VNU-HCM
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

PROJECT REPORT: FACIAL ACTION TRACKING

Subject: Pattern Recognition

Teachers:

- Associate professor Ph.D Lê Hoàng Thái
- Ph.D Nguyễn Ngọc Thảo
- Master Trương Tấn Khoa

Students:

- 20127121. Trần Ngô Gia Bảo
- 20127369. Lê Quốc Trung
- 18127042. Lê Phương Đào

I. KNOWLEDGE FROM BOOK

A. Parametric face modeling:

1. The primary factors can affects facial tracking issue are:
   - 3D motion and pose: 3D position and rotation of head.
   - Facial action: lip and eyebrow motion.
   - Shape and feature configuration: the shape of head, mouth, eyes,…
   - Illumination: lightning conditions.
   - Texture and color: skin
   - Expression: emotion.
2. Facial action coding system - FACS:
   FACS is the fruit of research in the field of psychologists. It describes 50 Action Units (AUs), with each being a combination of facial expressions. Each AU represents the activation of one facial muscle.
3. MPEG-4:
   MPEG-4 has two types of Facial Animation Parameters (FAPs): low-level & high-level. Key feature in the low-level is calculating distance between elements in face (Fig18.1). In high-level, it focuses on the points on the face (fig18.2).

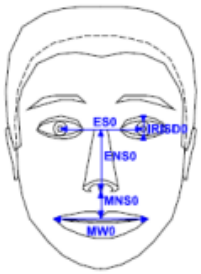

Fig18.1

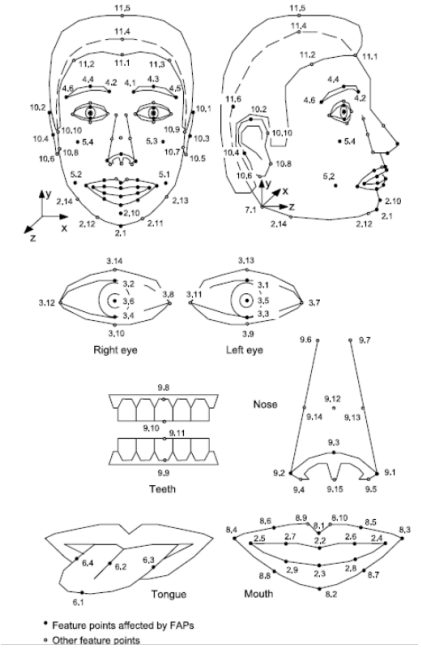

Fig18.2

4. Computer graphics models:
   When synthesizing face using computer graphics, the most common model is a wireframe model also known as polygonal mesh, which means vertex and edge.The polygons will “lock” onto the face. When the head moves, the polygons will follow that movement. Due to the large number of edges, and vertices we must apply Math to reduce them.
   The other popular way is just using vertex or point.

B. Tracking strategies:

1. Motion-Based vs. Model-Based Tracking:
   A motion-based tracker estimates the displacements of pixels (or blocks of pixels) from one frame to another. The motion of the object model is calculated from the estimated motion field using, for instance, least-squares, Kalman filtering, or any optimization technique. The object model is solely employed to convert the 2D motion vectors to 3D object model motion in this method; the motion estimation is ultimately based on the pixels in two frames. The drifting or lengthy sequence motion issue with such techniques is a problem. A model-based tracker, on the other hand, uses a model of the object's appearance and tries to change the object model's pose (and possibly shape) parameters to fit the new frame. Such a tracker does not suffer from drifting; instead, problems arise when the model is not strong or flexible enough to cope with the situation in the new frame.
2. Model-Based Tracking: First Frame Models vs. Pre-trained Models:
   The appearance model of ModelBased tracking is a reference image of the object captured from the first frame of the sequence. The image is geometrically transformed according to the estimated motion parameters, so one can compensate for changes in scale and rotation (and possibly nonrigid motion). Because the image is captured, the appearance model is deterministic, object-specific, and (potentially) accurate. A drawback with such a first frame model is the lack of flexibility—it is difficult to generalize from one sample only. This can cause problems with changing appearance due to variations in illumination, facial expression, and other factors. Another drawback is that the initialization is critical; if the captured image was for some reason not representative for the sequence (due to partial occlusion, facial expression, or illumination) or simply not the correct image (i.e., if the object model was not correctly aligned in the first frame) the tracker does not work well. Such problems can usually be solved by manual interaction but may be hard to automate. Note that the model could be renewed continuously (sometimes called an online model), so the model always is based on the previous frame. In this way, the problems with flexibility are reduced, but the tracker is then motion-based and might drift.
3. Features-based tracking:
   It uses pose parameterization and structure parameterization. The parameter estimation by Extended Kalman Filters (EKF). About the tracking process. The system uses a face detection algorithm to initialize the tracker and extract feature points. The feature points are selected based on the determinant of the Hessian, and their depth values are given by a 3D face model. For each frame, the model is rendered using the current model parameters, and patches are extracted around each feature point. The patches are matched with the new frame using a zero-mean normalized cross-correlation. The best match for each feature point is collected in the measurement vector and fed into the EKF update equation. The measurement noise covariance matrix can be estimated using the face model and the values from the normalized template matching. This tells the EKF in which directions in the image the measurements are reliable.In order to be both accurate and to handle drift, the tracker combines feature point patches from the first frame of the sequence with patches dynamically extracted from the previous frame.

4. Appearance-Based Tracking:
   appearance-based tracker estimaté the 3D pose and deformations of the face, it is based on Active Appearance Models (AAMs) and is improved later in various ways. To use the AAM search algorithm, we must first decide on a geometry for the face model, its parameterization of shape and texture, and how we use the model for warping images. Some improvement, Dornaika and Ahlberg introduced a feature/appearance-based tracker in 2002. The head pose is estimated using a RANdom SAmpling Consensus (RANSAC) technique combined with a texture consistency measure to avoid drifting. Matthews and Baker proposed the Inverse Compositional Algorithm in 2004 which allows for an accurate and fast fitting. Fanelli and Fratarvangeli incorporated the Inverse Compositional Algorithm in an AAM-based tracker to improve robustness. Zhou et al. and Dornaika and Davoine combined appearance models with a particle filter for improved 3D pose estimation

C. Fuse tracking:

1. Motion- and model-based Tracking combination:
   Lefèvre and Odobez used the Candide model and two sets of feature points. The first set is called the “trained set”, that is, a set of feature points trained from the first frame of the video sequence. The second set is the “adaptive set”, that is continuously adapted during tracking. Lefèvre and Odobez devised a hybrid method exploiting the advantages of both the adaptive and the trained method. Lefèvre and Odobez made a more thorough investigation of how to exploit the two methods. Another major difference is the choice of methods for 3D pose estimation from 2D measurements. Notably, Lefèvre and Odobez demonstrated stable tracking under varying lighting conditions.
2. Appearance- and features-based tracking:
   A recent development published by Caunce et al. which combines the feature-based and appearance-based approaches. The work was started with thorough statistical model-building of shape and texture of the face. Using profile and frontal photographs, a 3D shape model was adapted to each training subject and a PCA performed. A set of texture patches were extracted from the mean texture, and used to adapt the 3D model to a new image in a two-stage process. This method is primarily aimed at facial landmark localization/face model alignment, but is used for tracking of facial action as well by letting the parameter estimated from the previous frame serving as the initial estimation.
3. Commercial available trackers:
   A number of commercial products providing head, face and facial action tracking. Some of these include Eyematic, Seeing Machines, Visage Technologies, Omron, Image Metrics, Mova, Genemation, OKI, SeeStorm, and Visual Recognition. Face and facial feature tracking is also used in end-user products such as Fix8 by Mobinex and the software bundled with certain Logitech webcams.

II. THEORETICAL

- Facial action tracking is the process of identifying and measuring the movements of the face, it involves tracking the movement of specific facial muscles or regions. Facial

action tracking can be used for a variety of applications, including emotion recognition, animation, and virtual reality. It can also be used in fields such as psychology and neuroscience to study human behavior and cognitive processes. Some common techniques used for facial action tracking include facial landmark detection, which identifies specific points on the face such as the corners of the mouth, face and the edges of the eyes.

- SOTA paper for our approach: Attention Mesh [1] is our main SOTA paper and there are also some following related papers that we had mentioned in this report, presentation slide and during the presentation in class with Dr. Thao:
    - Real-time Facial Survey Geometry [2]
    - MobiNet [3]
    - BlazeFace [4]
    - OpenFace 2.0 [5]

A. MediaPipe motivation

- Mediapipe is an open-source, cross-platform framework for building machine learning pipelines to process real-time multimedia data. Developed by Google, it provides a set of pre-built, reusable building blocks for developers to create custom machine learning models for a variety of tasks, such as object detection, face detection, hand tracking, and pose estimation.
- One of the advantages of Mediapipe is that it can run on a variety of devices, including mobile phones, desktop computers, and embedded systems. This makes it suitable for a wide range of applications, such as virtual reality, augmented reality, and robotics.

B. OpenFace motivation

OpenFace 2.0 is an open-source software toolkit for facial behavior analysis, developed by the Computer Vision and Machine Learning Group at Carnegie Mellon University. It uses deep learning and computer vision techniques to detect and track facial expressions, head movements, eye gaze, and facial landmarks. OpenFace 2.0, which is an extension of the OpenFace toolkit, has the ability to handle a wide range of lighting conditions and facial orientations. The software is also designed to be more flexible and customizable, allowing users to fine-tune the algorithms for specific applications. For example, the software can be used to recognize a wide range of facial expressions, including subtle changes in facial movements.The new version of the software is designed to be faster, more accurate, and more robust than the original OpenFace. OpenFace 2.0 has been widely used in a variety of applications, including healthcare, entertainment, and human-computer interaction. The software has been used to study mental health conditions such as depression, post-traumatic stress disorder, and schizophrenia, as well as to improve human-robot interaction and driver safety in the automotive industry.

C. Ideas

1. MediaPipe and OpenCV

- We use mediapipe and opencv as a support library for our self-implement program. The main built-in function that mediapipe supports is solution.face_mesh(), this function will provide 468 points for facial landmarks on input image/video, then we will use opencv to handle these points.

2. OpenFace
    - Facial landmark detection and tracking

    OpenFace 2.0 uses the recently proposed Convolutional Experts Constrained Local Model (CE-CLM) for facial landmark detection and tracking. The two main components of CE-CLM are: Point Distribution Model (PDM) which captures landmark shape variations and patch experts which model local appearance variations of each landmark.
    - Head pose estimation

    CE-CLM internally uses a 3D representation of facial landmarks and projects them to the image using orthographic camera projection.
    - Eye gaze estimation

    A Constrained Local Neural Field (CLNF) landmark detector to detect eyelids, iris, and the pupil.
    - Facial action unit recognition

    A method based on a recent AU recognition framework that uses linear kernel Support Vector Machines. OpenFace 2.0 contains a direct implementation with a couple of changes that adapt it to work better on natural video sequences using person specific normalization and prediction correction.

D. Experiment

1. MediaPipe

   MediaPipe is a very strong and easy to use library that can be the base of some high-level application
2. OpenFace

   Overall the result of Openface demonstrates state-of-the-art performance on almost dataset comparison with methods in landmark detection, Head pose estimation, Eye gaze estimation and action unit recognition.

E. Discussion

1. MediaPipe

   After reading all related papers about mediapipe and core algorithms for these built-in functions, we had figured out how these built-in functions really work. Problem with these solution given by mediapipe that they don't break their solution pipeline into component and allow user to handle with each component to modify them, it will be better if they do it, though this is an open source library means we all can change everything inside but it would be more easy to use when we can a module program like that
2. OpenFace

Overall, OpenFace 2.0 has the potential to revolutionize various fields, including healthcare, entertainment, psychology and neuroscience,.... It is a useful tool for the computer vision, machine learning and affective computing communities and will stimulate research in facial behavior analysis and understanding. Furthermore, the future development of the tool will continue and it will attempt to incorporate the newest and most reliable approaches for the problem at hand.

III. PIPELINE

A. MediaPipe

- Face_mesh function in mediapipe include 2 main work are face detection then facial landmark

1. Detecting

- BlazeFace [4] is the main model that uses in detection phase, a lightweight and accurate face detector optimized for mobile GPU inference. BlazeFace models are suitable for applications like 3D facial keypoint estimation, expression classification, and face region segmentation. BlazeFace uses a lightweight feature extraction network similar to MobileNetV1/V2 [3]. BlazeFace also include 2 main version is for short-range image and full-range image
- This face detection phase will take an input as a full size input image/video and output is the face that cropped for next phase is facial landmark also tracking phase

2. Tracking

- We introduce 2 model that can be uses in this phase is Attention Mesh [1] and Real-time Facial Survey Geometry [2] you can choose by the parameter refine_landmarks in FaceMesh()

B. Openface

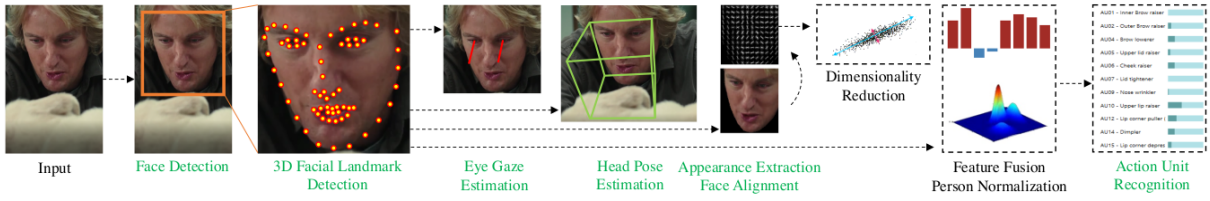

Fig 1: OpenFace 2.0 facial behavior analysis pipeline, including: landmark detection, head pose and eye gaze estimation, facial action unit recognition.

- Facial landmark detection and tracking: OpenFace 2.0 uses the recently proposed Convolutional Experts Constrained Local Model (CE-CLM) for facial landmark detection and tracking. The two main components of CE-CLM are Point Distribution Model (PDM) which captures landmark shape variations and patch experts which model local appearance variations of each landmark, in detail:
    - Facial landmark detection and tracking involves identifying and tracking specific points on the face, known as landmarks or facial keypoints, over time. These key points are used to describe the shape and position of the face such as the

corners of the eyes, the tip of the nose, and the corners of the mouth and are used in a variety of applications, including facial expression analysis, head pose estimation, and face recognition. Once the key points are detected, facial landmark tracking involves continuously updating their position and shape over time, even as the face moves and changes.

- Head pose estimation:They use the CE-CLM to take a 3D representation of facial landmarks and project them to the image using orthographic camera projection. This allows them to accurately estimate the head pose once the landmarks are detected.
- Eye gaze estimation: They use a Constrained Local Neural Field (CLNF) landmark detector to detect eyelids, iris, and the pupil. They use the detected pupil and eye location to compute the eye gaze vector individually for each eye. They fire a ray from the camera origin through the center of the pupil in the image plane and compute its intersection with the eye-ball sphere. This gives us the pupil location in 3D camera coordinates. The vector from the 3D eyeball center to the pupil location is our estimated gaze vector.
- Facial action recognition: They use the concatenation of dimensionality reduced HOGs from similarity aligned 112 × 112 pixel face image and facial shape features (from CE-CLM). In order to account for personal differences when processing videos the median value of the features is subtracted from the current frame. To correct for person specific bias in AU intensity prediction, we take the lowest nth percentile (learned on validation data) of the predictions on a specific person and subtract it from all of the predictions.

IV. PRACTICE

A. MediaPipe demonstration

- We use opencv and built-in function that given by mediapipe to build a real-time facial landmark tracking with the input can be come from webcam or video. Input video will be split into each frame and then we will apply mediapipe on each frame. face_mesh.process() of mediapipe will gave us 468 landmark point coordinate then according to the size of image we will draw a color point at that coordinate

B. Open face demonstration

In this project, we use OpenFace toolkit to perform facial action tracking. We use it to detect facial landmarks in images, tracking facial action units, facial landmarks, eyes gaze, pose head in a video with one person and even multiple people. We also perform a real-time facial tracking with input from our laptop's webcam. Below are the file structure of OpenFace and usage instructions.

1. OpenFace code layout

- ./lib
  - local - the actual meat of the code where the relevant computer vision algorithms reside
    - ➢ CppInterop - wrapper around C++ code for C# bindings
    - ➢ LandmarkDetector - The CE-CLM, CLNF, and CLM algorithms together with face tracking code

        - ➢ FaceAnalyser - Facial Action Unit detection and some useful code for extracting features for facial analysis
        - ➢ GazeAnalyser - Code for extracting eye gaze
        - ➢ Utilities - Utility code for image and video reading, result recording, result visualization, and rotation and image manipulation
    - 3rdParty - place for 3rd party libraries
        - ➢ CameraEnumerator - useful utility for listing and naming connected webcams
        - ➢ OpenCV3.4 - prepackaged OpenCV 3.4 library that is used extensively internally to provide support for basic computer vision functionality
        - ➢ dlib - a header only dlib library (includes the face detector used for in-the-wild images)
        - ➢ OpenBLAS - prepackaged OpenBLAS code for fast matrix multiplication
- ./exe - the runner and executables that show how to use the libraries for facial actions and head pose tracking, these best demonstrate how to use the libraries
    - FeatureExtraction - main workhorse executable for sequences - extracting all supported features from faces: landmarks, AUs, head pose, gaze, similarity normalized faces and HOG features
    - FaceLandmarkImg - main workhorse executable for images - extracting all supported features from faces: landmarks, AUs, head pose, gaze, similarity normalized faces and HOG features
    - FaceLandmarkVid - running single person landmark detection and gaze extraction on videos on disk or from a webcam
    - FaceLandmarkVidMulti - tracking multiple faces in sequences and extracting their features (from a webcam, video file or an image sequence)
    - releases - scripts for packaging the Windows binaries into a release
- ./gui - the Windows GUI executables
    - HeadPose-live - tracking and recording head pose, very specialist use
    - OpenFaceDemo - nice visualization of OpenFace results from a webcam
    - OpenFaceOffline - GUI for processing videos, image sequences, collections of images and webcam feeds, same functionality as that of FeatureExtraction
- ./matlab_runners helper scripts for running the experiments and demos, see ./matlab_runners/readme.txt for more info
- ./matlab_version A Matlab version of parts of OpenFace together with some training code, more details in ./matlab_version/readme.txt

- ./Release The created directory after compilation containing the desired executables for x86 architectures
- ./x64/Release The created directory after compilation containing the desired executables for x64 architectures
- ./model_training
    - AU_trainning - contains facial expression training modules
    - Bounding_box_mapping - how to convert from arbitrary bounding boxes of face detectors to ones needed by OpenFace
    - CCNF - CLNF and SVR patch training code
    - CE-CLM_trainning - CEN patch expert training
    - Learn_error_mapping - learning how much to trust the CE-CLM likelihoods at different views to help choose best hypotheses
    - pdm_generation - code for training the Point Distribution Model (PDM)

2. Usage instructions
    a To detect and track facial feature in a video with single person, execute the following command on the command line:

$$FeatureExtraction.exe\ -f\ <input\ file\ path>$$

    b To detect and track facial feature in a video with multiple people, execute the following command on the command line:

$$FaceLandmarkVidMulti.exe\ -f\ <input\ file\ path>$$

    c To extract facial feature from multiple images, execute the following command on the command line:

$$FaceLandmarkImg.exe\ -f\ <input\ file\ path>\ -f\ <input\ file\ path>$$

V. REFERENCES

- [1] Grishchenko, I. (2020, June 19). Attention Mesh: High-fidelity Face Mesh Prediction in Real-time. *arXiv*. https://arxiv.org/abs/2006.10962
- [2] Kartynnik, Y. (2019, July 15). *Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs*. arXiv. https://arxiv.org/abs/1907.06724
- [3] Zhu, M. (2017, April 17). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv. https://arxiv.org/abs/1704.04861
- [4] Kartynnik, Y. (2019, July 11). *BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs*. arXiv. https://arxiv.org/abs/1907.05047
- [5] Baltrusaitis, T. (2018, May 19). OpenFace 2.0: Facial Behavior Analysis Toolkit. *NSF PAR*. https://par.nsf.gov/servlets/purl/10099458
- [6] Google (2020). *MediaPipe Face Mesh*. github. mediapipe/face_mesh.md at master · google/mediapipe · GitHub
- [7] Google (2020). *Face detection guide*. MediaPipe. https://developers.google.com/mediapipe/solutions/vision/face_detector/?fbclid=IwAR0gEm7pTa7Y9u78vo6xj8dn5OyD2mxru-dnOU6qHXoVe1cG4qvBc5rhSrY