VR final Report – 京劇變臉(Mask Switching)

Team Members: B04902003 董書博 B04902021 陳弘梵 B04902092 張均銘

1. Problem definition

Due to the prevailing of 3D techniques, virtual reality is getting more and more connected to it. We thought of strengthening face effects, such as putting on makeup, enlarging eyes or nose, etc. However, we wanted to integrate all these components to a complete special effect on the whole face. Finally, we came out of "Mask switching" as an idea, which is also fun and useful.

(1) Face detection

- ✓ We use the camera to capture appearance of people. While any face appears, it will be marked and then we put mask on it.
- ✓ This is a real-time program and is considered capable with multi-users.
- ✓ An excellent model is trained and used to detect any moment shot by the camera.

(2) Face morphing

- ✓ The technique allows combining two images into one, through Delaunay triangulation.
- ✓ The weighting proportion of two images can be adjusted to highlight the mask.
- ✓ It does perfect work on our task. However, the complex calculation makes it a program executed with no real-time feature.

2. Tools

Code written in Python. Mask images and detection model are required in the directory.

Other libraries we use:

- *dlib*

A great machine learning tool that helps us predict face features through a trained model.

- *cv2 (opencv)*

All about opening the camera, adding features on the screenshots and background remove.

- *Imutils*

Used to resize the display window size.

3. Implementation

(1) Face detection

Figure 1 Face detection effect.

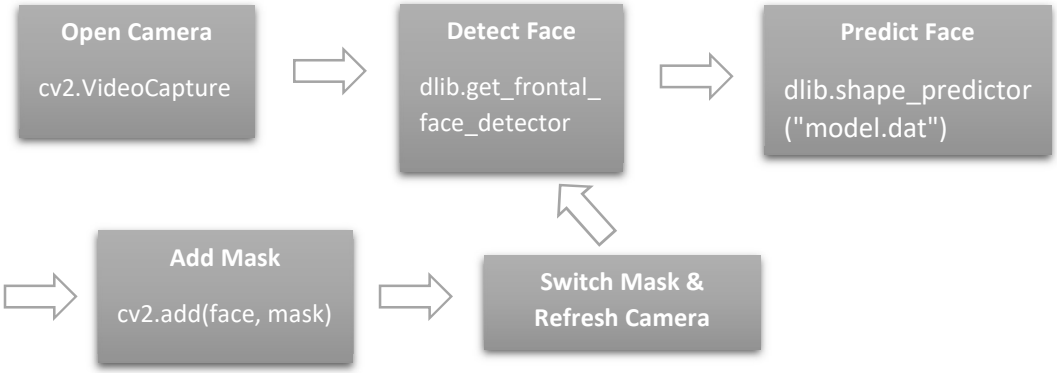

Chart 1 Face detection workflow.

(2) Face Morphing

Figure 2 Face morphing with mask highlighted.

Figure 3 Face morphing of front-face with perfect performance.

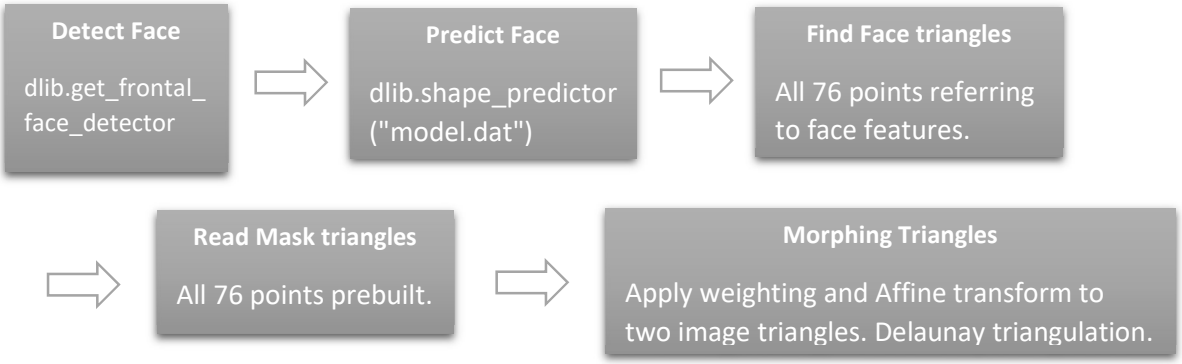

Chart 2 Face Morphing workflow.

※ Here we use the dataset from http://robotics.csie.ncku.edu.tw . They offer different head angles of poses so that we are able to deal with more situations. The dataset is given down below:

http://robotics.csie.ncku.edu.tw/Databases/FaceDetect_PoseEstimate.htm

4. **Execution – Team 8 on 6/26 demo**

(1) Real-time Face detection:
`Python test.py`
(2) Image-based Face detection:
`Python image.py [Face path] [Mask path]`
(3) Image-based Face morphing: (In faceMorph directory)
`Python faceMorph.py [Face path] [Mask path]`

Other revisable parameters:

(1) test.py – Detection Latency – `cv2.waitkey(time)`
(2) test.py – Detection Display size – `imutils.resize(frame, width)`
(3) faceMorph.py – Mask/Face transparency – `alpha`
(4) faceMorph.py – Face triangle manual correction – `offset`

5. Reference

FaceTracker, kylemcdonald, https://github.com/kylemcdonald/FaceTracker
利用 Dlib 訓練 Shape Predictor, HARDLIVER, August 3, 2017, https://hardliver.blogspot.com/2017/08/dlib-dlib-detector.html
ImgLab, NaturalIntelligence, https://naturalintelligence.github.io/imglab/
Face Morph Using OpenCV — C++ / Python, Satya Mallick, March 11, 2016, https://www.learnopencv.com/face-morph-using-opencv-cpp-python/