreshold = 100; // Tweak this value based on your requirements
        String result;
        if (greenishPixelCount > greenishThreshold) {
            result = "UNRIPEN";
        } else {
            result = "RIPEN";
        }
        // Display the result
        rgbValuesTextView.setText(result);
    }
  }
}
```

**XML FILE:**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">
    <Button
```

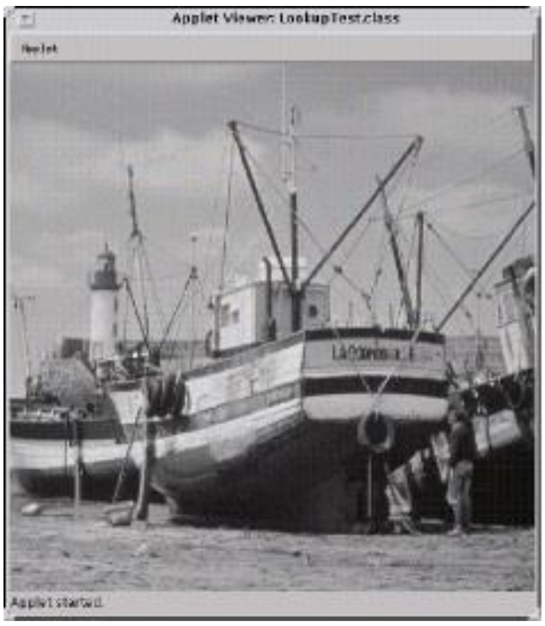
```
        android:id="@+id/btnOpenCamera"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="Open Camera"

        android:layout_centerHorizontal="true"

        android:layout_centerVertical="true" />

    <ImageView

        android:id="@+id/imageView"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:layout_below="@+id/btnOpenCamera"

        android:layout_marginTop="16dp"

        android:scaleType="fitCenter" />

    <TextView

        android:id="@+id/rgbValuesTextView"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_below="@+id/imageView"

        android:layout_centerHorizontal="true"

        android:layout_marginTop="16dp"

        android:text="RGB Values: "

        android:textSize="18sp" />

</RelativeLayout>
```

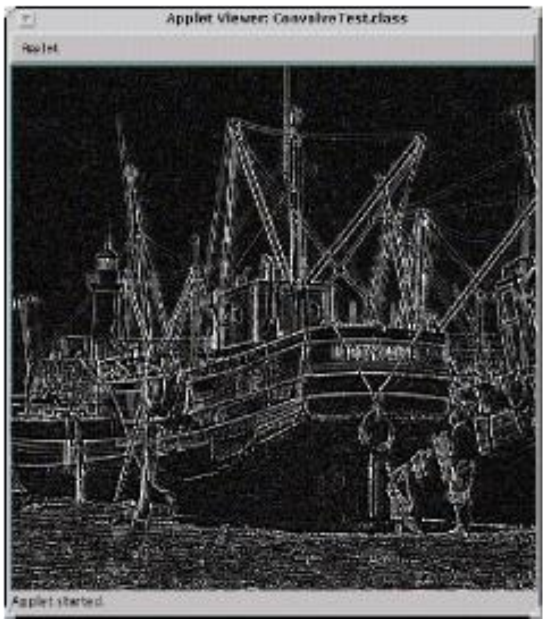
**MANIFEST FILE:**

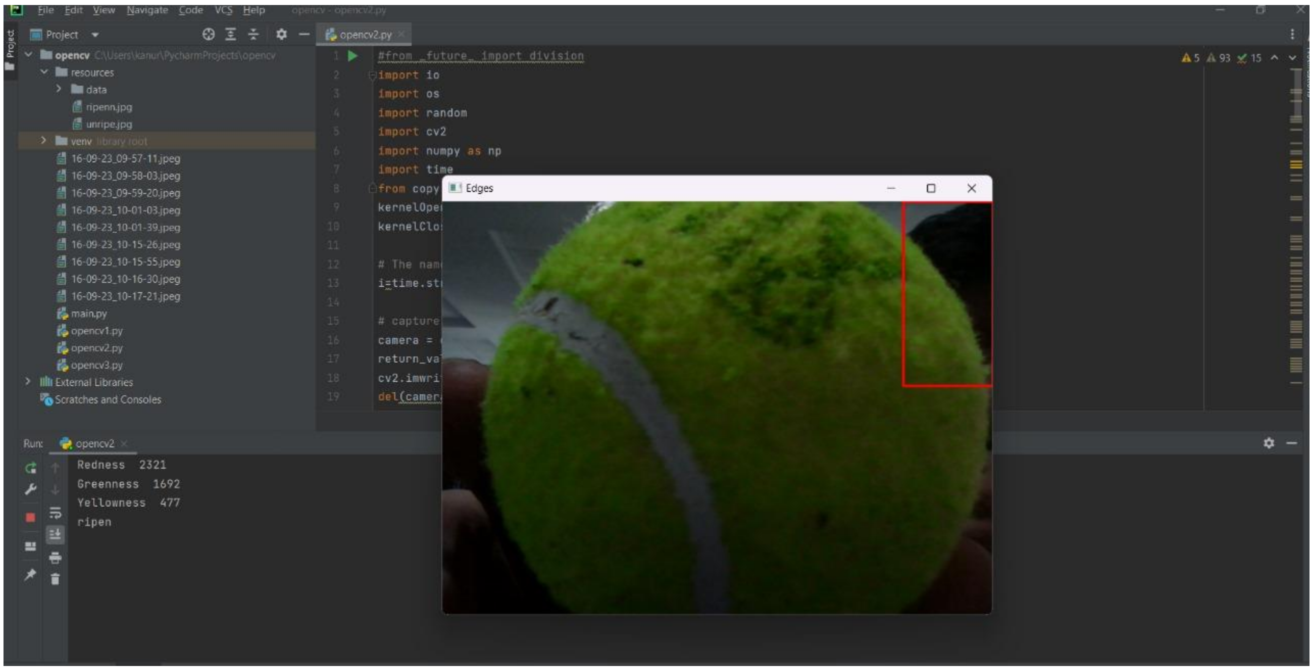
```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools">
```

```
<uses-permission android:name="android.permission.CAMERA" />

<application

    android:allowBackup="true"      android:dataExtractionRules="@xml/data_extraction_rules"

    android:fullBackupContent="@xml/backup_rules"

    android:icon="@mipmap/ic_launcher"

    android:label="@string/app_name"

    android:roundIcon="@mipmap/ic_launcher_round"

    android:supportsRtl="true"

    android:theme="@style/Theme.RipenessDitector"

    tools:targetApi="31">

    <activity

        android:name=".MainActivity"

        android:exported="true">

        <intent-filter>

            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />

        </intent-filter>  </activity> </application></manifest>
```

## Snapshots

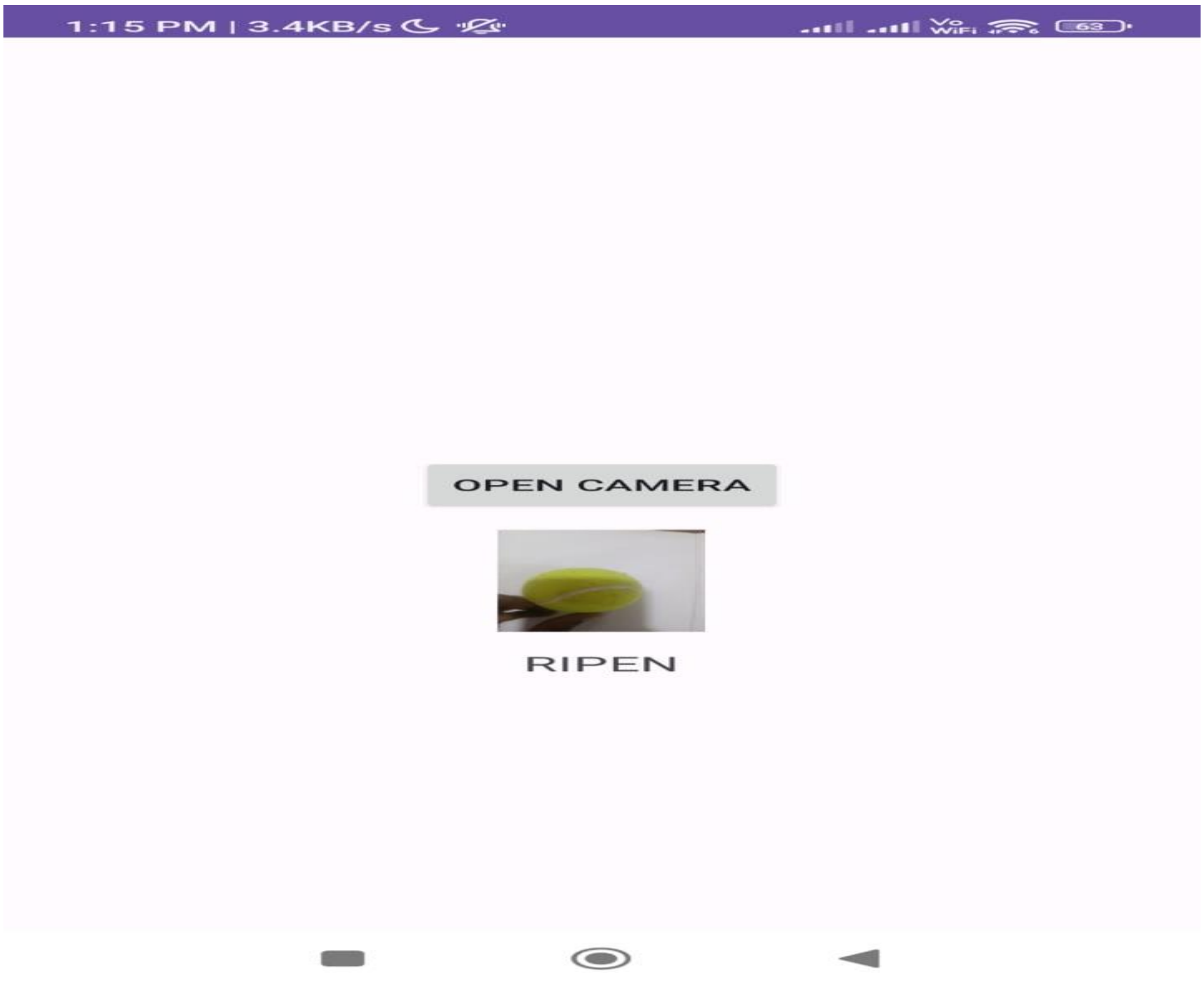

Applet Viewer: LookupTest.class
Applet
Applet started.


Applet Viewer: ConvolveTest.class
Applet
Applet started.



```
#from _future_ import division
import io
import os
import random
import cv2
import numpy as np
import time
from copy
kernelOpe
kernelClo

# The nam
i=time.st

# capture
camera = 
return_va
cv2.imwri
del(camer
```

```
Redness  2321
Greenness  1692
Yellowness  477
ripen
```



# Conclusion

Our project fully meets the objectives specified of the user for which it has been developed. The application operates at a high level of efficiency and the users associated with the system understand its advantage.

In order to get completed, finding what were the key concepts and what was the roles of them on the project's different parts was necessary. Outline of the major milestones that would be involved in completing the project, and worked backwards to break down the work that would need to be done at each stage was the procedure followed. After work and research, the project is now completely debugged and runnable.

By delivering this project, diseases classification techniques used for  fruit ripeness detection and canny algorithm for edge detection that can be used for automatic detection as well as classification of fruits. OpenCV is a sizable open-source library for image processing, machine learning, and computer vision.

In doing so we used Python as it combines elements of highly portable, syntax understandability, safety and security. Although the actual focus lies on the learning from the project. We expect that in the future there could be many other image detection programs could be developed using the knowledge of this project. Extending this project by linking it to other resources or creating new features and functions will maximize the added value of the hard work and time given in development, which will increase sustainable future aspects.