Pattern Mini-Project Zetona

This Python code uses **OpenCV**, **NumPy**, **Matplotlib**, and **MTCNN** (Multi-task Cascaded Convolutional Networks) to detect faces in two images, filter them by confidence and size, and visualize the results.

Let's break it down step by step:

🔹 Imports

```
import cv2  # For image loading and basic operations
import numpy as np  # Not used in this code, but often needed for array operations
import matplotlib.pyplot as plt  # For displaying images
from mtcnn.mtcnn import MTCNN  # For face detection using a pre-trained MTCNN model
```

🔹 Function: draw_faces_with_mtcnn()

Purpose:

Detects faces in an image using MTCNN, filters them based on:

- **Confidence level** (how sure the model is that it found a face)
- **Minimum face size**

Parameters:

- image: Input image (expected in BGR format as from OpenCV)
- confidence_threshold: Minimum confidence required to consider a detection valid (default is 0.90)
- min_face_size: Minimum width/height for a detected face to be accepted (default is 30 pixels)

Code:

```
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert from BGR (OpenCV) to RGB (MTCNN expects RGB)
detector = MTCNN()  # Instantiate face detector
```

```
faces = detector.detect_faces(rgb)  # Detect faces in image
```

Loop through detected faces:

```
for face in faces:
    x, y, w, h = face['box']  # Bounding box of the face
    confidence = face['confidence']  # Detection 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)  # Draw a red rectangle around the face
```

Return:

- The RGB image with rectangles drawn
- The number of valid faces detected

🔹 **Image Input**

```
image1_path = "C:/Users/maryam/Desktop/LVL 4, Semester 2/Pattern Recognition/Project/team.jpg"
image2_path = "C:/Users/maryam/Desktop/LVL 4, Semester 2/Pattern Recognition/Project/team2.jpg"
```

🔹 **Read Images**

```
img1 = cv2.imread(image1_path)
img2 = cv2.imread(image2_path)
```

🔹 **Check for successful loading**

```
if img1 is None or img2 is None:
    print("One or both images could not be loaded.")
```

🔹 **Detect and Annotate Faces**

```
img1_with_faces, count1 = draw_faces_with_mtcnn(img1)
img2_with_faces, count2 = draw_faces_with_mtcnn(img2)
```

🔹 Visualization using Matplotlib

```
fig, axes = plt.subplots(1, 2, figsize=(14, 7))  # Create two side-by-side plots

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

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

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

🔹 Summary

This script:

1. Loads two images.
2. Uses the MTCNN face detector.
3. Filters faces based on confidence and size.
4. Draws bounding boxes around valid faces.
5. Displays the results side-by-side with the count of detected faces.