# CIS581: Computer Vision and Computational Photography
# Final Project
# Due: Dec. 10, 2020 at 11:59 pm

## Project Tracks

We provide you with two tracks for this final project. You can either propose an open-ended project or finish the face swapping project.

1. Track 1 (Face Swapping): You are allowed to make use of **open-source code and libraries** to achieve near-perfect results for face swapping. Please be thorough in citing anything you use. You must provide a comprehensive `README.txt` about the libraries that need to be installed and how to run your code. You will have to address the challenges of exposure, lighting, and shadows, 'crowd' faces, etc. We will grade you based on visually pleasing results. The detailed instructions for this track are shown below.

2. Track 2 (open-ended project): This track of the final project is open-ended, you can formulate your own problem, so long it is computer vision-related. We encourage you to use the techniques that you will learn and implement in this class, such as Edge Detection, Warping, Morphing, Seam Carving, Blending, Image stitching, Optical Flow, and Deep Learning. You can add any features you wish to. The code implementation should be based on some of your own implementation from previous projects and open-source libraries. The project scale should be similar to track 1.

## Logistics

- The final project is a **team** project. The maximum size of a team is **six** students.
- There are **two** intermediate **Milestones**, a) initial project proposal, b) mid-point project update. You will be assigned one of our TAs as your project mentor one week after your initial project proposal. The instructor will be meeting with each team periodically to discuss project progress and provide feedback.
- You are encouraged to reuse as many of the weekly project modules to put this final project together. You are also allowed to use open source replacement to achieve a better result.

## Milestones

- Initial proposal due on **Sept 15th**.
  - You need to turn in a pdf file up to 2 pages long: including the following sections: 1) Project title and members, 2) Project summary, 3) Goal and objective, 4) Related works, 5) Proposed Approach.
  - Include a timeline and duty for each group member.
  - Include an optional PPT presentation of your project.
  - Submit it via Gradescope.
- Mid-point project report due on **Oct 22nd**.
  - You need to turn in a pdf file up to 3 pages long including the sections as in the initial project proposal.
  - List progress, questions and future plan for the final project.
  - Include an optional PPT presentation of your project.
  - Submit it via Gradescope.

# Final Project Submissions

- Final Submission due on **December 10th**.
  - submit a five minutes video presentation.
  - Write a report to your final project up to four pages long.
  - Submit your code and README file on Canvas.

# Instructions for Both Tracks

- You **must make one submission** on Canvas. You **must** include a `README.txt` file in your submissions to help us execute your code.
  - Place your **code** and **resulting videos** into a folder named "CIS 581 Final Video". Submit this as a zip file named `<Group_Member>_final_project.zip`
- Your submission folder should include the following:
  - .py scripts for the required functions
  - .py demo script for generating output videos, if applicable
  - .py files with helper functions
  - the input video/s you use, if applicable
  - the resulting video/s from the test set (if your videos are large, please feel free to upload them to YouTube and share the link with us)
  - .pdf report containing an introduction, methodology, results, future work and references to papers and/or code
  - .pdf or .ppt file of the final video presentation (methodology (algorithms, pipeline, libraries), video results).
- This main programming language for this project is Python. You can use any package such as Pytorch, Numpy and OpenCV. You need to provide a requirements.txt to specify packages you use. You can use Jupyter notebook if you like.
- **Start early!** If you get stuck, please post your questions on Piazza or come to office hours!
- **Follow the submission guidelines and the conventions strictly!**

# 1 Face Swapping in Video

## 1.1 Overview

The aim of this project is to automatically detect and swap faces between videos. Given two videos, you will automatically swap faces between the two videos. This project is highly open ended, so how you formulate it is completely up to you. However, we do want to see at least one set of videos where the face from one video has been replaced onto the face in another video without the emotion of the target face coming through.

Seamlessly swapping faces is a non-trivial process. To swap faces between two videos, you will need to complete the given pipeline. Firstly, you will need to detect faces and/or facial landmarks in the two videos. Once you detect these features in the videos, you will have to estimate the transformation from one face to another. We suggest exploring concepts such as affine transforms or homography, or triangulation or thin-plate splines. Once, you have the transformation between the two faces, you may swap the two faces. You may also experiment with various optical flow methods such as Kanade-Lucas-Tomasi or Meanshift or Camshift if you have to track faces in the videos. You are not limited to these and are highly encouraged to play around with other approaches that you may find. You will have to compensate for the changes in exposure, lighting and shadows, etc. between the two videos while swapping the faces.

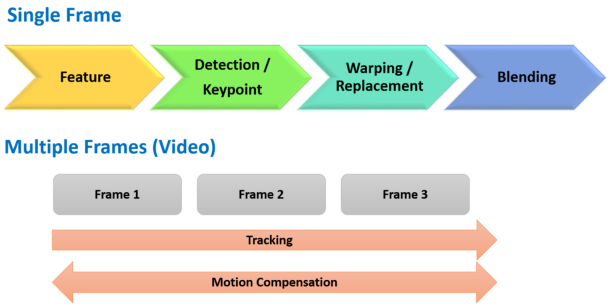


Figure 1: Possible Face Swapping Pipeline

## 1.2 Pipeline

One possible pipeline for swapping faces is as follows. After finding faces and facial landmarks, swap the two faces between the two videos. To leverage temporal information, you can experiment with various optical flow techniques.

Henceforth, the convention is as follows: We provide you with a dataset of videos containing face/s, to be called **source faces/videos**. The videos that you select to replace face/s, will be referred to as **replacement faces/videos**. You are required to swap faces between the two videos. However, while you replace the faces, keep in mind that the source face's emotion should be present on the target body. In other words, the source face should **not** emote the target face. This is the only mandatory requirement. Once you are done with this, you are allowed to experiment and showcase your creativity! *You are highly encouraged to shoot your own videos.*

The tasks in the pipeline can be as follows:

1. **Replacement Video/s Selection:** Select a second video/s with a face or faces to swap with.
2. **Source and Replacement Face and Facial Landmarks Detection:** Detect faces and facial landmarks in the source and replacement videos. We recommend OpenCV dlib library or any deep learning based facial landmark detection. You can find the open-source code on GitHub.
3. **Feature Extraction:** Since the emotion of the replacement face shouldn't seep through to the source face, the features you use to control the warp should just be the ones along the convex hull of the face. However, to control the warp better, you could use other features within the face as long as they don't change the emotions of the source face.
4. **Face Swapping:** For each frame, compute appropriate image transforms that warps the replacement face to the source face and vice-versa. Apply these transforms to both the faces. You may have to compute the convex hull of the source face and the replacement face while swapping the faces.
5. **Video Refinement** (Optional for Track 1): Make the face swapping seem natural. Use Gradient Domain Blending or any other technique that makes the swapped faces look real with their new bodies. Compensate for exposure, lighting and shadows, poses, skin tone, etc. Incorporate optical flow techniques to robustly track faces.
6. **Video to video replacement** Keep in mind that this is a video to video face replacement. The two videos may have the same number of frames or they may not. You should account for this while performing the replacement.
7. **Spiffify:** Eternal glory (extra-credit) will be awarded to teams that implement cool and creative features apart from the ones mentioned above. You can also attempt face swapping within a single video. For an example of the mandatory task done on a single video, see this video. Having your code run close to real time will be another way to earn a lot of extra credit!

You are by no means constrained to the example we have shown. This is the last project of the course and is very open-ended, so go crazy and have fun!

## 1.3 Scoring

We will release test videos of varying difficulty levels and your code is expected to work on those. You will be graded based on the results of your test videos. Tweaking your code to work with the test set should be trivial and we believe that four days should allow you do that.

- To receive full credit, your code must perform well on the easy videos in the test set.
- Robustly swapping faces in the medium and hard datasets will receive proportionate extra-credit.
- And of course, as mentioned earlier, extra-credits for cool and creative features.

## 1.4 Putting Things Together

Please do share the load, experiment with different methods and see what works best. The end goal is to create fun and creative face-swapped videos. Hope you guys have fun!