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 12: Facial Recognition & Landmark Detection (OpenCV DNN + Dlib)

Objective

To implement face detection, recognition using face embeddings, and facial landmark detection using OpenCV's DNN module and Dlib. This lab provides a foundation for building real-time face-based systems.

1. What is Facial Recognition?

Description: Facial recognition involves identifying or verifying a person's identity from an image. It often includes face detection, alignment, embedding extraction, and matching.

2. Environment Setup

```
pip install opencv-python dlib face_recognition imutils numpy
```

- **OpenCV DNN** for face detection
- **Dlib** for landmarks and embeddings
- **face_recognition** wrapper for simplified face encoding and comparison

3. Face Detection using OpenCV DNN

```python
import cv2
net = cv2.dnn.readNetFromCaffe('deploy.prototxt', 'res10_300x300_ssd_iter_140000.caffemodel')

img = cv2.imread('face.jpg')
h, w = img.shape[:2]
blob = cv2.dnn.blobFromImage(img, 1.0, (300, 300), (104.0, 177.0, 123.0))
net.setInput(blob)
detections = net.forward()

for i in range(detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > 0.5:
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (x1, y1, x2, y2) = box.astype("int")
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.imshow("Detected Face", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output Description: Displays green bounding boxes over detected faces using OpenCV DNN.

4. Facial Landmark Detection with Dlib

```python
import dlib
from imutils import face_utils

img = cv2.imread('face.jpg')
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
detector = dlib.get_frontal_face_detector()
rects = detector(gray, 1)

for rect in rects:
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
    for (x, y) in shape:
        cv2.circle(img, (x, y), 2, (0, 0, 255), -1)

cv2.imshow("Facial Landmarks", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output Description: Draws 68 facial landmarks including eyes, nose, jawline, and lips.

5. Face Recognition using Embeddings

```
import face_recognition

# Load known and test images
known_image = face_recognition.load_image_file("person1.jpg")
test_image = face_recognition.load_image_file("person2.jpg")

known_encoding = face_recognition.face_encodings(known_image)[0]
test_encoding = face_recognition.face_encodings(test_image)[0]

results = face_recognition.compare_faces([known_encoding], test_encoding)
print("Is the person a match?", results[0])
```

Output Description: Outputs True or False indicating whether the two faces belong to the same person.

6. Real-Time Face Recognition

```
import cv2
import face_recognition

known_image = face_recognition.load_image_file("person.jpg")
known_encoding = face_recognition.face_encodings(known_image)[0]

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    rgb_frame = frame[:, :, ::-1]

    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    for (top, right, bottom, left), encoding in zip(face_locations, face_encodings):
        match = face_recognition.compare_faces([known_encoding], encoding)[0]
        label = "Matched" if match else "Unknown"
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, label, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

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

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

Output Description: Performs real-time identity matching from webcam input using face embeddings.

7. Summary

- **OpenCV DNN** provides fast face detection.
- **Dlib** enables precise landmark localization.
- **Face embeddings** are robust descriptors for recognition.
- Real-time systems can be built using webcam + pre-trained encoders.

Suggested Exercises

1. Build a multi-face recognition system.
2. Record embedding vectors for 5 individuals and match against a test feed.
3. Display landmarks on live video.
4. Extend the pipeline to recognize expressions using landmarks.