# RealSense Face Filters with Gesture Control

Authors: Benedek Vass, Gergő Honyák, Vince Németh

Date: 25.04.2025

Technology Stack: Python, Intel RealSense SDK, MediaPipe, OpenCV, NumPy

## Introduction

This project is an interactive real-time face filter application that leverages the Intel RealSense depth camera to detect faces, apply different visual effects (filters), and switch between them using intuitive hand and facial gestures. The application is written in Python and makes use of advanced computer vision libraries such as OpenCV and MediaPipe, combined with the Intel RealSense SDK for depth sensing.

The project is available on Github:

<https://github.com/BenedekVass240814/DepthVision>

## Objective

The main goal of the project is to demonstrate a touchless, gesture-controlled filter system that overlays various effects on the human face using depth and color data from a RealSense camera. By recognizing facial landmarks and hand gestures, the system enables natural and interactive control over visual modifications in the video feed.

## System Requirements

To run the project, the following hardware and software components are required: An Intel RealSense depth camera (e.g., D435 or D415), Python 3.7 or later, OpenCV for real-time image processing, MediaPipe for facial and hand landmark detection, pyrealsense2 for accessing RealSense camera data, and NumPy for numerical operations. The required Python packages can be installed using pip:

pip install opencv-python mediapipe pyrealsense2 numpy

## Project Structure

The project is organized into the following directories and files: The 'assets' folder contains image files used by the filters, such as sunglasses.png and mustache.png. The 'src' directory contains the core modules of the project: 'face\_filters.py' has functions to apply various filters like blur, mustache, sunglasses, and depth vision. 'facial\_landmark\_recognition.py' detects facial landmarks using MediaPipe's Face Mesh. 'gesture\_recognition.py' analyzes both hand and facial landmark data to detect gestures. 'hand\_landmark\_detection.py' detects hand landmarks from the image frame. 'realsense\_capture.py' includes the main application logic that initializes the RealSense camera, processes frames, detects gestures, and applies the appropriate filters. 'webcam\_constant.py' stores constants for keys, fonts, filter identifiers, and the on-screen menu. The 'main.py' file acts as the main entry point to launch the RealSense-based filter application.

## Functional Overview

When the application is launched, the RealSense camera is initialized to stream both color and depth data. MediaPipe is used to extract facial and hand landmarks from the color frames. These landmarks are used to determine which gestures the user is performing. Each gesture corresponds to a specific filter: An open palm resets the filters and shows the raw color image. One finger up activates a facial landmark overlay. Raising the hand above the face applies a blur filter to the face area. Opening the mouth triggers the sunglasses filter. Placing a finger under the nose while raising a finger overlays a mustache image. Closing the eyes enables a depth-based vision effect using the depth stream. Filters are applied in real time, and the video frame is resized and padded to fit the window dynamically. An on-screen text menu displays instructions to the user.

## Depth Processing

Depth frames are processed using RealSense’s spatial, temporal, and hole-filling filters to reduce noise and fill missing values. The depth-based filter provides a colormap-style visualization based on distance from the camera, adding a unique aesthetic and functional mode.

## Gesture Recognition Logic

Gesture recognition is based on relative positions of landmark points. For instance, an open palm is identified when all fingers and the thumb are extended, while the mouth open gesture checks the vertical distance between the upper and lower lips. Hand landmarks are used to recognize pointing, palm orientation, and gestures in relation to facial landmark positions such as the nose or eyes.

## Filters Implementation

Each filter is implemented as a modular function: The blur filter applies a Gaussian blur to the face region. The sunglasses and mustache filters use asset images and landmark positions to overlay the graphics accurately on the user's face. The landmark filter simply draws green dots on all facial landmarks. The depth filter replaces the color frame with a processed depth visualization using a color map. These filters are implemented in a way that allows easy addition of new effects in the future.

## Future Enhancements

This project can be expanded further by incorporating the following ideas: Support for multiple users and multiple faces, voice commands in addition to gestures, more advanced filters using augmented reality techniques, a GUI overlay for manual filter selection, and exporting the video feed to a file or stream.

## Demo

<https://github.com/BenedekVass240814/DepthVision/blob/81c6a775fbd72355740f86cb91a71fe08a992009/demo.mp4>

## Conclusion

This project demonstrates a creative and interactive use of computer vision and depth sensing for real-time face filtering. By integrating gesture recognition with facial analysis, it creates an engaging user experience that is both hands-free and visually dynamic. The modular codebase also makes it easy to extend or adapt for other interactive computer vision applications.