Experiment-8 (A).py

```python
# 18BCS6201-CV Practical-8 (A) (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

#drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
# Creating a variable to store the video using the '.VideoCapture()' function.
cap = cv2.VideoCapture(r"C:\Users\gauri\Desktop\OpenCV Media\To Plan or Not to Plan_.mp4")

# Specifying the detection and tracking confidence uisng 'mp_face_mesh.FaceMesh()'.
with mp_face_mesh.FaceMesh(
        max_num_faces=1,
        min_detection_confidence=0.5,
        min_tracking_confidence=0.5) as face_mesh:
    # While the video is running.
    while cap.isOpened():
        # Capturing the video frame by frame using the '.read()' method.
        success, image = cap.read()

        # If there are empty frames.
        if not success:
            print("Ignoring empty camera frame.")
            # If loading a video, use 'break' instead of 'continue'.
            continue

        # To improve performance, marking the video 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 video 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(1)' displays a frame for 1ms after which it moves to the next frame in the video.
    # Setting 'x' as the quitting button.
        if cv2.waitKey(5) & 0xFF == ord('x'):
            break

# Releasing the variable/object 'cap'.
cap.release()
```