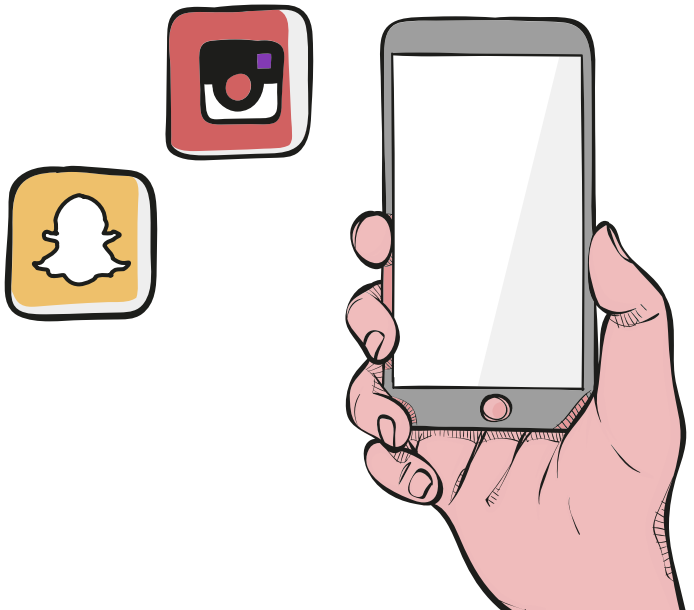
Create AR Face Filters with MediaPipe

ECE 495 Digital Image Processing
Spring 2022 Term Project

Yilei Li, Rishi Mitra

Introduction: Background

Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide thousands of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.



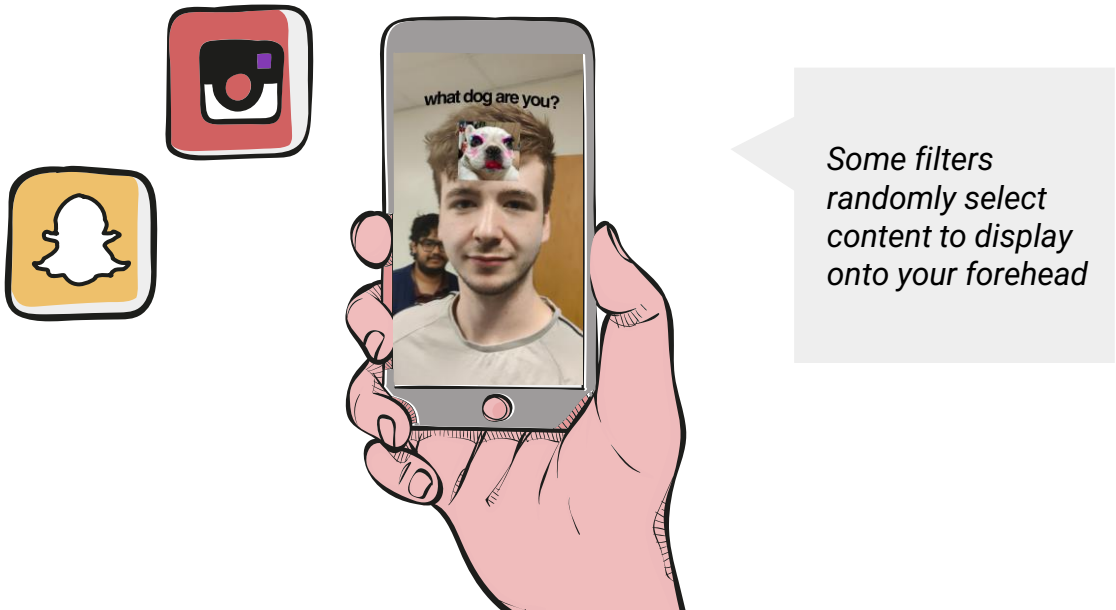
Introduction: Background

Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide hundreds of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.

Some filters consists of static images overlaid on to specific locations of your face

Introduction: Background

Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide hundreds of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.



Some filters randomly select content to display onto your forehead

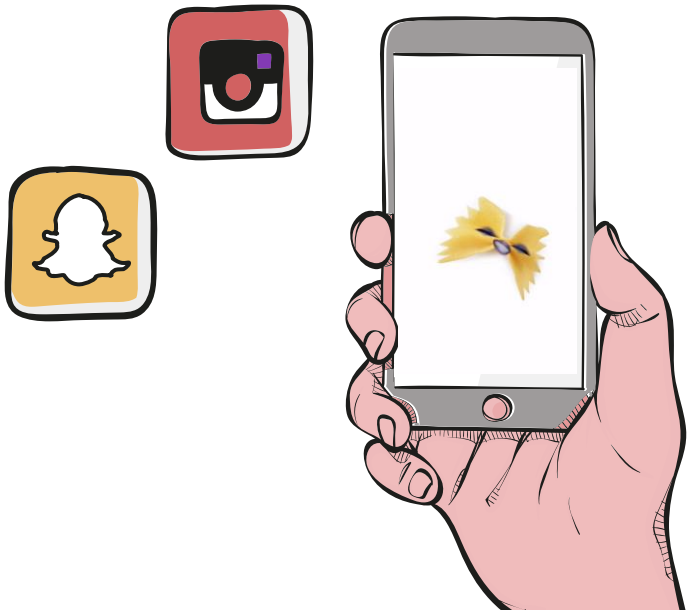
Introduction: Background

Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide hundreds of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.

Some filters consists of static graphics/texts overlaid on to specific locations of your face

Introduction: Background

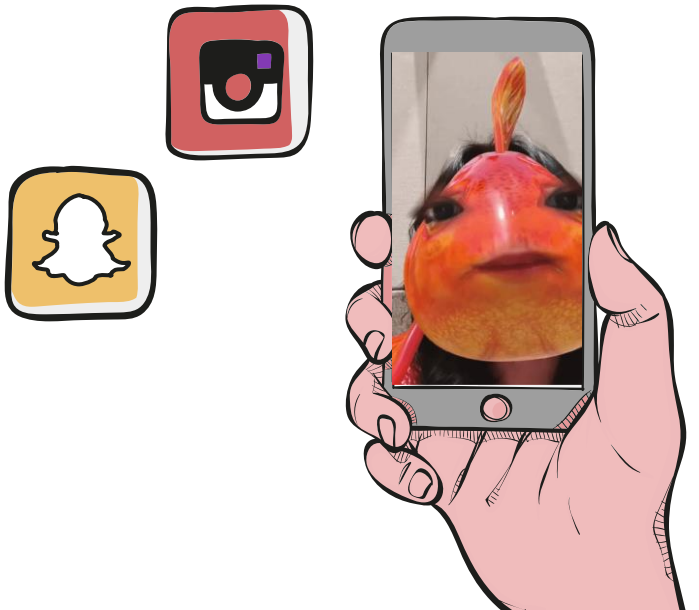
Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide hundreds of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.



Some filters are morphed onto your entire face and adjust to your movements and expressions

Introduction: Background

Augmented Reality (AR) is an interactive experience of a real-world environment where computer-generated digital objects can reside. Powered by AR, popular social media platforms such as Snapchat and Instagram provide hundreds of filters that can make you look as wild, exotic, or beautiful as you wish, that too in a matter of seconds.



Some filters are animated and moves on their own

Objectives

Through this project, we learned how these Augmented Reality Filters work and how we can design and implement our own filters using modern image processing frameworks such as OpenCV and MediaPipe in a Python environment.

Our objective is to develop a program that implements morphed, static, and animated camera filters on a detected human face.

Morphed filter sample source image

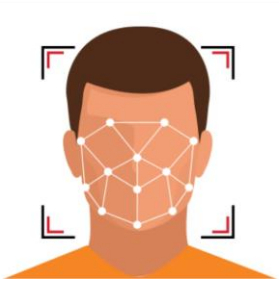
Static filter sample source image

Animated filter sample source image

Methodology

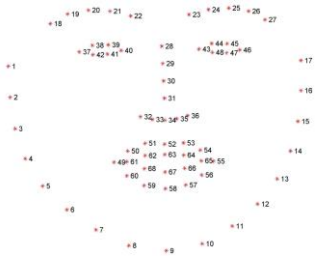
Creating an AR filter

01 Face Detection



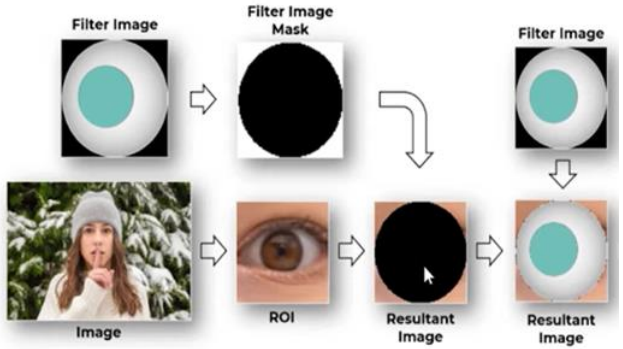
First and foremost, detection of face(s) must be made in the camera frame at real-time.

02 Feature/key points detection



Then, a landmark predictor is used to identify the key points of the facial features (face landmarks), which aligns the predefined set of feature points to a human face.

03 Applying filter

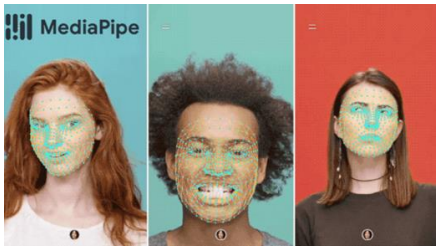


Finally, the filter image with detected key points and annotations can be morphed, or a filter can be applied on top of it to change the appearance of the face in the camera frame.

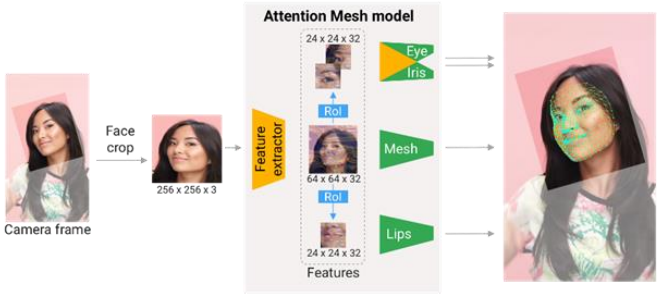
Development Details I

01 Face Detection

1.1 Detect 468 Facial Landmarks using Mediapipe Face Mesh

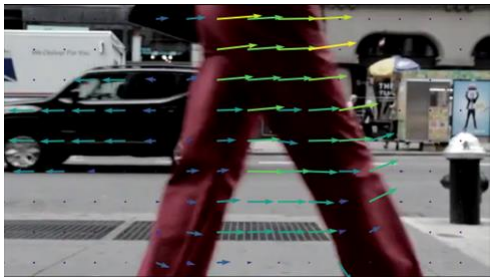


MediaPipe Face Mesh is a solution that estimates 468 3D face landmarks in real-time. It employs machine learning (ML) to infer the 3D facial surface, requiring only a single camera input without the need for a dedicated depth sensor.



We applied attention to semantically meaningful face regions, and therefore predicting landmarks more accurately around lips, eyes and irises, at the expense of more compute.

1.2 Stabilize the landmark points detection



Finally, we used Optical Flow implemented with OpenCV to estimate the detected face's displacement vector caused by its motion or camera movements. Using that we can predict the location of landmarks in the next frame and stabilize them.

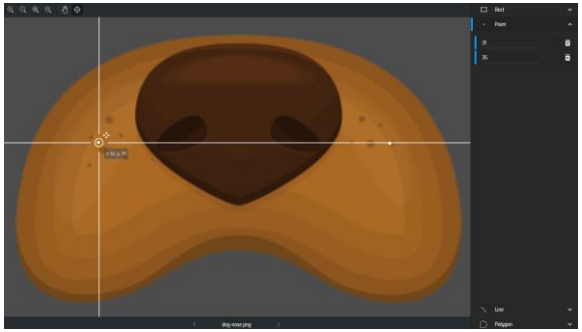
Development Details II

02 Feature/key points detection

2.1 Select the relevant landmarks from the 468 landmarks

Key features of the face, i.e., Eyes, Nose, Lips, Eyebrows, and jawline were selected (75 landmarks), similar to the Dlib 68 points face landmark detector with a few additional points for the forehead.

2.2 Annotate the filters with regards to the selected landmarks



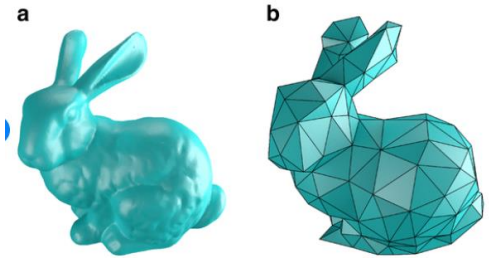
Makesense, an annotation web tool was used to create annotation points associated with the selected face landmarks to obtain the coordinates of each points

For a morphed filter, it is necessary to annotate all 75 selected face landmarks. For an overlayed or animated filter, we only need to annotate enough key points to signify the location of which the filter image(s) will be overlayed onto the face.

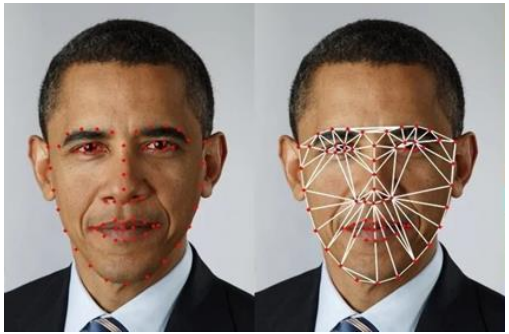
Development Details III

03 Applying filter

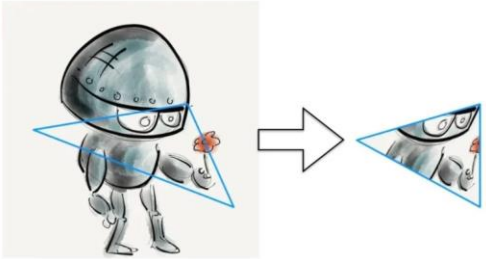
3.1 Transform the Filter on the face using the landmarks



In computer graphics, warping triangles is a common technique because any 3D surface can be approximated by triangles. Given a set of points in a plane, Triangulation refers to the subdivision of the plane into triangles, with the points as vertices. In a Delaunay triangulation, triangles are chosen such that no point is inside the circumcircle of any triangle.



Functions were developed to find convex hull for delaunay triangulation using the landmark points and function to load a filter and apply denaulay to obtain triangulations of both the filter image and the detected face.



Once we have triangulations of both the images we can transform the triangles from one image(filter) to fit onto another image(person's face). We achieved this by using Affine transform that requires 3 points (aka a triangle) on the images to warp them onto each other. Gaussian filters was then used to blur the transformed filter for natural appearance.

Additional Notes on Animated Filters

The image source of the animated filters come from gifs with transparent background. We implemented those filter by splitting up the gif into frames and annotated them in the same fashion as Static filters.

The mask applied to the face is changed sequentially, a frame at a time, at a frame rate of 0.01-0.06 fps. The smoothest results were achieved at a frame rate 0.01 fps.

Transparent GIF

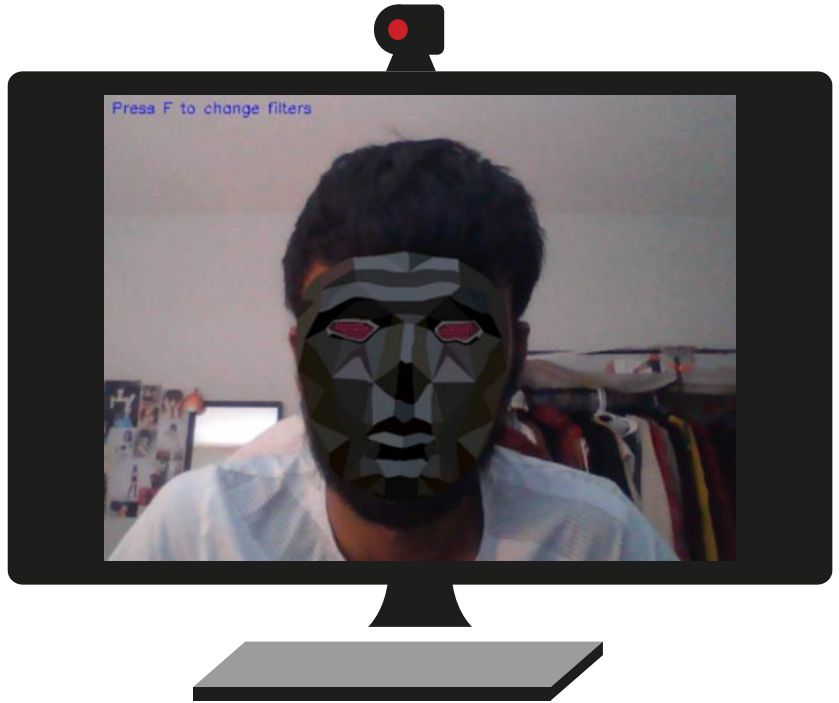
Splitted into 8 transparent PNG

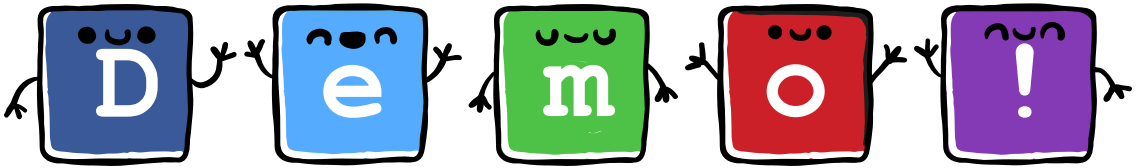
Result: Sample Morphed Filter

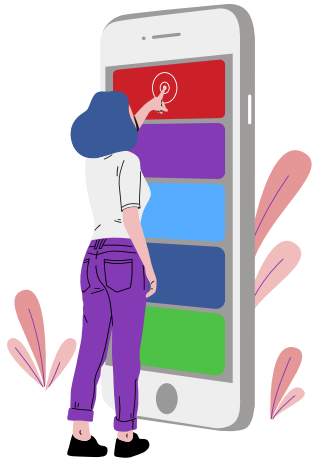
PNG image

54	515	Squid-Game-Front-Man-Mask.png	542	739
55	526	Squid-Game-Front-Man-Mask.png	542	739
56	557	Squid-Game-Front-Man-Mask.png	542	739
57	573	Squid-Game-Front-Man-Mask.png	542	739
58	582	Squid-Game-Front-Man-Mask.png	542	739
59	574	Squid-Game-Front-Man-Mask.png	542	739
60	554	Squid-Game-Front-Man-Mask.png	542	739
61	534	Squid-Game-Front-Man-Mask.png	542	739
62	530	Squid-Game-Front-Man-Mask.png	542	739
63	530	Squid-Game-Front-Man-Mask.png	542	739
64	529	Squid-Game-Front-Man-Mask.png	542	739
65	531	Squid-Game-Front-Man-Mask.png	542	739
66	541	Squid-Game-Front-Man-Mask.png	542	739
67	542	Squid-Game-Front-Man-Mask.png	542	739
68	542	Squid-Game-Front-Man-Mask.png	542	739
69	174	Squid-Game-Front-Man-Mask.png	542	739
70	77	Squid-Game-Front-Man-Mask.png	542	739
71	17	Squid-Game-Front-Man-Mask.png	542	739
72	4	Squid-Game-Front-Man-Mask.png	542	739
73	10	Squid-Game-Front-Man-Mask.png	542	739
74	68	Squid-Game-Front-Man-Mask.png	542	739
75	167	Squid-Game-Front-Man-Mask.png	542	739
76				
77				

75 points Annotation




D
e
m
o
!



Conclusion

Performance

- *Most filter implemented functions as expected*
- *Some distortion issues were notable for morphed filters. Improved annotation and better mapping strategy between key points and facial landmarks could be developed to resolve the issue.*

Limitations

This program is limited to working with filters in the 2 dimensions.

Although it gives the illusion of being 3D with the morphing techniques, our approach falls short when 3D object filters come into play such as sun glasses.

Thank you for your time!

(questions?)