Real-Time Virtual Try-On using Face Landmark Analysis

Farit Sharafutdinov
B23 - AI01
Innopolis University
Innopolis, Russia
f.sharafutdinov@innopolis.university

Grigorii Belyaev
B23 - AI01
Innopolis University
Innopolis, Russia
g.belyaev@innopolis.university

***Abstract*—This project presents a real-time Augmented Reality (AR) application that allows users to virtually try on 2D accessories such as glasses or hats. The system works by capturing video from a webcam, detecting the user's face, and estimating its key landmarks. Based on these landmarks, the application positions and transforms the chosen accessory so that it naturally follows the user's head movements. To make the overlay look smooth and realistic, we combine geometric warping techniques with alpha-blending. The main contribution of our work is an optimized pipeline for transformation and blending that enhances the stability of real-time AR experiences.**

***Index Terms*—Computer Vision, Real-Time Systems, Face Tracking, Face Landmarks, Augmented Reality, OpenCV, Affine Transformation**

I. Introduction

The aim of this project is to build an interactive AR application that can overlay 2D accessories directly onto a person's face captured by a webcam. The challenge lies in ensuring that the accessory moves in sync with the user's head, taking into account changes in pose, scale, and rotation. Achieving this requires both fast face detection and precise geometric transformations. The final system demonstrates how computer vision can be used to create engaging and practical AR tools.

II. Methodology

The system follows a four-step pipeline, applied to each video frame:

1. **Face Detection:** A pre-trained model (such as MediaPipe or Dlib) is used to detect the face in the video stream.
2. **Landmark Estimation:** A landmark model (e.g., Dlib 68-point or MediaPipe Face Mesh with 468 points) extracts key facial features like eyes, nose, and mouth corners.
3. **Geometric Transformation:** Using reference points from the accessory and the corresponding face landmarks, we compute a transformation matrix that accounts for perspective and head orientation.
4. **Overlay and Blending:** The accessory image (PNG with alpha channel) is warped according to the matrix and blended into the video frame using alpha-blending (`cv2.addWeighted`).

Contribution

Our main contribution is the custom implementation of the geometric transformation logic and optimized blending routines. These make the AR overlay more stable and visually convincing compared to using standard landmark libraries alone.

III. Dataset and Resources

We build on existing pre-trained models and resources rather than training models from scratch:

- **Model Weights:** The project uses models trained on datasets like **iBUG 300-W** (via Dlib) or those integrated into **MediaPipe**.
- **Accessory Images:** A small set of custom PNG images (with transparent backgrounds) was created to serve as virtual accessories.
- **Reference Example:** Dlib's C++ landmark detection models trained on iBUG 300-W.

IV. Project Plan and Timeline

The project spans seven weeks and is divided between two members:

- **Weeks 1-2:** Set up the environment (Python, OpenCV, Dlib/MediaPipe). Implement video capture and integrate facial landmark tracking.
- **Weeks 3-4:** Develop geometry logic, including source-to-destination point mapping and perspective transformation (`cv2.getPerspectiveTransform`, `cv2.warpPerspective`).
- **Weeks 5-7:** Create an interface for switching accessories, integrate all modules, and perform evaluation (FPS, accuracy, and final reporting).

References

[1] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
[2] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10, 1755-1758.
[3] Zhang, Y., et al. (2020). MediaPipe Hands: On-Device Real-time Hand Tracking. *Google AI Blog*.
[4] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
[5] Porter, T., & Duff, T. (1984). Compositing digital images. *SIGGRAPH Comput. Graph.* 18, 3, 253–259.