# Bottle Recognition

## 1. Introduction

This document specifies the “Bottle Recognition” project specifications and work flow, in order to create the first milestone of the autonomous Beach Cleaning Robot.

This project is based on 2 main features:

- HSV Masking & Edge Detection Contours (Only Coca Cola/Red Bottles)
- Machine Learning Bottle Detection (All bottles).

Project Tools & Hardware:

- Python3
- OpenCV 4
- TensorFlow & TensorFlow LITE
- Teachable Machine (by google)
- Raspberry Pi 4 (2 GB RAM)
- Pi Camera v2






## 2. HSV Masking & Edge Contours

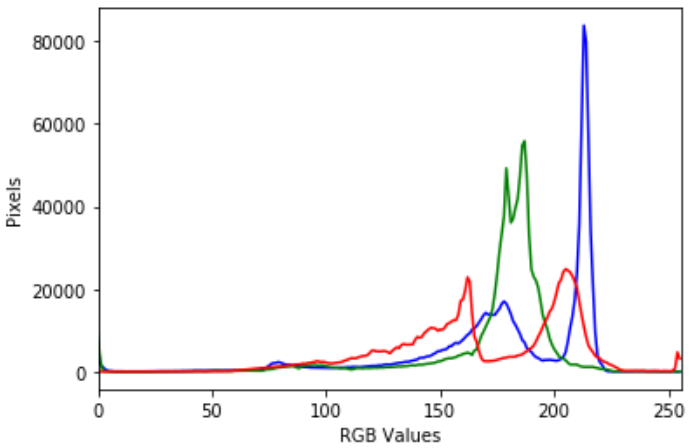
In order to recognize a “Coca Cola” bottle I have used the HSV Color space and extracted the right values to define the “Lower Red Threshold” and “Upper Red Threshold”, after getting these values it is easy to create the right mask and eliminate all the values that is out of this red boundary limit.

After completing the mask, we can implement the “Canny Edge Detection” algorithm to restore the wanted contours on the original image.

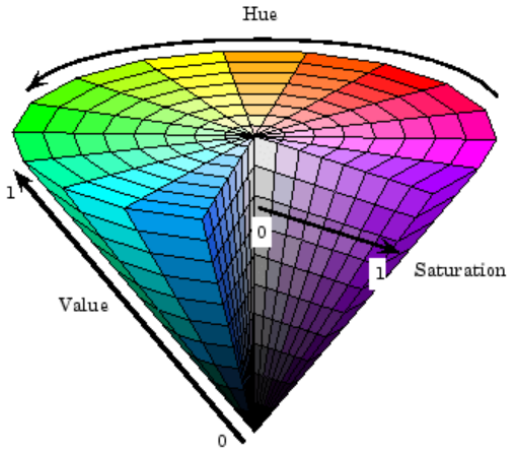
HSV Color space is more likely to use in this project rather than the RGB Color space, RGB values are scattered, therefore it is difficult to extract the wanted color from the image.

***Original Picture***

***RGB Plot***




*From this figure we can learn that the extraction of the values is difficult and not recommended to use the RGB Color space*




**Creating the wanted red range:**

**Lower Mask:**

$Lower\ Red\ Range\ H(Degrees°) = 0°$

$Upper\ Red\ Range\ H(Degrees°) = 20°$

$Lower\ Red\ Range\ S(0 - 255) = 120$

$Upper\ Red\ Range\ S(0 - 255) = 255$

$Lower\ Red\ Range\ V(0 - 255) = 70$

$Upper\ Red\ Range\ V(0 - 255) = 255$

*Using the cv2.inRange function we can create the lower mask.*

**Upper Mask:**

$Lower\ Red\ Range\ H(Degrees°) = 340°$

$Upper\ Red\ Range\ H(Degrees°) = 360°$

$Lower\ Red\ Range\ S(0 - 255) = 120$

$Upper\ Red\ Range\ S(0 - 255) = 255$

$Lower\ Red\ Range\ V(0 - 255) = 70$

$Upper\ Red\ Range\ V(0 - 255) = 255$

*Using the cv2.inRange function we can create the upper mask.*

After implementing the desired masks, this output is presented:

***Implementing the masks***

After creating the right masks, now it's time to implement the "Canny Edge Detection" Algorithm in order to restore the detection to the original image.

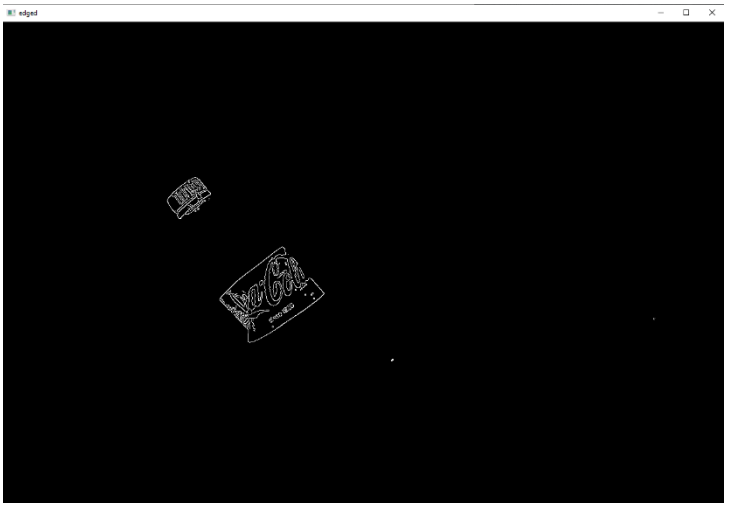
This process has 4 steps:

- Noise Reduction – 5x5 Gaussian Filter.
- Finding the Intensity Gradient of the Image $Edge_{Gradient(G)} = \sqrt{Gx^2 + Gy^2}$ $Angle(\theta) = (\frac{Gy}{Gx})$
- Non-maximum Suppression
- Hysteresis Thresholding

Dean Aviv Haklai

# BOTTLE RECOGNITION

## *Implementing Canny Edge Detection Algorithm*



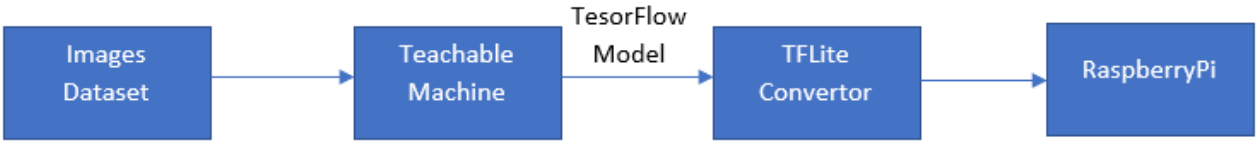

## *Restore Original Image*

# 3. Machine Learning Bottle Detection

In order create the model, I have used an online bottle images dataset (from Kaggle) and some own pictures of “nothing” to create at least 2 classes the model would predict.

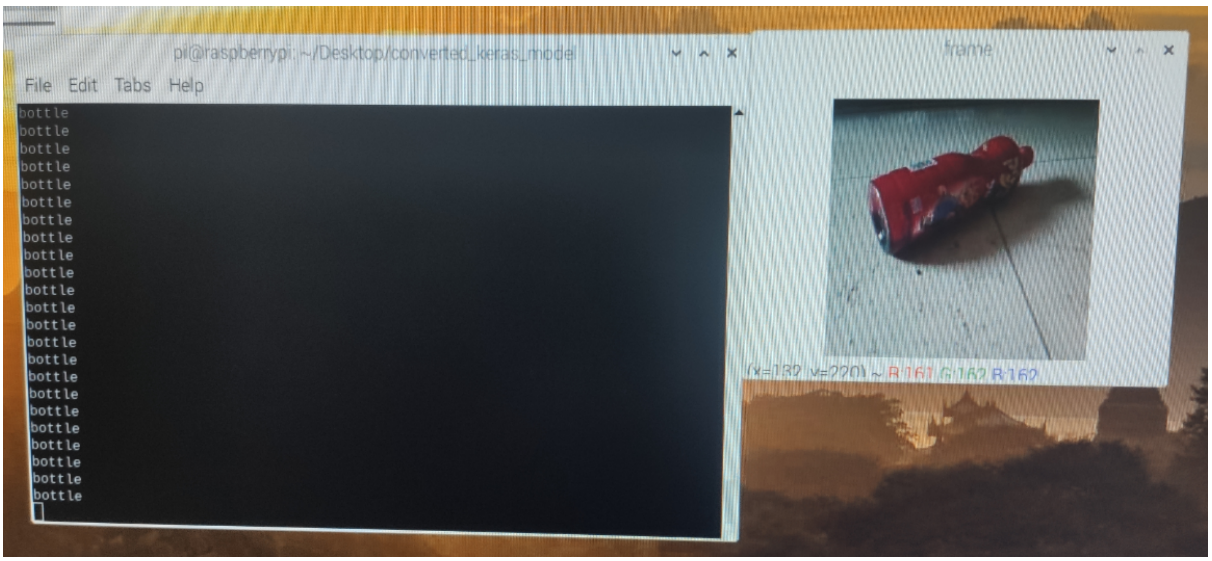
I have used the “Teachable Machine” (Online ML TensorFlow model creator by Google) in order to create exactly the model that is needed for this project.

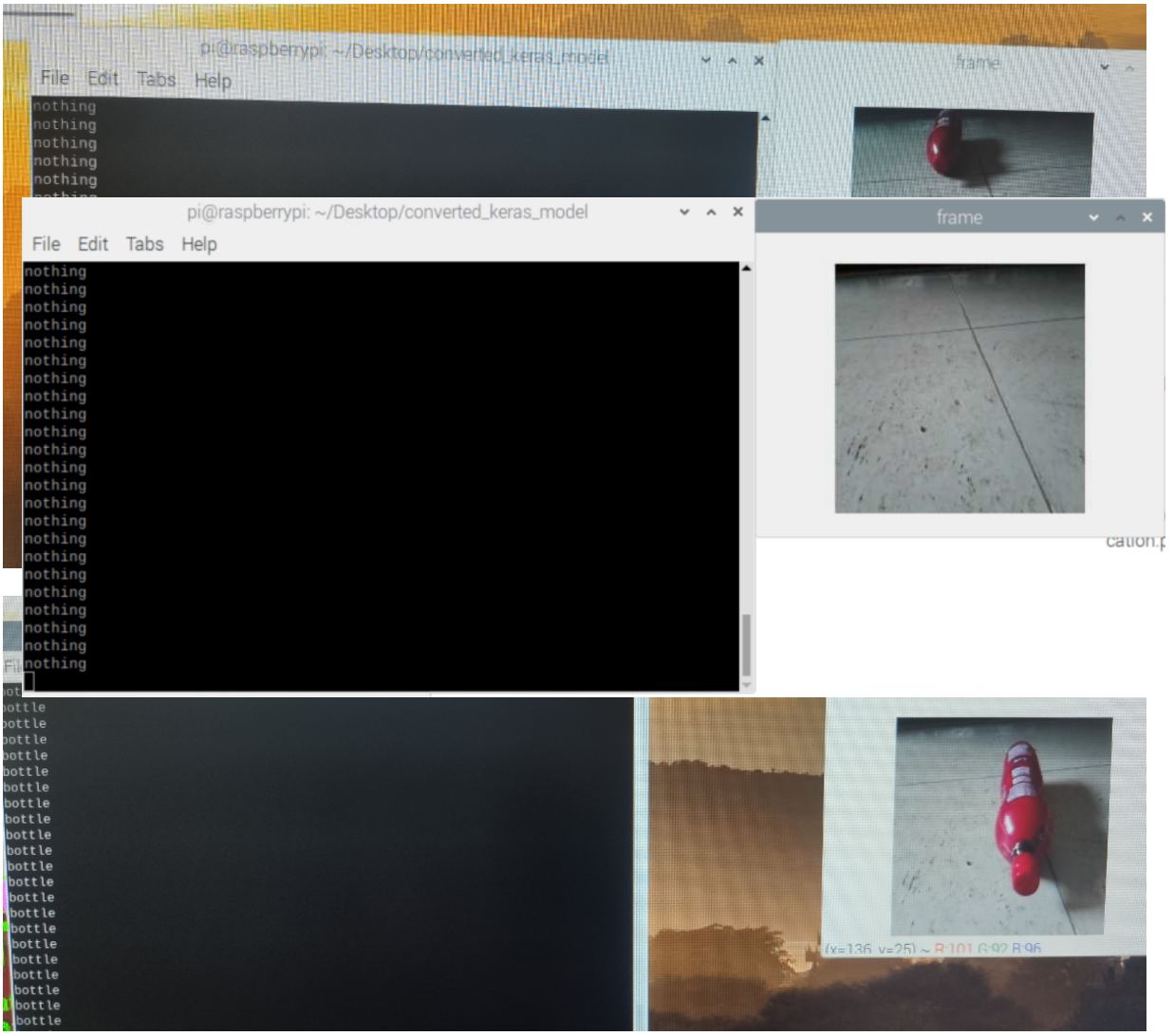
The TensorFlow CNN model that created is modified for Processors that can handle these kind of tasks, this project uses a Raspberry Pi (much less computing power, small processor) therefor I have needed to convert this model into a TFLite model that can run on small computing power consumption.

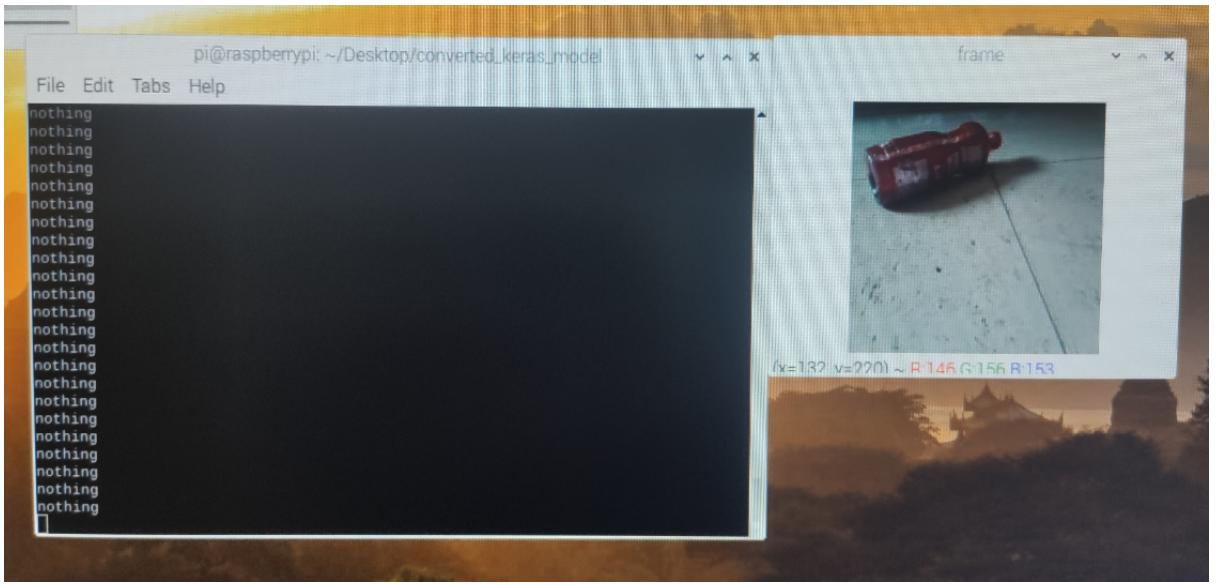



**Testing**:

As part of the project, I have been tested the model with the RaspberryPi (Camera attached) with different angles and distances.






We can learn from these tests that the model responding correctly for certain distances and angles, of course it depends on the dataset images background and distances.

**Success ratings (Ideal lightning):**

**Bottle Recognition**

Below 30 cm – approximately 85% success (no depending on the angle)

Above 30 cm – approximately 30% success (depending on the angle)

**Nothing Recognition**

100% success.

# 4. Conclusions

This project is part of the Autonomous Beach Cleaning Robot, all of the results in this sheet will take in count in the main project.

For example:

- Camera Angle.
- Distance to bottle.
- Canny Edge Detection to get the relative position to the robot.

# 5. Resources


- Canny Edge Detection
  https://en.wikipedia.org/wiki/Canny_edge_detector
  https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

- Teachable Machine
  https://teachablemachine.withgoogle.com/

- HSV
  https://en.wikipedia.org/wiki/HSL_and_HSV
  https://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html

- TensorFlow & TensorFlow Lite
  https://www.tensorflow.org/lite/tutorials
  https://www.tensorflow.org/tutorials/keras/save_and_load

# 6. Code

## 6.1. HSV-cv2

```
import cv2
import numpy as np
from matplotlib import pyplot as plt



while True:


    frame = cv2.imread('C:/Users/Roie/Desktop/bottle_HSV/1.jpg')
```

```
color = ('b','g','r')
for i,col in enumerate(color):
    histr = cv2.calcHist([frame],[i],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.xlabel("RGB Values")
plt.ylabel("Pixels")
plt.show()
blurred_frame = cv2.GaussianBlur(frame,(5,5),0)

#convert to hsv
hsv = cv2.cvtColor(blurred_frame,cv2.COLOR_BGR2HSV)



hist = cv2.calcHist([hsv],[0,1],None,[180,256],[0,180,0,256])



#range for lower red mask
#lower red range
l_r = np.array([0,120,70])
#upper red range
u_r = np.array([10,255,255])
mask1 = cv2.inRange(hsv,l_r,u_r)

#range for upper red mask
l_r = np.array([170,120,70])
u_r = np.array([180,255,255])
mask2 = cv2.inRange(hsv,l_r,u_r)

#final mask
final_mask = mask1+mask2
```

```
res = cv2.bitwise_and(frame,frame,mask = final_mask)
edged = cv2.Canny(res, 30, 200)
contours, hierarchy = cv2.findContours(edged,
                cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)


cv2.drawContours(frame, contours, -1, (0, 255, 0), 3)




cv2.drawContours(frame,contours,-1,(0,255,0),3)
print(contours)
cv2.imshow("frame",frame)
cv2.imshow("mask",final_mask)
cv2.imshow("res",res)
cv2.imshow("edged",edged)
plt.xlabel("HSV Values")
plt.ylabel("Pixels")
plt.hist(frame.ravel(),256,[0,256])
plt.show()

key = cv2.waitKey(1)
if key == 27:
   cv2.destroyAllWindows()
   break
```

## 6.2. RaspberryPi Python Script (TFLite & Model Prediction)

```
import numpy as np

import cv2

import tensorflow as tf

import threading

import json

import edgeDetection

import RPi.GPIO as GPIO


#set GPIO pins

GPIO.setmode(GPIO.BCM)

#blue

GPIO.setup(27,GPIO.OUT)
```

```
GPIO.output(27,False)

#red
GPIO.setup(22,GPIO.OUT)
GPIO.output(22,False)
lock = threading.Lock()
# Load the TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path="converted_model.tflite")
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()



# Test the model on random input data.
input_shape = input_details[0]['shape']
print(input_shape)
input_data = np.array(np.random.random_sample(input_shape), dtype=np.float32)
interpreter.set_tensor(input_details[0]['index'], input_data)

interpreter.invoke()

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

labels = ['bottle','can','nothing']

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.
output_data = interpreter.get_tensor(output_details[0]['index'])
```

```
bottle_counter = 0
bottle_state = 0


cap = cv2.VideoCapture(0)

def main():

  while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    b = cv2.resize(frame,(224,224),fx=0,fy=0, interpolation = cv2.INTER_CUBIC)
    # Our operations on the frame come here

    gray = cv2.cvtColor(b,1)
    normalized_image_array = (gray.astype(np.float32) / 127.0) - 1
    normalized_image_array.resize([1,224,224,3])

    input_data = normalized_image_array
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()


    output_data = interpreter.get_tensor(output_details[0]['index'])


    prediction = output_data
```

```
bottle = prediction[0,0]
cans = prediction[0,1]
nothing = prediction[0,2]

if nothing > 0.6:
    print('nothing')
    nothing_state = 1
    GPIO.output(27,True)

else:
    nothing_state = 0
    GPIO.output(27,False)

if bottle > 0.6:
    print('bottle')
    bottle_state = 1
    bottle_counter = bottle_counter+1
    GPIO.output(22,True)
    if bottle_counter > 20:
        cv2.imwrite("./Captures/bottle.jpg",frame)
        edgeDetection.main()
        bottle_counter = 0



else:
    GPIO.output(22,False)
    bottle_state = 0
    bottle_counter = 0
```

```
        lock.acquire()
        data = {"nothing_state":nothing_state,"bottle_state":bottle_state}
        with open('data2.txt','w') as outfile:
            json.dump(data,outfile)
        lock.release()

        # Display the resulting frame
        cv2.imshow('frame',gray)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            GPIO.output(22,False)
            GPIO.output(27,False)
            break


    # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()


main()
```

## 6.3. Sending Data to Arduino via BlueTooth

```
import time
import json
import serial
try:
    bluetoothSerial = serial.Serial( "/dev/rfcomm{bt}".format(bt=4), baudrate=9600 )
except NameError:
    print("Please connect to the Bluetooth")
```

```
while(True):
    time.sleep(0.2)
    try:
        with open('data2.txt','r') as json_file:
            data = json.load(json_file)



        #print(list(data.values()))


        values_list = list(data.values())
        nothing_msg = str(values_list[0])
        bottle_msg = str(values_list[1])
        msg = nothing_msg + bottle_msg
        print(msg)


        if msg == '01':
            flag_bottle = 101
            j = str('P')
            b = j.encode()
            bluetoothSerial.write(b)
            time.sleep(0.1)


        elif msg == '10':
            j = str('N')
            b = j.encode()
            bluetoothSerial.write(b)
            time.sleep(0.1)


    except:
```

```
import testing
```