(Optional) Assignment 12

Please refer to MyCourses/Gradescope for assignment deadlines.

The goal of this assignment is to give you hands-on practice with the following concepts:

- Downloading and running vision models from online sources
- Creating an image processing 'pipeline' where outputs from one model are used as inputs to another
- Having fun creating a "filter" that adds sunglasses to faces.

Instructions

This problem involves:

1. detecting faces in an image. We require you to use OpenCV's `CascadeClassifier` for this. You will need to download a pre-trained face detector (XML file) from the internet. Fill in the missing code in `face_detector.py` for this.
2. for each detected face, detecting keypoints on the face. We'll use DLIB for this. Modify the missing code in `face_keypoint_finder.py`. Note that this file is smart enough to download the model automatically. Importantly, pay attention to whether the keypoint IDs are zero-indexed or one-indexed. when figuring out which keypoints to use for the sunglasses filter.
3. drawing a pair of sunglasses on the face. This is in some ways the hard part. You'll need to look into the DLIB model's documentation to figure out which keypoint refers to which part of the face, then you'll need to figure out how to warp the sunglasses image to "fit" the face. (Hint: the AnswerKey isn't perfect here, but what we've done is picked four keypoints from the face features and solve a homography to map the sunglasses coordinates to the face coordinates).

The bulk of the work is in the `sunglasses_filter.py` file. Running things happens in `run.py`, so you may start there to see how it will be called. The only files you need to modify are

- `face_detector.py`
- `face_keypoint_finder.py`
- `sunglasses_filter.py`

All OpenCV functions are allowed for this assignment. The challenge here is what is sometimes called "integration hell:" getting all the components to work together. Some of the `utils.py` files might be useful for, say, getting the OpenCV definition of a bounding box to play nicely with the DLIB code.

If all goes well, you'll generate images like this:

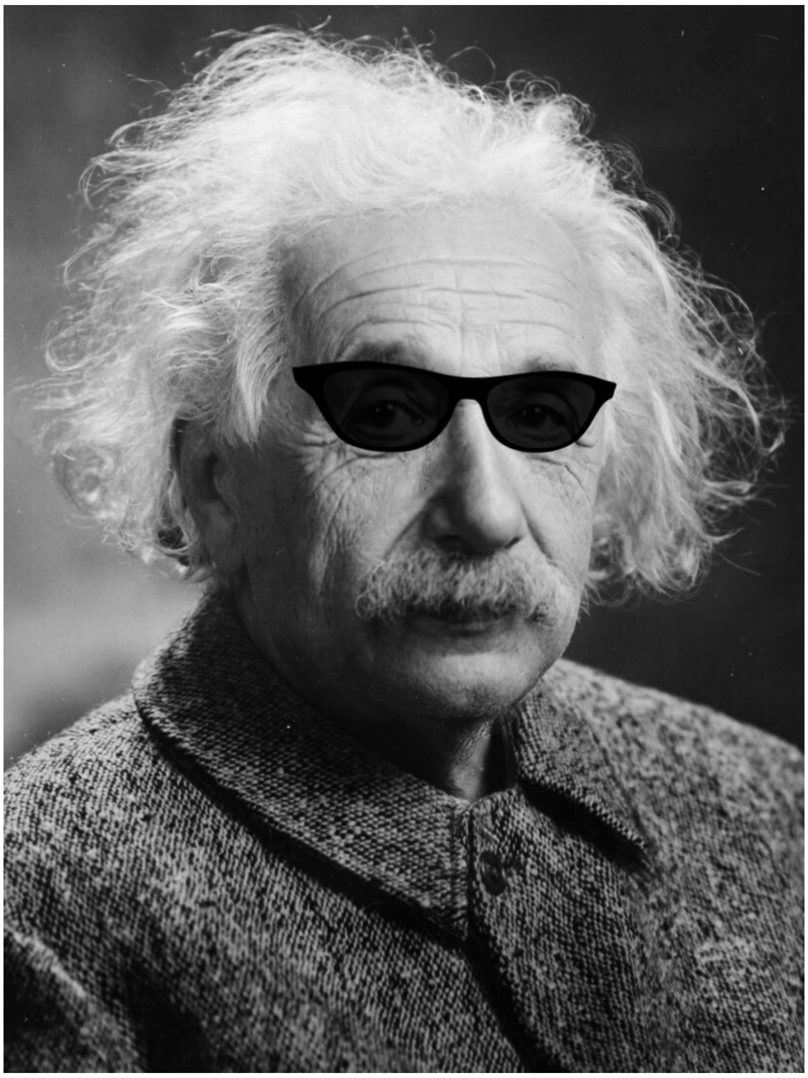

Figure 1: Output of `python run.py --image=images/einstein.jpg`

And you should be able to run it "live" like you would a snapchat filter. Note that you'll get more robust results if you define the face keypoints in terms of facial features that move around less (e.g. if you map the sunglasses to where the eyebrows are, then they will move/stretch as you raise your eyebrows).

This is not easily unit-test-able, so about 1/3 of the grade for this assignment will be based on how well it works in "live" mode.

Collaboration and Generative AI disclosure and Assignment Reflection

Did you collaborate with anyone? Did you use any Generative AI tools? How did the assignment go? Briefly explain what you did in the `collaboration-disclosure.txt` file. **Completing the**

collaboration disclosure for each assignment is required. Completing the reflection part of the file is optional, but it is a good way to give feedback to the instructor and grader about how the course is going.

Submitting

As described above, use `make submission.zip` to generate a zip file that you can upload to Gradescope. You can upload as many times as you like before the deadline. To account for late penalties, contact the instructor or grader directly if you plan on uploading any further revisions after the deadline.