CIS-5810 - FINAL PROJECT PROPOSAL

Track 1 - Face Swapping in Video

Members

1. Tri Le - trilev@seas.upenn.edu
2. Vyaas Valsaraj - vyaas@seas.upenn.edu
3. Jae Young Lee - jaeyoule@seas.upenn.edu

1. Project Summary

The objective of this project is to automatically detect and swap faces between two videos. The following steps would be involved in the completion of the project (See pipeline for more details):

1. Selecting another video with a face to swap to.
2. Detecting the face and facial landmarks in the two videos.
3. Appropriate feature extraction so as to retain the source face's emotions and suppress the target's.

Methodologies of this project will mainly be conventional computer vision techniques, such as KT optical flow, gradient blending, etc. OpenCV, Dlib and several other open source libraries will provide the framework for this project.

2. Project Goals and Objectives

- Swap face of a replacement video with a face from a source video:
    - Emotion of source video is preserved and carried over.
    - Visually acceptable (exposure, blending, etc.).
    - Facial landmarks matching are good in terms of composition and orientation.
    - Program should work adequately with different types of videos (increasing order of difficulty in facial tracking):
        - Videos in which mostly faces are visible and are the main focus (Youtuber, interviews, news reporter).
        - Videos in which bodies and face are visible, both are main focuses.
        - Videos in which there are multiple actors and sceneries, no real main focus.
- Execute in a timely manner:
    - Ideally work in real time processing.
    - Python based.
- Final project report is to be written with Latex.

3. Proposed Approach + Extras

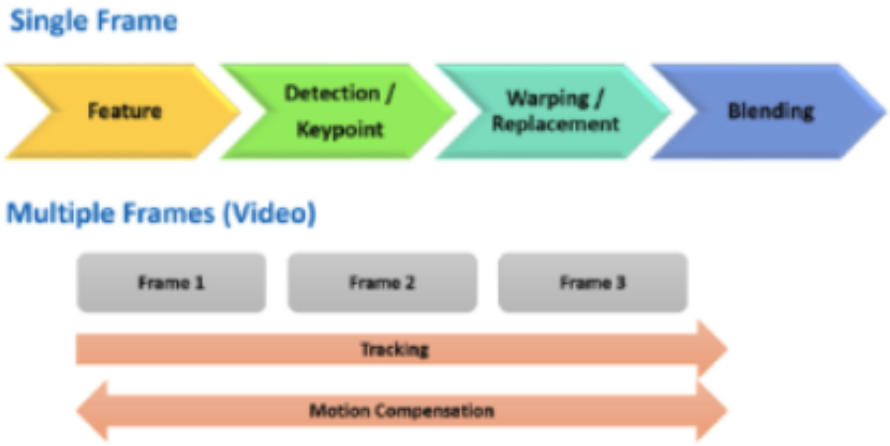

Figure 1: Possible pipeline for face swap

Pipeline

Pipeline will most likely be based on the one provided in the Final Project instructions.

1. **Feature / keypoint detection - some of the available choices:**
    a. OpenCV's Face Module [https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html]
        i. Eigenfaces uses Principal Component Analysis (PCA). Eigenface is an orthogonal basis set of faces. The eigenfaces are created using a set of real faces. Then, the minimum number of eigenfaces that represents the entire training set is found.
        ii. Fisherfaces is a method based on Eigenfaces. After the PCA method is applied, Linear Discriminant Analysis (LDA) is used to obtain features of image characteristics. The LDA method uses a set of faces with labels.
        iii. Local Binary Patterns Histograms extracts features in a low-dimensionality. The Local Binary Patterns obtains the local structure by comparing each pixel with neighboring pixels.
    b. Dlib
        i. Get_frontal_face_detector (HOG + Linear SVM).
        ii. cnn_face_detection_model_v1(MMOD CNN). [https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/]
        iii. Extract specific facial features (eyes, nose, lip, eyebrows, etc.).

[https://pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/]

c. Python's DeepFace Module is another viable option for facial feature detection.
d. Optical flow (pyramidal KT tracker) is to be run in parallel with feature detection. Tracking provides necessary motion information in case facial feature detection misbehaves. Feature detection and optical flow provide checking and act as insurance for each other to ensure robustness of the program.

2. **Feature extraction:**
    a. Sampling mostly points along the convex hull / perimeter to avoid unwanted feature extraction and source emotions.
    b. Add control to regulate the inner points selection. It may be necessary to isolate and extract specific facial features - such as eyes, mouth, nose.
    c. Emotions are portrayed mostly via eyes and mouth. Nose typically remains stationary.
    d. Delaunay triangulation may be useful.
3. **Warping / replacement:**
    a. Finding transformation:
        i. Perform feature / keypoint detection on source and target's frame.
        ii. Use these landmarks to find transformation from source to target.
        iii. Additionally, maintain optical flow tracking to ensure the fidelity of the frame and provide back-up transformation if facial features detection fails.
    b. Homography transformation:
        i. Warp source features to fit them into target.
        ii. Blend source to target using Gradient blending.

Extra / Possible optional tasks

1. Add additional sources of face
    a. Combine difference sources into a blended sources
    b. Perform facial swap of blended source with target
2. Introducing microexpression into some frame
    a. Expressive facial features (mouth, lips, eyes, etc.) should be emphasized during the swapping process.
    b. These expressions are inserted to a small interval of frames only. This implementation attempts to make the emotions more convincing and natural.
3. In-situ face swap. Ie. If the test video contains multiple faces, perform swapping on those.
4. Compare results with other open source learning based face swapping
    a. Public repository [https://github.com/topics/faceswap]
    b. Faceswap software [https://forum.faceswap.dev/app.php/faqpage#f0r0]

Related Works

- Dlib
  [http://dlib.net/]
- Optical flow - pyramidal KLT
  [https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html]
  [http://robots.stanford.edu/cs223b04/algo_tracking.pdf]
- Gradient image blending
  [https://www.cs.jhu.edu/~misha/Fall07/Papers/Perez03.pdf]
- Homography transform
  [https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html]
- Thin plate spline transform
  [https://github.com/yoyo-nb/Thin-Plate-Spline-Motion-Model]
- Face parsing project
  [https://github.com/zllrunning/face-parsing.PyTorch]