Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-2024	
Subject Name & Code: Image Processing: ADUA31205(B)		
Title of Assignment: Perform object detection from an image		
Date of Performance: 01-11-2023	Date of Submission: 11-11-2023	

ASSIGNMENT NO. 8

Name : Siddhesh Dilip Khairnar
PRN no : 2211039 8
Roll no : 372028

Page No.
Date

Assignment no 8

Aim: Perform object detection from an image

Learning objective :→

1) Learn the concept of object detection, which involves identifying & localizing object within image or video frames.
2) familiarize with different object detection models, such as faster R-CNN, YOLO & SSD, and select one that suits your needs.
3) learn about class prediction for detected objects, often using softmax activation to estimate class probabilities.

Theory :→

Object detection is a computer vision task that involves identifying and locating objects within an image. Here's a high-level overview of the theory & steps involved in performing object detection from an images :

1) **Data collection:**
Gather a dataset of image that contain the object you want to detect. Annotate those images to specify the location of each object with bounding boxes.

2) **select a Model :→**
choose a pre-trained object detection model or design your own. popular models includes faster R-CNN, yolo and ssd (single shot Multi-Box Detector)

3) Preprocessing :→
prepare your input images by resizing, normalizing and augmenting them as needed.

4) feature Extraction :→
The selected model will use convolution layer to extract feature from the input image.

5) Object localization :→
The model's localization head will predict bounding boxes (coordinates) for each detected object within the image.

6) ~~Obje~~ object classification :→
The classification head of the model will predict the class of each detected object. This is often done using a softmax activation function to estimate class probabilities.

7) Non Maximum suppression :→
To eliminate duplicate or overlapping detection, apply non-maximum suppression (NMS) to retain the most confident predictions.

8) Post-processing :→
Translate the model's output into human-readable result by converting coordinates to pixel positions, associating class labels, and confidence score with each detected object.

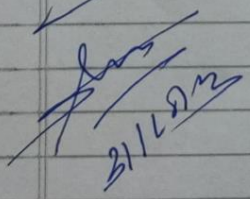
9) Visualization :→
Draw bounding boxes and labels on the original image to display the detected objects.

10) Evaluation and fine-tuning :→
Evaluating the model's performance using metrics like mAP (mean Average precision). fine tune the ~~message~~ model if necessary to improve detection accuracy.

11) Deployment :→
Deploy the object detection model in your application or system, whether it's for real-time processing or batch processing.

12) Inference :→
Use the trained model to perform object detection on new, ~~ns~~ unseen images.

~~Remember~~ :
Conclusion :→
Learn the concept of object detection, which involves identifying and localizing object within image or video frames.



Program Code:

```
import cv2
from matplotlib import pyplot as plt
import os

# Get the full path to the current script
script_path = os.path.dirname(os.path.abspath(__file__))

# Load Haar cascade file for eyes
cascade_path = os.path.join(
    script_path, "C:/Users/asus/Downloads/haarcascade_eye.xml")
stop_data = cv2.CascadeClassifier(cascade_path)

# Check if the cascade classifier is loaded successfully
if stop_data.empty():
    print(f"Error: Unable to load cascade classifier from {cascade_path}")
    exit()

# Opening image
img = cv2.imread("C:/Users/asus/Downloads/Rahul-Gandhi.webp")

# Convert the image to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect eyes
found = stop_data.detectMultiScale(img_gray, minSize=(20, 20))

# Display the original image
plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

# Display the image with detected eyes
for (x, y, width, height) in found:
    cv2.rectangle(img, (x, y), (x + width, y + height), (0, 255, 0), 5)

plt.subplot(1, 2, 2)
plt.title('Image with Detected Eyes')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

XML LINK: [haar-cascade-files/haarcascade_eye.xml at master · anaustinbeing/haar-cascade-files (github.com)]

Output:

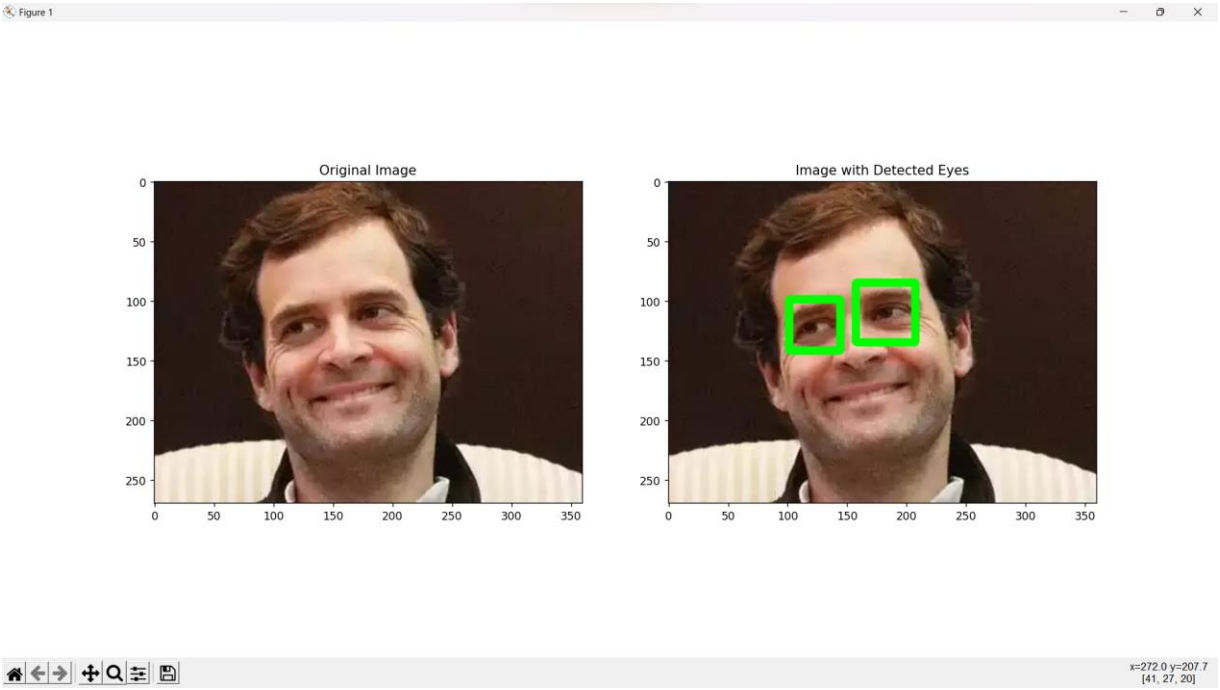


Figure 1
Original Image
Image with Detected Eyes
0
50
100
150
200
250
300
350
x=272.0 y=207.7
[41, 27, 20]