Background

Face reenactment is a fairly interesting research topic in the field of computer vision. In general, it takes images or live video feed as input, detect faces in them, and superimposes these faces, including facial movements, to another person in a natural-looking way.

Traditional 3D model based approach[1] requires massive computational resources, and no open source solution is available online. However, recent appearance of GAN (Generative Adversarial Networks) made it possible to do image translation with deep learning. Our project is based on Pix2Pix[2], an open source image translation framework using GAN, and Face2Face[3], an open source face reenactment application.

A GAN consists of two neural networks, one learns to generate fake samples that are close to target, the other learns to discriminate the generated fakes and the authentic targets.

Methodology

Pix2Pix aims to solve the image translation problem. The training data contains a source and a target, and Pix2Pix trains a GAN on such data. The trained generator can then construct pictures given an input image. The input image must have similar context with the source images in the training dataset to get satisfactory results.

We used the method introduced in Face2Face to train the GAN model for face reenactment and translate input images using the trained models. First, we preprocess frames captured from a video, selecting frames that have a face in it, and generate abstracts of facial landmarks.

Second, we train the GAN that is used in Pix2Pix, with the source being the facial landmarks, and the target being the video capture.

Third, we get the testing image from camera stream and generate its corresponding facial landmark. The facial landmarks are transformed to match the characteristics of facial landmarks in the training data.

Last, we use the transformed facial landmarks as input to the trained GAN generator, and get our facial reenactment result.

System Design

Below is the design diagram for our system.

Our design is based on Gabriel, a cloudlet-based computation platform that supports offloading of mobile workloads. More specific, the front end and the backend are modifications of Gabriel’s front end and backend.

Elements in solid lines represents the major workflow of our system and are all implemented. Elements in dashed lines represents the alternate workflow and are potential future improvements on the system. Detailed workflow is listed below.

1. The backend controller gathers performance data and dynamically adjusts the fidelity of the output to display on the frontend.

2. Upon successful connection, the camera on the frontend sends the frame to the controller on the backend.

3. The preprocessor on the backend preprocesses the input video and send result to the Face2Face algorithm implementation on the backend.

4. The Face2Face algorithm implementation on the backend processes the video and streams back the output to display on the frontend.

5. The backend controller gathers performance data and dynamically adjusts the fidelity of the output video stream.

References

