Overview

The goal of this project is automatic detection and replacement of faces in videos. Given a test video, you will automatically replace a target face. Ideally, you should do this in the most stable way possible.

Seamlessly face replacement is a non-trivial process. For the rest of this document, we shall use the following terminology: face(s) in the video = target face(s), face(s) used for replacement = replacement face(s). First, you will build a model of the replacement face. You can use a single, well aligned photo, or a sequence under different poses. You will have to devise a scheme to be able to detect and localize similar looking faces in video frames. One way to do this is with feature detection, matching, and voting. Once you detect a face in a test image, you must estimate the warping between the candidate replacement face(s) and the detected candidate target face(s). We could suggest extending previously explored concepts such as simple affine, warp, or you can allow additional deformity using TPS. You are of course not limited to these and are encouraged to experiment with any other suitable approaches you may find. If you are using multiple faces for training, you should match against all of them and use the one that requires the least morphing.

Appearance matching may be necessary to compensate for shadows, lighting and skin tone. The target face will need to be cropped out, and a morph of the appropriate replacement should be inserted in its place, and the final video frame blended seamlessly. When dealing with videos, you might expect the faces in the video to move. For extra credit, you are also encouraged to explore methods for motion compensation.

We highly recommend you do this project with a partner. You may leverage code from previous projects.

Concentration Options

We provide you with options for this relatively open-ended project. You can either go deep into certain topics, or use 3rd party libraries to produce full face replacement result.

Option 1 Focus on a specific topic related to face replacement (eg, Voting, DPM, etc.), implement it on your own, apply it to the face replacement project, and analyze the result. For this option, you need to achieve a full face replacement result only using MATLAB built-in function. But we expect you to go deep into the certain algorithm. You will need to compare your results with one or more 3rd party implementation - you are welcome to seek advice on potential packages to use.

Option 2 You are allowed to use standalone 3rd party libraries (that can be compiled directly on Windows/Linux/OS X) to achieve satisfactory overall results for face replacement. You do not need to limit yourself to MATLAB, but please provide comprehensive instructions on how to use your code/program. You will be facing challenges on exposure, multiple face instances, face distortion, etc. We will evaluate your performance based on overall visual results.
Please specify your option at your checkpoint, presentation, and submission. You are welcome to consult with us before starting the project.

Procedure

Included on the course webpage will be links to download video clips that are of easy/medium/hard difficulty classifications. You can use these to develop and test your code. Four days before the project presentations, we will release new test videos and your code will be expected to work on these too. You will be graded based on the testing videos. Ideally, changing your code to work with the new set should not take much effort, but the 4 days should allow you some time for tweaking.

Again, you are encouraged to work with a partner on this project, as part A.

Schedule

- **Dec 8: Checkpoint** We expect some progress to be made on face detection at this point. You need to submit 3 slides in PDF form to the appropriate canvas submission portal. It should include the option you choose, all the team members’ name, a brief introduction of the system you plan to build, and some preliminary result.

- **Dec 17: Official Test Set** The official test set will be released. It contains both easy and very difficult cases. Please run your code on this for your presentation.

- **Dec 21: Presentations** Each team will present the results between 12am-4pm on December 21th. The presentation format will be the same as previous projects: 5 minutes/5 slides. You need to show at least one result video. The time limit will be enforced so we can get through everyone. If you want to do presentation earlier, you can come on 19th. The specific schedule will be post on piazza later.

- **Dec 23: Code and Report Due** Submit code and report to canvas with additional information on anything we should know to compile/run your code. Include all of your result videos, and also a script that runs your face replacement code on videos included in the official test set. The report should include sections as Objective, Algorithm, Implementation, Visualization/Analysis, Comparison, and Reference.

Tasks

One of the possible pipelines (Figure 1) for face replacement is as follows. For single frame, we first extract image features, and then utilize those local features to detect faces and facial keypoints. After localizing faces and keypoints, we warp the replacement face(s) into target face(s), and blend them together seamlessly.

To leverage temporal information, we can use tracking and motion compensation for multiple frames. This part will be counted as extra credits since it is not covered in class.

Specific tasks are as follows.

1. **Replacement Face(s) Selection** Select the face(s) you will be inserting over the test sets.

2. **Image Feature Extraction** Extract local features for further face detection. You can refer to lectures for details on HoG [1], Shape Context features [2], Haar-like features [3] and Harris Corners [4]. You can also refer to 3rd party libraries [5][6] for more features.

3. **Target Face Detection and Keypoints Localization** Attempt to locate instance(s) of faces in the test set using an automated detector. You can refer to lectures for details on Pictorial Structure [7], Chamfer Matching [8], Voting [9], etc.. You may use 3rd party libraries [10][11][12] (Option 2) or
implement it yourself (Option 1). You may either target the face that looks most like your own, or

target all faces in each test image.

As a start point, we provide a framework for face detection based on Voting [9]. Note that you
are not limited to this framework. If you want to follow it, we highly recommend you to read through
[9].

(a) Compute edge maps of training image (replacement face) and target image.

(b) Manually segment face mask from training image and extract shape context features only inside
the mask. Build up the codebook \(CE = \{ce_i\}\). Each codebook entry \(ce_i = (u_i, \delta_i, m_i, w_i)\) records
the feature for point \(i\) of the training image, where \(u_i\) is the shape context feature, \(\delta_i\) is the relative
distance of point \(i\) and the face center, \(m_i\) is a binary figure-ground mask for the patch centered
at point \(i\), and \(w_i\) is the weight mask computed from \(m_i\).

(c) Extract shape context features \(f_i\) of target image.

(d) For each codebook entry \(ce = (u, \delta, m, w)\) and a shape context feature \(f\) from target image, each
bin of \(f\) is weighted by mask \(w\) to remove the background clutter. Formally, the weight mask \(w\)
and distance function are computed by:

\[
w(k) = \frac{\text{Area}(\text{bin}(k) \cap m)}{\text{Area}(\text{bin}(k))}, \quad k = 1, 2, \ldots, nr \times n \theta
\]

\[
D_m(ce, f) = D(u, w \cdot f) = ||u - w \cdot f||^2
\]

where \(\cdot\) is element-wise product.

(e) Compare each shape context feature \(f_i\) from target image to every entry of codebook, and find
the best match \(ce_k\). Define the probability of feature matching as \(P(ce_k|l_i) \propto \exp (-D_m(ce_k, f_i))\).
Given the match of \(ce_k\) to \(f_i\), define the probability of center locating at \(c\) as \(P(c|ce_k, l_i) \propto \exp (-||c + \delta_k - l_i||^2)\). Accumulating the matching score over the whole image, the probability of
face with center \(c\) is given by

\[
P(c) = \sum_{i,k} P(c|ce_k, l_i)P(ce_k|l_i)P(l_i).
\]

\(P(c)\) gives a voting map of different locations \(c\). Face center hypothesis is given by prediction
with maximum score.

4. **Face Warping** For each instance of the target faces, find a deformable image transform that warps
the reference face to the detected instance. Apply this transform to the replacement face. An affine
warp or a TPS warp with low number of keypoints may be appropriate.

5. **Face Replacement** To automatically select the face region, you can use a variant of the seamless
carving algorithm you implemented in part A:
(a) First reorganize the image in log-polar coordinate, where the origin is center of face;
(b) Find a minimum-energy seam carving of the reorganized image in the angle direction. The first
and the last angle should have same carving radius;
(c) Transfer the log-polar coordinate to image grid, you will have a complete cycle which is the face
region.

Then replace the face region in the test image with the warped replacement face.

6. **Refinement**

Here are some directions to make the replacement more natural.

(a) Use Laplacian image blending/Gradient domain blending to get a seamless integration of the new
face.
(b) Find the appearance difference of the detected face to the warped version of the detected face. This
could be a per-pixel additive offset to compensate for shadowing and lighting of the target face.
Apply this brightness/color offset to the warped replacement face to achieve a better appearance
match.

7. **(Optional) Spiffify** Eternal glory (extra credit) will be awarded for teams that implement features
such as motion compensation and any other cool features you can think of. This is a VERY open
ended project, so go nuts and have fun!

**Scoring**

- To receive full credit, your code must perform well on the easy videos in the test set by your presentation
time.
- The optional tasks and successfully replacing faces in the difficult set will receive extra credit.
- Replacing the faces in the difficult set successfully will also compensate for not having working code
at the presentation time, but this will be difficult and is not a recommended alternative to finishing
before the presentation.
- And of course, as mentioned props (credit) for cool extra features

**Reference**

**Image Features**

1. Matlab extractHOGFeatures function
4. Matlab corner function
5. Matlab Feature Extraction
6. VLFeat (Matlab support available)
   [http://www.vlfeat.org/index.html](http://www.vlfeat.org/index.html)

**Face Detection**
10. Matlab Face Detection and Tracking (CamShift)
11. Matlab Face Detection and Tracking (KLT)
12. Matlab Cascade Object Detector
13. Face++ MATLAB API
    https://github.com/FacePlusPlus/facepp-matlab-sdk

FAQ

What to do in Option 1?
First of all, you should implement a specific topic related to face replacement which didn’t covered by MATLAB built-in function. Then apply it to the face replacement. There are two differences between Option 1 and Option 2: first is that your work will be evaluated mainly on the specific topic you chose, other than the face replacement visual result; second is that you can only use MATLAB build-in functions other than use 3rd-party libraries.

How much face coverage is required when morphing?
Morphing eyes, nose, cheek and mouth is sufficient.

Operating system compatibility
Some 3rd party libraries are only compatible with certain operating system. While we prefer general compatibility, we will accept codes that just work for either Windows / Linux / OS X. Please inform us of potential compatibility issues upon submission.

Can I use late days?
Late days may be used for the final code submission, but not the checkpoint or presentation; it is ok if your preliminary results/presentation are not very good if your final submission is good. If you and your partner have different number of late days left, you can use the average.

Can I reuse code from previous projects?
Yes, you are encouraged to do so.

Changing project option
You may change your option any number of times before the checkpoint - as you’re not expected to declare it until then. Any changes after the checkpoint will be considered on a case-by-case basis.

Limit of 3rd party libraries
If you use the 3rd party library that completes majority task in this project, you may ask first on Piazza.