Experiment-8 (B).py

```python
# 18BCS6201-CV Practical-8 (B) (Gauri Prabhakar) (AI-ML-2)(B)
# Aim: To implement face detection and mesh using mediapipe in python and OpenCV.

# Importing necessary modules.
import cv2
import mediapipe as mp

# We will use 'drawing_utils' to draw the key points.
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
# We will use 'face_mesh' to draw the mesh points.
mp_face_mesh = mp.solutions.face_mesh

IMAGE_FILES = [r"C:\Users\gauri\Desktop\OpenCV Media\jennie.jpg"]
#drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)

# Specifying the detection and tracking confidence uisng 'mp_face_mesh.FaceMesh()'.
with mp_face_mesh.FaceMesh(
    static_image_mode=True,
    max_num_faces=1,
    #refine_landmarks=True,
    min_detection_confidence=0.5) as face_mesh:

    # Iterating through IMAGE_FILES list.
  for idx, file in enumerate(IMAGE_FILES):
      # Reading the list.
    image = cv2.imread(file)
    # Reading the image and converting it to RGB.
    results = face_mesh.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # Print and draw face mesh landmarks on the image.
    #if not results.multi_face_landmarks:
      #continue
    #annotated_image = image.copy()
    #for face_landmarks in results.multi_face_landmarks:
      #print('face_landmarks:', face_landmarks)
      #mp_drawing.draw_landmarks(
          #image=annotated_image,
          #landmark_list=face_landmarks,
          #connections=mp_face_mesh.FACEMESH_TESSELATION,
          #landmark_drawing_spec=None,
          #connection_drawing_spec=mp_drawing_styles
          #.get_default_face_mesh_tesselation_style())
      #mp_drawing.draw_landmarks(
          #image=annotated_image,
          #landmark_list=face_landmarks,
          #connections=mp_face_mesh.FACEMESH_CONTOURS,
          #landmark_drawing_spec=None,
          #connection_drawing_spec=mp_drawing_styles
          #.get_default_face_mesh_contours_style())
    #cv2.imwrite('/tmp/annotated_image' + str(idx) + '.png', annotated_image)

    # To improve performance, marking the image as not writable and passing by reference.
    image.flags.writeable = False
    # Reading the frames and converting them to RGB.
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Detecting facess in the frame using the function 'face_mesh.process()'.
    results = face_mesh.process(image)

    # Drawing the face mesh annotations on the image.
    image.flags.writeable = True
    # Reading the frames and converting them to RGB.
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    # If faces are detected that is 'results.multi_face_landmarks' returns true:
    if results.multi_face_landmarks:
      # For 'face_landmarks' variable in 'results.multi_face_landmarks':
      for face_landmarks in results.multi_face_landmarks:
        # Connecting the key points using the function 'mp_drawing.draw_landmarks()'.
        # Face Mesh Tesselation.
        mp_drawing.draw_landmarks(
            image=image,
            landmark_list=face_landmarks,
            connections=mp_face_mesh.FACEMESH_TESSELATION,
            landmark_drawing_spec=None,
            connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_tesselation_style())

        # Connecting the key points using the function 'mp_drawing.draw_landmarks()'.
        # Face Mesh Contours.
        mp_drawing.draw_landmarks(
            image=image,
            landmark_list=face_landmarks,
            connections=mp_face_mesh.FACEMESH_CONTOURS,
            landmark_drawing_spec=None,
            connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_contours_style())

    # Rendering the image with effective face tracking to the console by using the function '.imshow()'.
    cv2.imshow('Face Mesh using Mediapipe', image)

    # Setting up '.waitkey()' to wait for a specific time until any key is pressed and break the loop.
    # '.waitkey(9000)' displays a frame for 9000ms after which it moves to the next frame in the video.
    # Setting 'x' as the quitting button.
    if cv2.waitKey(9000) & 0xFF == ord('x'):
            break
    else:
        break
```