CS3008: IMAGE AND VIDEO PROCESSING

LABORATORY REPORTS

Name: Jayashre
Roll No.: 22011103020
College: Shiv Nadar University, Chennai

Chapter 1

Eye Detection Using OpenCV

1.1 Abstract

This report documents the implementation of an eye detection system using the OpenCV library. The system utilizes Haar cascade classifiers to identify faces and eyes in images and marks them with bounding boxes. The project aims to provide a foundational understanding of image processing techniques and their practical applications.

1.2 Introduction

Eye detection plays a vital role in computer vision applications such as gaze tracking, facial recognition, and user interaction systems. This project implements a detection system using pre-trained Haar cascade classifiers, which efficiently identify facial and ocular features in images.

1.3 Methodology

1.3.1 Data Collection

The input data consists of static images containing human faces. The images were uploaded manually in the Google Colab environment for processing.

1.3.2 Tools and Libraries

- **OpenCV**: For image processing and detection.
- **Google Colab**: For executing Python code and visualizing results.

1.3.3 Detection Algorithm

The steps followed in the implementation are:

1. Convert the input image to grayscale for easier processing.
2. Use the Haar cascade classifier to detect faces in the image.
3. For each detected face, apply the Haar cascade classifier for eyes within the facial region.
4. Highlight the detected features (faces and eyes) using bounding boxes.

1.4 Implementation

The implementation was carried out in Python using OpenCV. The following code snippet demonstrates the detection process:

Listing 1.1: Eye Detection Code

```python
import cv2
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
image = cv2.imread('input_image.jpg')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
    roi_gray = gray_image[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
cv2.imwrite('output_image.jpg', image)
cv2.imshow('Eye Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.5 Results and Discussion

The system was tested with several images under various conditions. The results are summarized below:

- Faces were detected accurately in well-lit images.

- Eye detection was successful but occasionally struggled with images where faces were partially obscured.

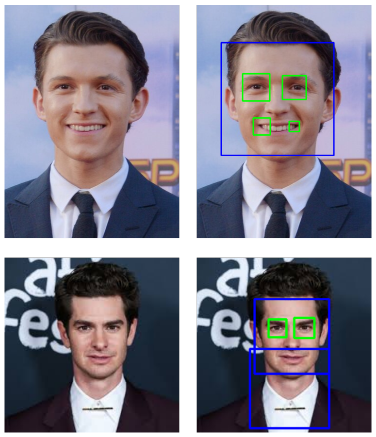

Figure 1.1: Detected faces and eyes in the input image.

1.6 Conclusion

The project successfully implemented an eye detection system using Haar cascades in OpenCV. While the system is efficient for basic detection, future improvements can involve the use of deep learning-based models for enhanced accuracy and robustness.