Modern University
Faculty of
Computers
& Artificial Intelligence

MTI
Modern University
for technology & information

Pattern Recognition

Project: Face Detection

Suppervised By:

Pr. Dr. Mohamed ElGazzar

T.A. Maryam Essam

T.A. Nayera Mostafa

T.A. Mohamed Ameen

Group Members:

1. Malak Mohamed Muhannad Al-Sati (99573)
2. Maryam Homam Oyoun Alsoud (96816)
3. Dina Ahmed Ghoneim Mohamed Shalaby (98865)
4. Alaa Bahaa Eldeen Yousef Kobasiy (98042)
5. Mariam Mohamed AlSayed AlAraby (98582)

Introduction

Face detection is a fundamental task in computer vision with applications in security, biometrics, photography, and human-computer interaction. This project implements a face detection system using the **MTCNN** (Multi-Task Cascaded Convolutional Neural Network) algorithm, which is a deep learning-based approach for accurately detecting faces in images.

Key Concepts

- Detect multiple faces in an image with high precision.
- Filter detections based on confidence scores and minimum face size to reduce false positives.
- Visualize detected faces with bounding boxes for easy interpretation.
- Compare results across multiple images in a side-by-side view.

Features

- Face detection using MTCNN algorithm
- Confidence-based filtering of detected faces
- Minimum face size filtering
- Visualization of detected faces with bounding boxes
- Side-by-side comparison of results from multiple images

Dependencies

1. OpenCV

Purpose:
OpenCV (Open Source Computer Vision Library) is a powerful open-source library for real-time computer vision and image processing.

Role in This Project:

- Reads input images (cv2.imread)
- Converts color spaces (cv2.cvtColor)
- Draws bounding boxes around detected faces (cv2.rectangle)

Why OpenCV?

- Optimized for fast image processing.
- Provides essential tools for image manipulation.
- Widely used in computer vision applications.

2. NumPy (numpy)

Purpose:
NumPy is a fundamental package for scientific computing in Python, providing support for large multi-dimensional arrays and matrices.

Role in This Project:

- Handles image data as numerical arrays.
- Supports matrix operations for efficient computations.
- Works seamlessly with OpenCV (since OpenCV images are NumPy arrays).

Why NumPy?

- Enables fast numerical operations on image pixels.
- Essential for interfacing with deep learning models.

3. Matplotlib (matplotlib)

Purpose:
Matplotlib is a plotting library for creating static, interactive, and animated visualizations in Python.

Role in This Project:

- Displays detected faces in a side-by-side comparison (plt.subplots).
- Adds titles and annotations (axes.set_title).
- Renders images with bounding boxes (axes.imshow).

Why Matplotlib?

- Provides high-quality image visualization.
- Useful for comparing multiple results in a single figure.

4. MTCNN (mtcnn)

Purpose:
MTCNN (Multi-Task Cascaded Convolutional Neural Network) is a deep learning-based face detection algorithm that detects faces and facial landmarks.

Role in This Project:

- Detects faces with bounding box coordinates (detector.detect_faces).
- Provides confidence scores for each detection (face['confidence']).
- Handles variations in pose, lighting, and occlusion better than traditional methods.

Why MTCNN?

- More accurate than Haar cascades or HOG-based detectors.
- Detects faces at different scales and angles.
- Filters weak detections using confidence thresholds.

Code

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from mtcnn.mtcnn import MTCNN

# Function to detect and draw faces with confidence filtering
def draw_faces_with_mtcnn(image, confidence_threshold=0.90, min_face_size=30):
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    detector = MTCNN()
    faces = detector.detect_faces(rgb)

    filtered_faces = []
    for face in faces:
        x, y, w, h = face['box']
        confidence = face['confidence']
        if confidence >= confidence_threshold and w >= min_face_size and h >=
min_face_size:
            filtered_faces.append(face)
            cv2.rectangle(rgb, (x, y), (x + w, y + h), (0, 0, 255), 2)

    return rgb, len(filtered_faces)

# Load image paths
image1_path = "C:/Users/maryam/Desktop/LVL 4, Semester 2/Pattern
Recognition/Project/football_team.jpg"
image2_path = "C:/Users/maryam/Desktop/LVL 4, Semester 2/Pattern
Recognition/Project/national_team.jpg"

# Read images
img1 = cv2.imread(image1_path)
img2 = cv2.imread(image2_path)

# Check loading status
if img1 is None or img2 is None:
    print("One or both images could not be loaded.")
else:
    print("Images loaded successfully.")

    # Detect and annotate faces
    img1_with_faces, count1 = draw_faces_with_mtcnn(img1)
    img2_with_faces, count2 = draw_faces_with_mtcnn(img2)

    # Plot side-by-side
    fig, axes = plt.subplots(1, 2, figsize=(14, 7))

    axes[0].imshow(img1_with_faces)
    axes[0].set_title(f"Faces in image1: {count1}")
    axes[0].axis('off')

    axes[1].imshow(img2_with_faces)
    axes[1].set_title(f"Faces in image2: {count2}")
    axes[1].axis('off')

    plt.tight_layout()
    plt.show()
```

Output

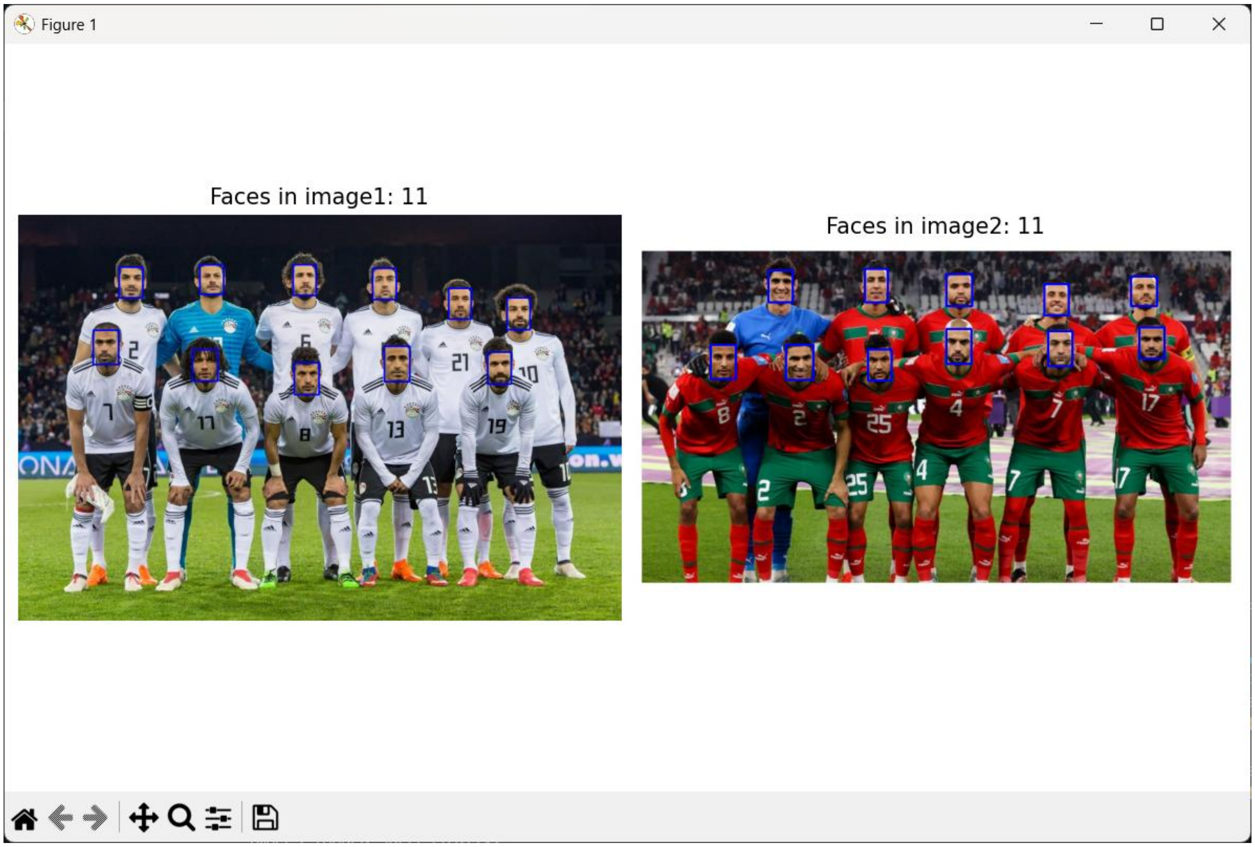

Figure 1
Faces in image1: 11
Faces in image2: 11