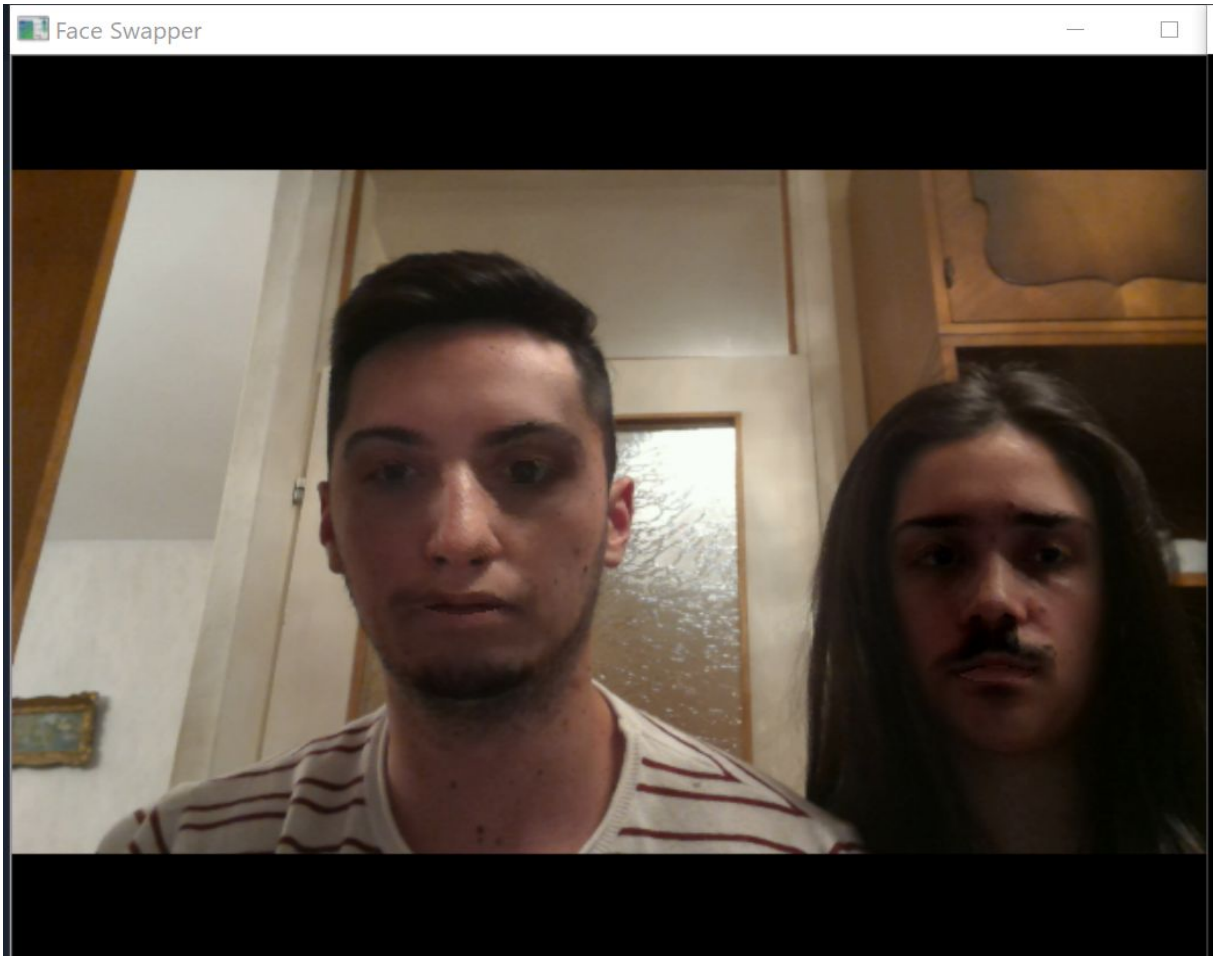
FaceSwapper

The app is written in Python and uses face alignment, Gauss Newton optimization and image blending to swap the faces of two persons seen by the camera.

Expectations

Input picture

Output Picture



How it works

The general outline of the method is as follows:

First we take the images (the image of a “left” person and the image of a “right” person) and find the faces region and its landmarks. Once we have that we fit the 3D model to those landmarks. The vertices of that model projected to the image space will be our texture coordinates.

Once that is finished and everything is initialized the camera starts capturing images. For each captured images the following steps are taken:

1. The face region is detected and the facial landmarks are located.
2. The 3D model of a “left” person is fitted to the located landmarks of a “right” person.
3. The 3D models are rendered using pygame with the texture obtained during initialization.
4. The image of the rendered model is blended with the image obtained from the camera using feathering (alpha blending) and very simple color correction.
5. The final image is shown to the user.

The most crucial element of the entire process is the fitting of the 3D model. The model itself consists of:

- the 3D shape (set of vertices) of a neutral face,
- a number of blendshapes that can be added to the neutral face to produce mouth opening, eyebrow raising, etc.,
- a set of triplets of indices into the face shape that form the triangular mesh of the face,
- two sets of indices which establish correspondence between the landmarks found by the landmark localizer and the vertices of the 3D face shape.

The model is projected into the image space using the following equation:

$$s = aP\left(S_0 + \sum_{i=1}^{i=n} w_i * S_i\right) + t$$

where *s* is the projected shape, *a* is the scaling parameter, *P* are the first two rows of a rotation matrix that rotates the 3D face shape, *S_0* is the neutral face shape, *w_1-n* are the blendshape weights, *S_1-n* are the blendshapes, *t* is a 2D translation vector and *n* is the number of blendshapes.

The model fitting is accomplished by minimizing the difference between the projected shape and the localized landmarks. The minimization is accomplished with respect to the blendshape weights, scaling, rotation and translation, using the Gauss Newton method.