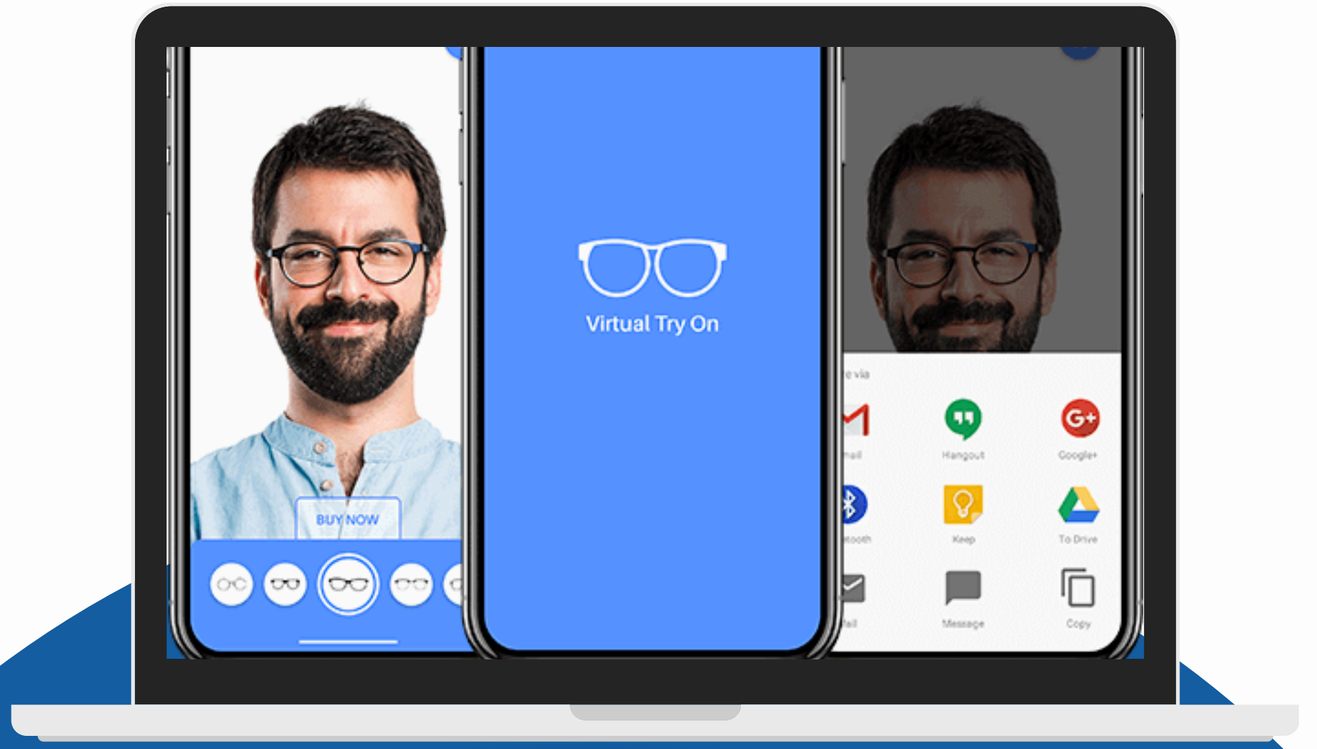
Glasses Virtual Try On

PRESENTED BY
Nida Pervaiz
Urwah Rasheed
Wajeeha Jahangir



Introduction

Virtual try-on uses computer vision to overlay virtual objects, like glasses, onto a live camera feed.

Objective of the Project:

To create a real-time application that detects facial landmarks and dynamically adjusts virtual glasses to fit seamlessly.

Tools and Libraries Used

MediaPipe For detecting and tracking facial landmarks.

OpenCV

(OpenCV): For image processing and webcam integration

NumPy For mathematical operations on image data.

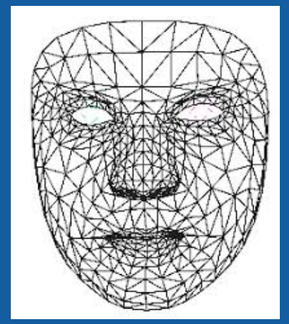
Technical Workflow

Webcam Feed Acquisition

OpenCV

MediaPipe

Facial Landmark Detection



NumPy

Dynamic Glasses Overlay

Alpha blending

Display and Interaction

Step 1:

Convert the image to HSV color space.

```
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

Applied a color threshold to detect the background.

Generated and alpha mask for transparency

Merged alpha channel with glasses image

Facial Landmark Detection

Tracks 468 facial landmarks in real-time.
Key landmarks used:

- Left Eye: Landmark 33.
- Right Eye: Landmark 263.
- Nose Bridge: Landmark 6.

Output: Normalized coordinates (x, y) relative to the frame size

```
# MediaPipe FaceMesh setup
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False, max_num_faces=1, refine_landmarks=True)
```

```
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = face_mesh.process(rgb_frame)
```

```
if results.multi_face_landmarks:
    for face_landmarks in results.multi_face_landmarks:
        # Get key landmarks: left eye, right eye, and nose tip
        left_eye = face_landmarks.landmark[33]  # Left eye corner
        right_eye = face_landmarks.landmark[263]  # Right eye corner
        nose_bridge = face_landmarks.landmark[6]  # Nose bridge landmark

        # Convert normalized coordinates to pixel coordinates
        left_eye_coords = (int(left_eye.x * w), int(left_eye.y * h))
        right_eye_coords = (int(right_eye.x * w), int(right_eye.y * h))
        nose_bridge_coords = (int(nose_bridge.x * w), int(nose_bridge.y * h))
```

Positioning and Rotating Glasses

- Calculated Position:
  - Midpoint between the left and right eyes as the glasses' center.
  - Adjusted vertical offset to align with the nose bridge.
- Dynamic Scaling:
  - Measured distance between eyes to set glasses size.
- Rotation:
  - Computed tilt angle using the slope between eye landmarks.

Calculating Position

```
# Calculate midpoint between eyes
eye_center_x = (left_eye_coords[0] + right_eye_coords[0]) // 2
eye_center_y = (left_eye_coords[1] + right_eye_coords[1]) // 2

# Adjust vertical position to sit slightly above the nose bridge
vertical_offset = nose_bridge_coords[1] - eye_center_y
adjusted_center = (eye_center_x, eye_center_y + vertical_offset // 2)
```

Dynamic Scaling

```
#rotation angle (eye tilt)
delta_y = right_eye_coords[1] - left_eye_coords[1]
delta_x = right_eye_coords[0] - left_eye_coords[0]
angle = np.degrees(np.arctan2(delta_y, delta_x))

#dynamic glasses size
eye_distance = int(np.hypot(delta_x, delta_y))
glasses_width = int(eye_distance * 1.8)
glasses_height = int(glasses_width * glasses_image.shape[0] / glasses_image.shape[1])
```

Rotation

```
#rotation angle (eye tilt)
delta_y = right_eye_coords[1] - left_eye_coords[1]
delta_x = right_eye_coords[0] - left_eye_coords[0]
angle = np.degrees(np.arctan2(delta_y, delta_x))
```

Alpha Blending for Overlay

Purpose:

To ensure smooth blending of glasses onto the face.

Steps:

- Blend using the alpha channel (transparency) to integrate the glasses with the frame seamlessly.

```
# Rotate the image
center = (w // 2, h // 2)
rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
rotated_overlay = cv2.warpAffine(overlay_resized, rotation_matrix, (w, h), flags=cv2.INTER_LINEAR, borderMode=cv2.BORDER_CONSTANT, borderValue=(0, 0, 0, 0))
```

```
# Overlay image with alpha blending
x, y = position
for i in range(h):
    for j in range(w):
        if x + j >= background.shape[1] or y + i >= background.shape[0]:
            continue
        alpha = rotated_overlay[i, j, 3] / 255.0  # Alpha channel
        if alpha > 0:  # Blend only non-transparent pixels
            background[y + i, x + j] = (
                1 - alpha
            ) * background[y + i, x + j] + alpha * rotated_overlay[i, j, :3]
return background
```

USER INTERACTION

Use keys to switch between glasses styles:

'n': **Next style.**
'p': **Previous style.**
'q': **Quit the application.**

Display selected glasses in a side preview window

DEMO AND LIVE EXAMPLE


Selected Glasses
Virtual Glasses Overlay
InShOt

THANK YOU!