Addis Ababa University

Master's in Artificial Intelligence

Computer Vision (CV) Laboratory Manual

Prepared by: Dr. Natnael Argaw Wondimu

Environment: OpenCV and Visual Studio

Preface

This manual is prepared in a workbook style for AI Master's students at Addis Ababa University. It provides a hands-on introduction to Computer vision with OpenCV, covering advanced topics. Each lab includes objectives, theoretical background, procedures, OpenCV code, checkpoints, try-it-yourself prompts, and collaborative assignments. The manual is intended to serve as a base for advanced studies in Computer Vision.

Chapter 9: Object Detection with OpenCV (Python)

Objective

To learn and implement classical object detection techniques in OpenCV using Haar Cascades and HOG + SVM. This lab includes face and pedestrian detection from images and video streams.

1. What is Object Detection?

Description: Object detection is the task of locating and classifying objects within an image. It outputs bounding boxes around objects of interest.

2. Haar Cascade Classifiers

Description: Haar cascades use features trained with AdaBoost and cascade classifiers to detect objects quickly in real-time.

2.1 Face Detection with Haar Cascades

```python
import cv2
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

img = cv2.imread('face_sample.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

cv2.imshow('Detected Faces', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output Description: Draws rectangles around detected faces using pre-trained Haar cascades.

3. HOG + SVM Detector

Description: HOG (Histogram of Oriented Gradients) captures edge and gradient structure. Paired with SVM, it’s effective for detecting pedestrians.

3.1 Pedestrian Detection

```python
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

img = cv2.imread('pedestrians.jpg')
img = cv2.resize(img, (640, 480))

(rects, weights) = hog.detectMultiScale(img, winStride=(8,8), padding=(8,8), scale=1.05)

for (x, y, w, h) in rects:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

```
cv2.imshow('Pedestrian Detection', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output Description: Draws green rectangles around pedestrians in the image using HOG + SVM detector.

4. Real-Time Detection with Webcam

```
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    cv2.imshow('Real-Time Face Detection', frame)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Output Description: Performs live face detection from the webcam and displays it in real-time.

5. Summary

- **Haar Cascades** are fast but sensitive to lighting and scale.
- **HOG + SVM** is more robust and well-suited for pedestrian detection.
- Both methods work without deep learning and run in real time on CPUs.

Suggested Exercises

1. Detect eyes or smiles using haarcascade_eye.xml or haarcascade_smile.xml.
2. Replace the webcam input with a video file and detect people frame-by-frame.
3. Tune the HOG detector parameters (e.g., scale, winStride) for different performance.
4. Combine face and body detection into a single pipeline.