Computer Vision Assignments

CONTENTS		
1.Basic Functions		Page 3- 13
	1.1 Displaying Image	Page 3
	1.2 Drawing a Line on a Image	Page 4
	1.3 Drawing a Circle on a Image	Page 5
	1.4 Drawing a Rectangle on a Image	Page 6
	1.5 String on a Image	Page 7
	1.6 Colour Conversion- Gray Scale	Page 8
	1.7 Colour Conversion- HSV	Page 9
	1.8 Drawing a Border on a Image	Page 10
	1.9 Image as Border	Page 11
	1.10 Gaussian Blur vs Median Blur vs Bilateral Blur	Page 12
2. Moving Object Detection		Page 14-15
3. Harry Potter's Invisible Cloak		Page 16-17
4. Leaf Disease Detection		Page 18- 20
5. Brain Tumor Detector Matlab		Page 21- 25

1.BASIC FUNCTIONS

1.1 DISPLAYING IMAGE

Code

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

def display_image(image_path):
    # Read the image
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image1.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results

1.2 Drawing a line on a image

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Draw a line on the image
    start_point = (300, 500)
    end_point = (780, 0)
    color = (255, 0, 0)  # Color for the line (blue in BGR)
    thickness = 75
    image_with_line = cv2.line(image.copy(), start_point, end_point, color, thickness)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_line, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image1.jpeg"

# Call the function to display the image
display_image(image_path)
```

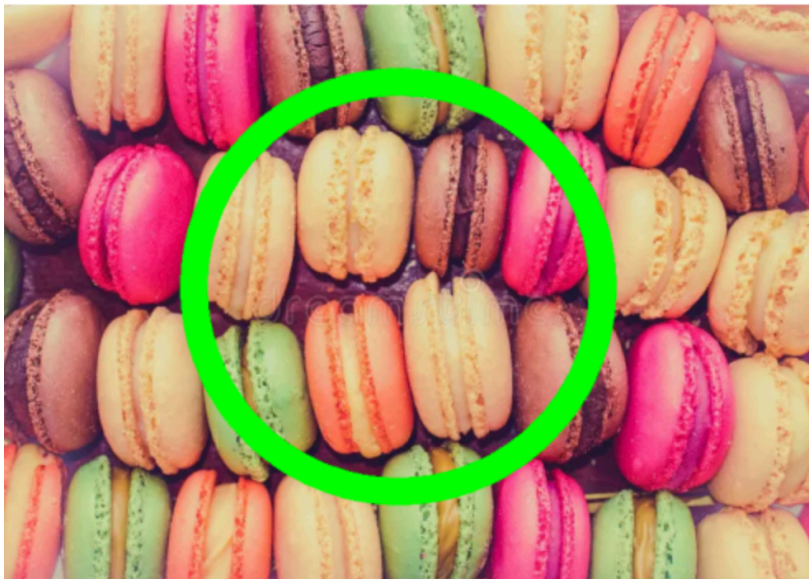
Results

1.3 Drawing a circle on a image

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Draw a circle on the image
    center_point=(390,275)
    radius=200
    color = (0, 255, 0)  # Color for the line (green in BGR)
    thickness = 25
    image_with_circle = cv2.circle(image.copy(), center_point, radius, color, thickness)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_circle, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image1.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results



1.4 Drawing a rectangle on a image

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Draw a rectangle on the image
    start_point = (150, 500)
    end_point = (720, 70)
    color = (0, 0, 255)  # Color for the line (blue in BGR)
    thickness = 25
    image_with_rectangle = cv2.rectangle(image.copy(), start_point, end_point, color, thickness)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_rectangle, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image1.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results

1.5 String on a image

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # string on the image
    font=cv2.FONT_HERSHEY_SIMPLEX
    org=(50,250)
    fontScale=6
    color = (255, 0, 0)
    thickness = 10
    image_with_text = cv2.putText(image.copy(), 'Cookies', org, font, fontScale, color,
thickness,cv2.LINE_AA)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_text, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image1.jpeg"

# Call the function to display the image
display_image(image_path)
```

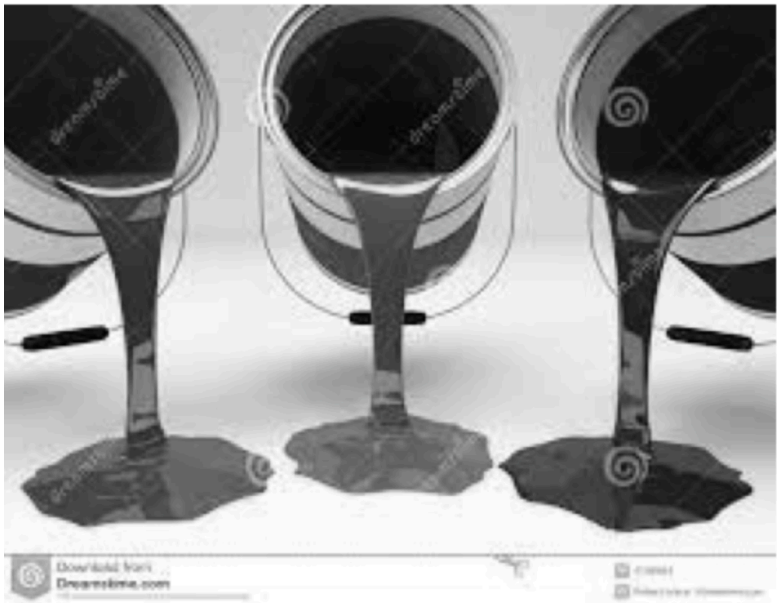
Results

1.6 Colour Conversion- Gray Scale

Code

```
def display_image(image_path):
    # Read the image
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

     # Convert BGR to Grayscale
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Display the image using matplotlib with grayscale colormap
    plt.imshow(image_gray, cmap='gray')
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image2.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results

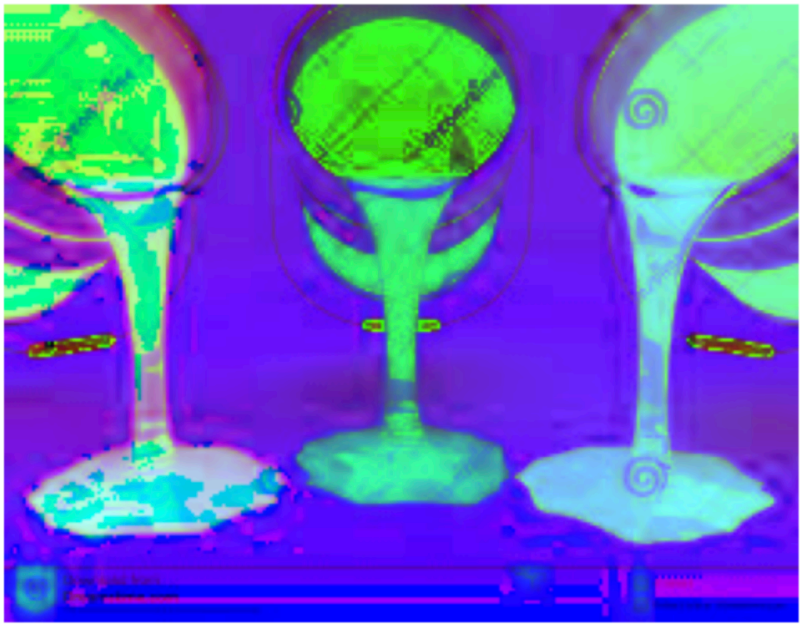


1.7 Colour Conversion- HSV

Code

```
def display_image(image_path):
    # Read the image
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Convert BGR to RGB
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Convert RGB to HSV
    image_hsv = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2HSV)

    # Display the image using matplotlib
    plt.imshow(image_hsv,cmap='hsv')
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image2.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results



1.8 Drawing a border on a image

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Draw a border on the image
    image_with_border = cv2.copyMakeBorder(image.copy(),
35,20,35,10,cv2.BORDER_CONSTANT, None, value=1)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_border, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image2.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results

1.9 Drawing image as a border

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    # Draw a border on the image
    image_with_border = cv2.copyMakeBorder(image.copy(),
100,50,35,10,cv2.BORDER_REFLECT)

    # Convert BGR to RGB for displaying with matplotlib
    image_rgb = cv2.cvtColor(image_with_border, cv2.COLOR_BGR2RGB)

    # Display the image using matplotlib
    plt.imshow(image_rgb)
    plt.axis('off')  # Turn off axis numbers and ticks
    plt.show()

# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image2.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results

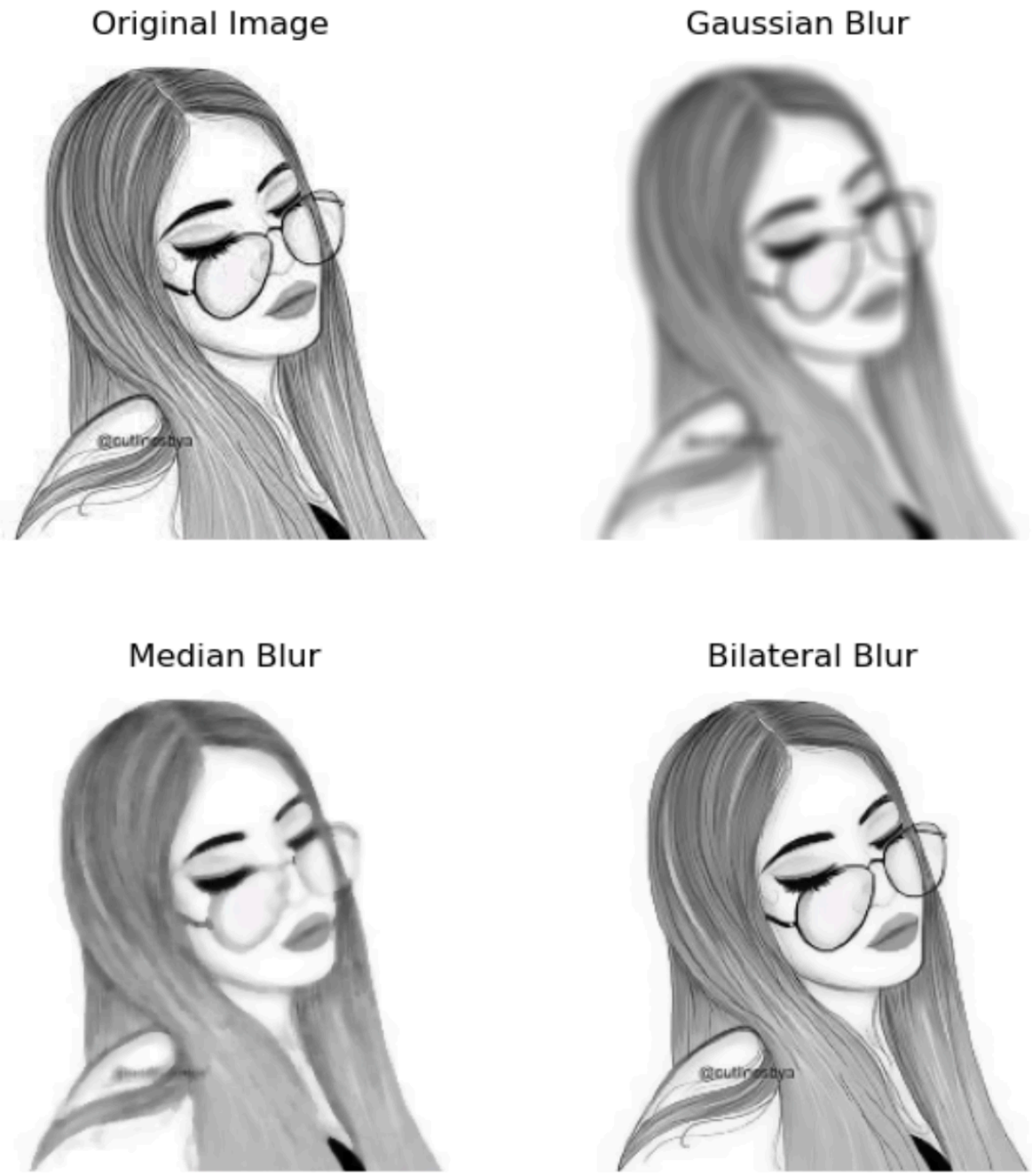
1.10 Gaussian Blur vs Median Blur vs Bilateral Blur

Code

```
def display_image(image_path):
    # Read the image in color
    image = cv2.imread(image_path)

    # Check if the image was successfully loaded
    if image is None:
        print("Error: Could not open or find the image.")
        return

    #Displaying original Image
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Apply Gaussian Blur on the image
    image_with_GBlur = cv2.GaussianBlur(image, (7, 7), cv2.BORDER_DEFAULT)


    # Convert BGR to RGB for displaying with matplotlib
    image_GBlur = cv2.cvtColor(image_with_GBlur, cv2.COLOR_BGR2RGB)


    # Apply Median Blur on the image
    image_with_MBlur = cv2.medianBlur(image, 5)


    # Convert BGR to RGB for displaying with matplotlib
    image_MBlur = cv2.cvtColor(image_with_MBlur, cv2.COLOR_BGR2RGB)


    # Apply Bilateral Blur on the image
    image_with_BBlur = cv2.bilateralFilter(image, 9,75,75)


    # Convert BGR to RGB for displaying with matplotlib
    image_BBlur = cv2.cvtColor(image_with_BBlur, cv2.COLOR_BGR2RGB)

    # Display the images using matplotlib
    fig, axes = plt.subplots(1, 4,  figsize=(15, 15))

    axes[0].imshow(image_rgb)
    axes[0].axis('off')  # Turn off axis numbers and ticks
    axes[0].set_title('Original Image')

    axes[1].imshow(image_GBlur)
    axes[1].axis('off')  # Turn off axis numbers and ticks
    axes[1].set_title('Gaussian Blur')

    axes[2].imshow(image_MBlur)
    axes[2].axis('off')  # Turn off axis numbers and ticks
    axes[2].set_title('Median Blur')

    axes[3].imshow(image_BBlur)
    axes[3].axis('off')  # Turn off axis numbers and ticks
    axes[3].set_title('Bilateral Blur')

    plt.show()
```

```
# Specify the path to your image
image_path = "/Users/dhayaneyv/Desktop/fl_image3.jpeg"

# Call the function to display the image
display_image(image_path)
```

Results



2. MOVING OBJECT DETECTION

Code

```
import numpy as np
import imutils
import cv2
import time

prototxt="/Users/dhayaneyv/Downloads/MobileNetSSD_deploy.prototxt.txt"
model="/Users/dhayaneyv/Downloads/MobileNetSSD_deploy.caffemodel"
confThresh=0.2

CLASSES=["background","aeroplane","bicycle","bird","boat","bottle","bus","car","cat","chair","c
ow","dinigtable","dog","horse","motorbike","person","pottedplant","sheep","sofa","train","tvmoni
tor"]
COLORS=np.random.uniform(0,255,size=(len(CLASSES),3))

print("Loading Model...")
net=cv2.dnn.readNetFromCaffe(prototxt,model)
print("Model Loaded")
print("Starting Camera Feed...")
vs=cv2.VideoCapture(0)
time.sleep(2.0)

while True:
    _,frame=vs.read()
    frame+imutils.resize(frame,width=1920)

    (h,w)=frame.shape[:2]
    imResizeBlob=cv2.resize(frame,(300,300))
    blob=cv2.dnn.blobFromImage(imResizeBlob,0.007843,(300,300),127.5)

    net.setInput(blob)
    detections = net.forward()
    detShape = detections.shape[2]
    for i in np.arange(0,detShape):
        confidence = detections[0, 0, i, 2]
        if confidence > confThresh:
            idx = int(detections[0, 0, i, 1])
            print("ClassID:",detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            label = "{}: {:.2f}%".format(CLASSES[idx],
                confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)
            if startY - 15 > 15:
                y = startY - 15
            else:
                startY + 15

            cv2.putText(frame, label, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

            if CLASSES[idx] == "bottle":
                cv2.putText(frame, "I need Water", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
255, 0), 2)
```

```
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1)
    if key == 27:
        break
vs.release()
cv2.destroyAllWindows()
```

Results

Output detecting Bottle with “I need Water”

3. Harry Potter's Invisible Cloak

Code

```
import cv2
import numpy as np
import time

# Initialize the video capture object
cap = cv2.VideoCapture(0)
time.sleep(3)

# Capture the background
background = 0
for i in range(30):
    ret, background = cap.read()
background = np.flip(background, axis=1)

while cap.isOpened():
    ret, img = cap.read()
    if not ret:
        break

    img = np.flip(img, axis=1)

    # Convert the image to HSV color space
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Define the range for maroon color in HSV space
    lower_maroon = np.array([160, 100, 20])
    upper_maroon = np.array([179, 255, 255])

    # Create a mask for the maroon color
    mask = cv2.inRange(hsv, lower_maroon, upper_maroon)

    # Morphological operations to remove noise
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, np.ones((5, 5), np.uint8))

    # Replace the maroon color with the background
    img[np.where(mask == 255)] = background[np.where(mask == 255)]

    # Display the result
    cv2.imshow('Display', img)

    # Exit if 'q' is pressed
    k = cv2.waitKey(10)
    if k == ord('q'):
        break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

Results

Image 0

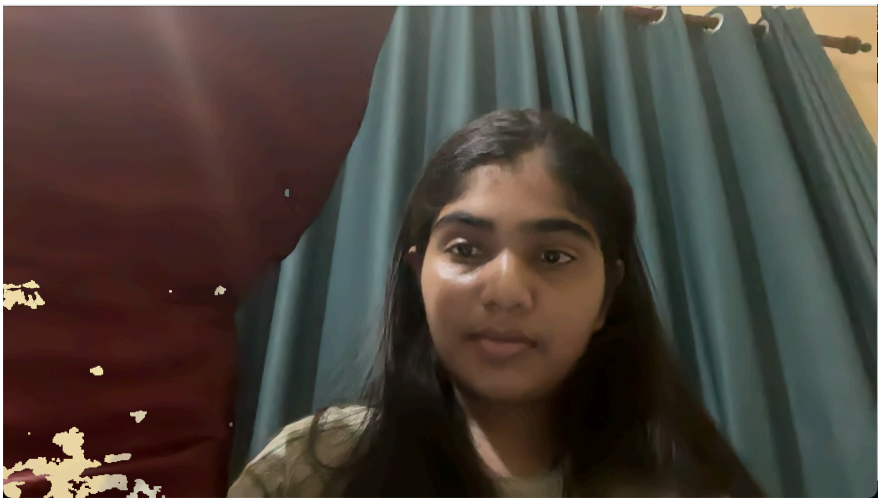


Image with Maroon Cloak

Image with Invisible Maroon Cloak

4. Leaf Disease Detection

Code

```
from keras.models import Sequential
#initialize nn

from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense

from keras.layers import BatchNormalization
from keras.layers import Dropout

#basic cnn
model = Sequential()
model.add(Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(128,128, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(96, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation = 'softmax'))

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])

train_datagen = ImageDataGenerator(rescale = None,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('/Users/dhayaneyv/Desktop/LeafDisease_2/
dataset/train',
                                                 target_size = (128, 128),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
#print(test_datagen);
labels = (training_set.class_indices)
print(labels)

test_set = test_datagen.flow_from_directory('/Users/dhayaneyv/Desktop/LeafDisease_2/dataset/
test',
```

```
                                target_size = (128, 128),
                                batch_size = 32,
                                class_mode = 'categorical')

labels2 = (test_set.class_indices)
print(labels2)

model.fit_generator(training_set,
                    steps_per_epoch = 375,
                    epochs = 1000000,
                    validation_data = test_set,
                    validation_steps = 125)


model_json=model.to_json()
with open("model1.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
    model.save_weights("model1.h5")
    print("Saved model to disk”)

import numpy as np
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
import os
import cv2

json_file = open('model1.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("model1.h5")
print("Loaded model from disk")


label=["Apple___Apple_scab","Apple___Black_rot","Apple___Cedar_apple_rust","Apple___Health
y",
      "Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot","Corn_(maize)___Common_rust_",
      "Corn_(maize)___Healthy","Corn_(maize)___Northern_Leaf_Blight","Grape___Black_rot",

"Grape___Esca_(Black_Measles)","Grape___Healthy","Grape___Leaf_blight_(Isariopsis_Leaf_Spot
)",

"Potato___Early_blight","Potato___Healthy","Potato___Late_blight","Tomato___Bacterial_spot",

"Tomato___Early_blight","Tomato___Healthy","Tomato___Late_blight","Tomato___Leaf_Mold",
      "Tomato___Septoria_leaf_spot","Tomato___Spider_mites Two-
spotted_spider_mite","Tomato___Target_Spot",
      "Tomato___Tomato_Yellow_Leaf_Curl_Virus","Tomato___Tomato_mosaic_virus"]

test_image = image.load_img('/Users/dhayaneyv/Desktop/LeafDisease_2/
im_for_testing_purpose/t.mosaicvirus.JPG', target_size = (128, 128))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = loaded_model.predict(test_image)
print(result)
#print(result)
```

```
fresult=np.max(result)
label2=label[result.argmax()]
print(label2)
```

Results

```
Found 375 images belonging to 25 classes.
{'Apple___Apple_scab': 0, 'Apple___Black_rot': 1, 'Apple___Cedar_apple_rust': 2, 'Apple___Healthy': 3, 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 4, 'Corn_(maize)___Common_rust': 5, 'Corn_(maize)___Healthy': 6, 'Corn_(maize)___Northern_Leaf_Blight': 7, 'Grape___Black_rot': 8, 'Grape___Esca_(Black_Measles)': 9, 'Grape___Healthy': 10, 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 11, 'Potato___Early_blight': 12, 'Potato___Healthy': 13, 'Potato___Late_blight': 14, 'Tomato___Bacterial_spot': 15, 'Tomato___Early_blight': 16, 'Tomato___Healthy': 17, 'Tomato___Late_blight': 18, 'Tomato___Leaf_Mold': 19, 'Tomato___Septoria_leaf_spot': 20, 'Tomato___Spider_mites Two-spotted_spider_mite': 21, 'Tomato___Target_Spot': 22, 'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 23, 'Tomato___Tomato_mosaic_virus': 24}
Found 127 images belonging to 25 classes.
{'Apple___Apple_scab': 0, 'Apple___Black_rot': 1, 'Apple___Cedar_apple_rust': 2, 'Apple___Healthy': 3, 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 4, 'Corn_(maize)___Common_rust_': 5, 'Corn_(maize)___Healthy': 6, 'Corn_(maize)___Northern_Leaf_Blight': 7, 'Grape___Black_rot': 8, 'Grape___Esca_(Black_Measles)': 9, 'Grape___Healthy': 10, 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 11, 'Potato___Early_blight': 12, 'Potato___Healthy': 13, 'Potato___Late_blight': 14, 'Tomato___Bacterial_spot': 15, 'Tomato___Early_blight': 16, 'Tomato___Healthy': 17, 'Tomato___Late_blight': 18, 'Tomato___Leaf_Mold': 19, 'Tomato___Septoria_leaf_spot': 20, 'Tomato___Spider_mites Two-spotted_spider_mite': 21, 'Tomato___Target_Spot': 22, 'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 23, 'Tomato___Tomato_mosaic_virus': 24}
Epoch 1/1000000
/var/folders/yk/4thk018s5h75b3sldp4192400000gn/T/ipykernel_4332/578943485.py:61: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(training_set,
 12/375 [..............................] - ETA: 41s - loss: 3.3215 - accuracy: 0.1013WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 375000000 batches). You may need to use the repeat() function when building your dataset.
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 125 batches). You may need to use the repeat() function when building your dataset.
375/375 [==============================] - 2s 4ms/step - loss: 3.3215 - accuracy: 0.1013 - val_loss: 3.2433 - val_accuracy: 0.0394
Saved model to disk
```

```
Loaded model from disk
1/1 [==============================] - 0s 40ms/step
[[1.23152553e-04 3.96056239e-05 3.83000411e-02 9.22000595e-07
  1.01418345e-13 2.06834719e-01 2.55822409e-02 7.91219146e-10
  8.02073941e-07 7.27908254e-01 6.21561824e-08 1.94143081e-06
  2.90114986e-06 1.57453606e-09 2.04618500e-05 5.02030258e-11
  3.37402307e-04 4.57990041e-07 1.00302833e-08 2.17412935e-06
  7.26999679e-06 8.16618340e-05 2.27949713e-05 8.01543898e-09
  7.33174500e-04]]
Grape___Esca_(Black_Measles)
```

5. Brain Tumor Detector-Matlab

Code

```
function varargout = Braintumordetector(varargin)
% BRAINTUMORDETECTOR MATLAB code for Braintumordetector.fig
%      BRAINTUMORDETECTOR, by itself, creates a new BRAINTUMORDETECTOR or raises the
existing
%      singleton*.
%
%      H = BRAINTUMORDETECTOR returns the handle to a new BRAINTUMORDETECTOR or
the handle to
%      the existing singleton*.
%
%      BRAINTUMORDETECTOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BRAINTUMORDETECTOR.M with the given input arguments.
%
%      BRAINTUMORDETECTOR('Property','Value',...) creates a new BRAINTUMORDETECTOR or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Braintumordetector_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Braintumordetector_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Braintumordetector

% Last Modified by GUIDE v2.5 23-May-2024 11:21:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Braintumordetector_OpeningFcn, ...
                   'gui_OutputFcn',  @Braintumordetector_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Braintumordetector is made visible.
function Braintumordetector_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

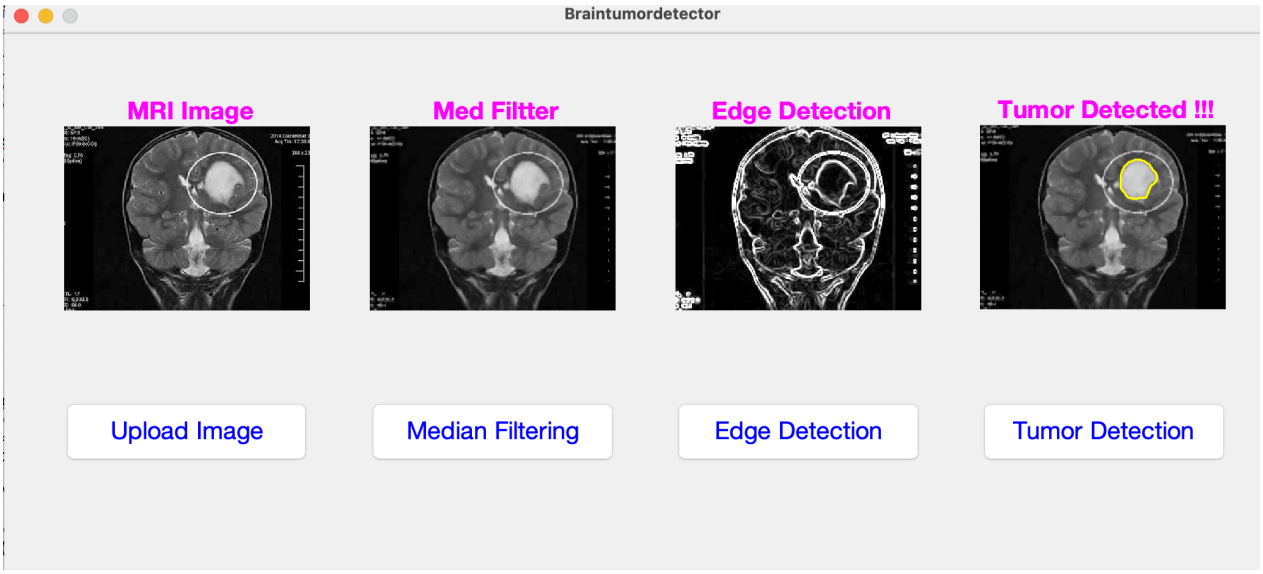
```
% varargin   command line arguments to Braintumordetector (see VARARGIN)

% Choose default command line output for Braintumordetector
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Braintumordetector wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Braintumordetector_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in select_image.
function select_image_Callback(hObject, eventdata, handles)
% hObject    handle to select_image (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global img1 img2

[path, nofile] = imgetfile();

if nofile
    msgbox(sprintf('Image not found!!!'), 'Error', 'Warning');
    return
end

img1 = imread(path);
img1 = im2double(img1);
img2 = img1;

axes(handles.axes1);
imshow(img1);
title('\fontsize{20}\color[rgb]{1,0,1} MRI Image');



% --- Executes on button press in median_filtering.
function median_filtering_Callback(hObject, eventdata, handles)
% hObject    handle to median_filtering (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global img1
axes (handles.axes2)
if size(img1,3) == 3
    img1 = rgb2gray(img1);
end
K = medfilt2(img1);
```

```
axes(handles.axes2);
imshow(K);title('\fontsize{20}\color[rgb]{1,0,1} Med Filtter');


% --- Executes on button press in edge_detection.
function edge_detection_Callback(hObject, eventdata, handles)
% hObject    handle to edge_detection (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global img1
axes(handles.axes3);

if size(img1,3) == 3
    img1 = rgb2gray(img1);
end
K = medfilt2(img1);
C = double(K);

for i = 1:size(C,1)-2
    for j = 1:size(C,2)-2
        %Sobel mask for X-direction
        Gx = ((2*C(i+2,j+1)+ C(i+2,j)+C(i+2,j+2))-(2*C(i,j+1)+C(i,j)+C(i,j+2)));
        %Sobel mask for Y-Dirction
        Gy = ((2*C(i+1,j+2)+ C(i,j+2)+C(i+2,j+2))-(2*C(i+1,j)+C(i,j)+C(i+2,j)));

        %The Gradient of The Image
        % B(i,j) abs(Gx) + abs(Gy)
        B(i,j) = sqrt(Gx.^2+Gy.^2);
    end
end
axes(handles.axes3)
imshow(B);title('\fontsize{20}\color[rgb]{1,0,1} Edge Detection');



% --- Executes on button press in tumor_detection.
function tumor_detection_Callback(hObject, eventdata, handles)
% hObject    handle to tumor_detection (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global img1
axes(handles.axes4);
K = medfilt2(img1);
bw = imbinarize(K, 0.7);
label = bwlabel(bw);

stats = regionprops(label,'Solidity','Area');
density  = [stats.Solidity];
area = [stats.Area];
high_dense_area = density > 0.5;
max_area = max(area(high_dense_area));
tumor_label = find(area == max_area);
tumor = ismember(label,tumor_label);

se = strel('square',5);
tumor = imdilate(tumor,se);

Bound = bwboundaries(tumor,'noholes');
```

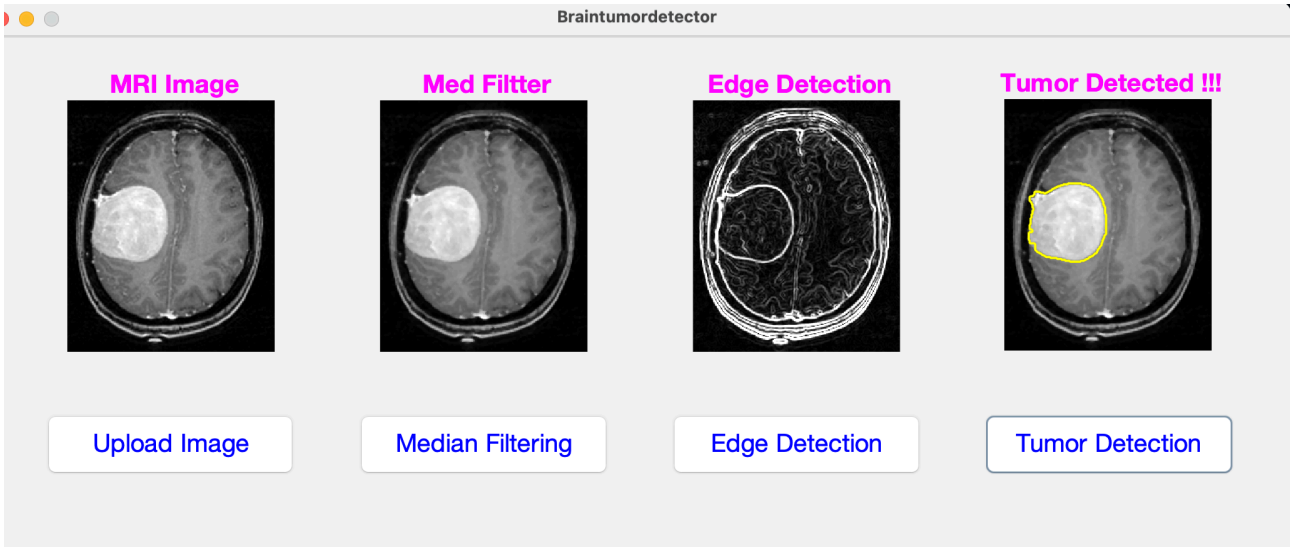
```
imshow(K);
hold on
for i = 1:length(Bound)
    plot(Bound{i}(:,2),Bound{i}(:,1), 'y','linewidth',1.75)
end
title('\fontsize{20}\color[rgb]{1,0,1} Tumor Detected !!!');

hold off
axes(handles.axes4)
```

Results

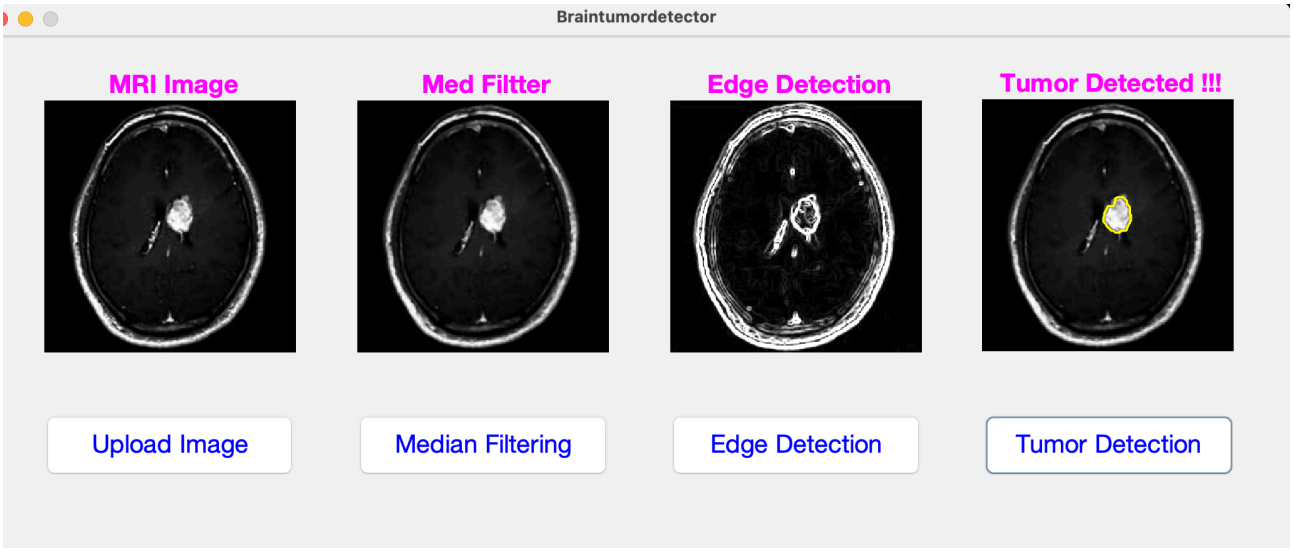
Test Image 1



Test Image 2



Test Image 3



Test Image 4