Face Morphing

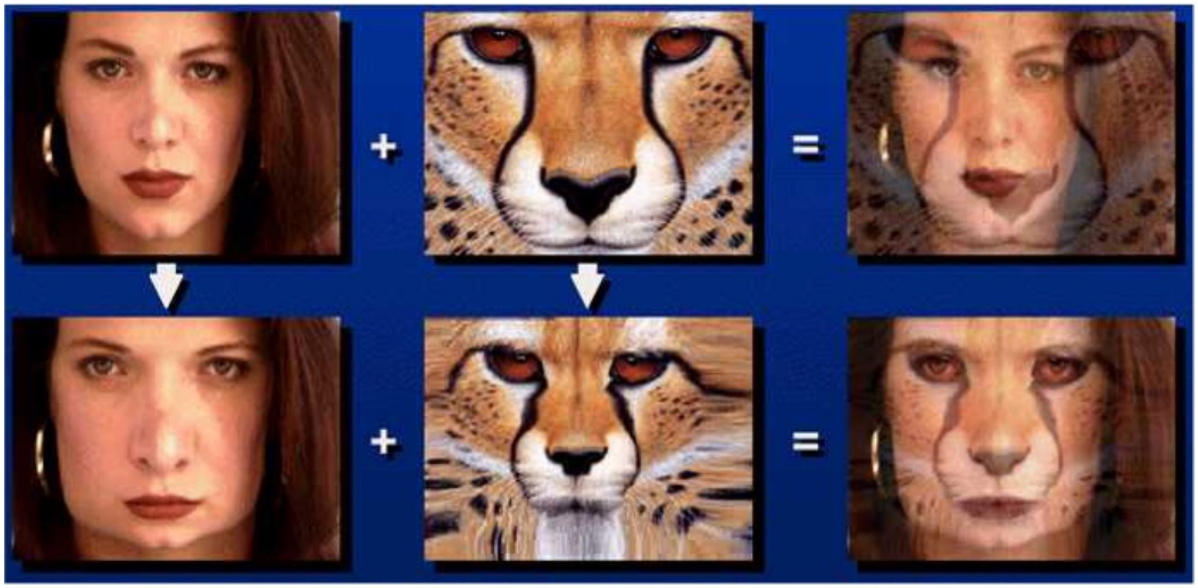
1° Assignment

Reference

- Ch. 3 of Szeliski

Morphing

- Morphing is the process of warping and blending two or more images



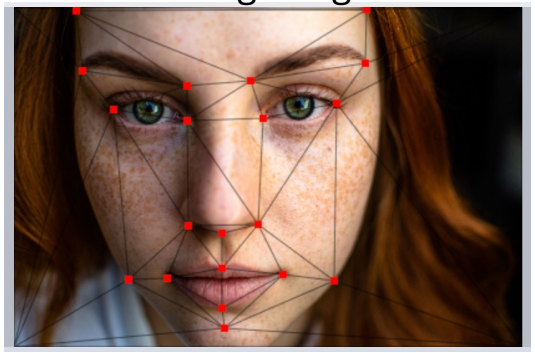
Cross-dissolving between two images leads to ghosting

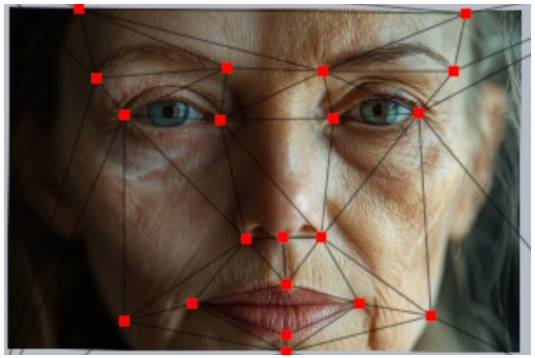
If instead we warp each image toward the other before blending, corresponding features in the 2 images are aligneed and no ghosting results

Morphing

- Given a set of correspondences between the two images, the alignment and blending is repeated using different blend factors and amount of deformation at each interval.
- This allows us to obtain a sequence of interpolated frames

floating image





reference image

Morphing

- To do face morphing we need to:
  - **implement a geometric transformation** to transform one image (floating image) into a transformed image that is «geometrically» to a reference image. You will use a piecewise affine transformation (see next slides)
  - **the transformation must be applied gradually** by generating a sequence of intermediate transformed images that, once assembled in a video, will give the effect of the transformation from the floating to the reference images
  - To change the face appearance, it will be enough to implement a **blending** function as a weighted sum of the transformed images

Geometrical Transformation

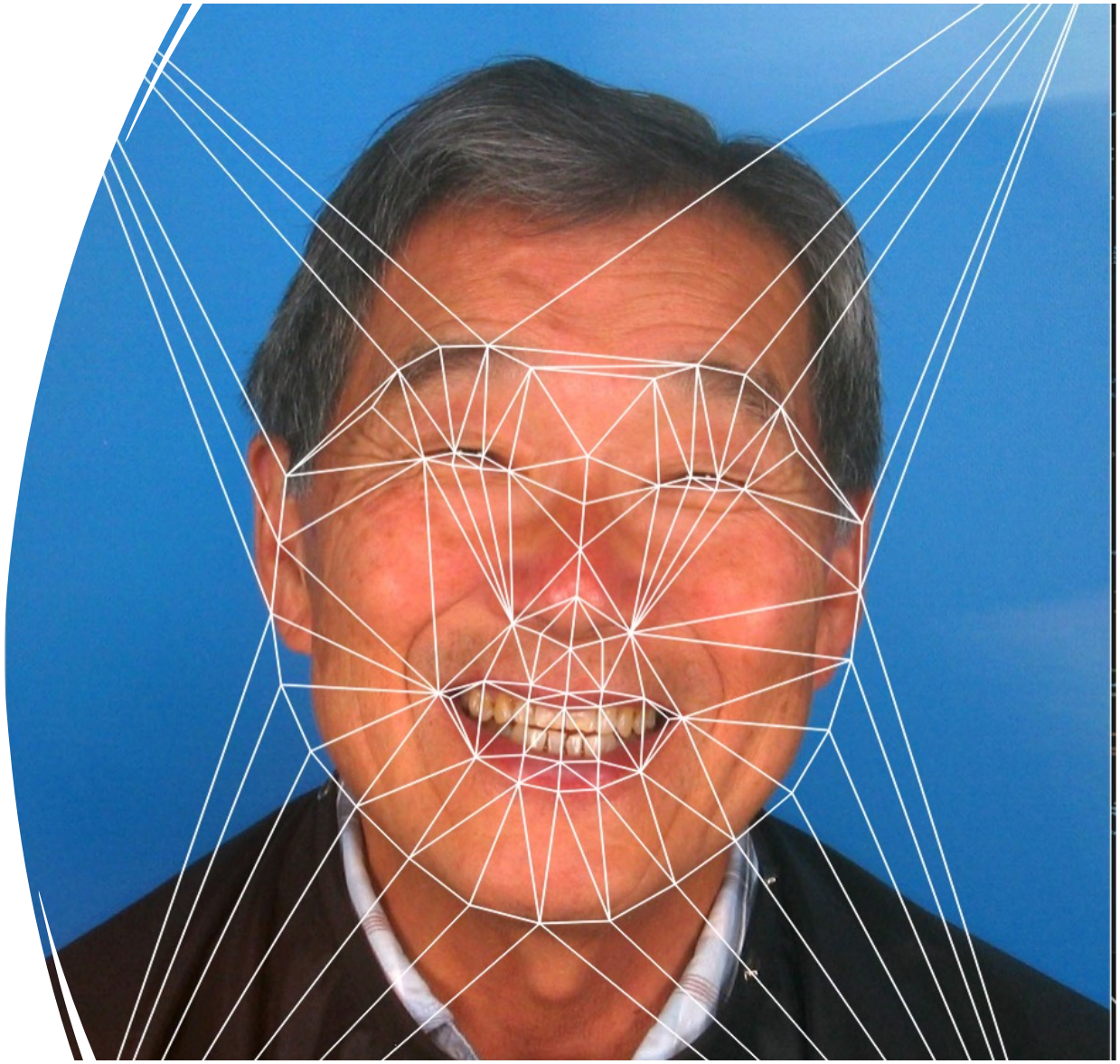
- We need to find point correspondences in the two images
- This can be done manually but, to make the process automatic, we can take advantage of face landmarks

Geometrical Transformation

- **From the landmarks, we can get a triangle mesh**. This can be done by a Delaunay Triangulation (if you never heard about it, check https://en.wikipedia.org/wiki/Delaunay_triangulation)

- Triangles are important because we can **transform the image by means of a piecewise affine transformation**

- You can **decide how many points to use to get the triangulation** (the showed one is an example)



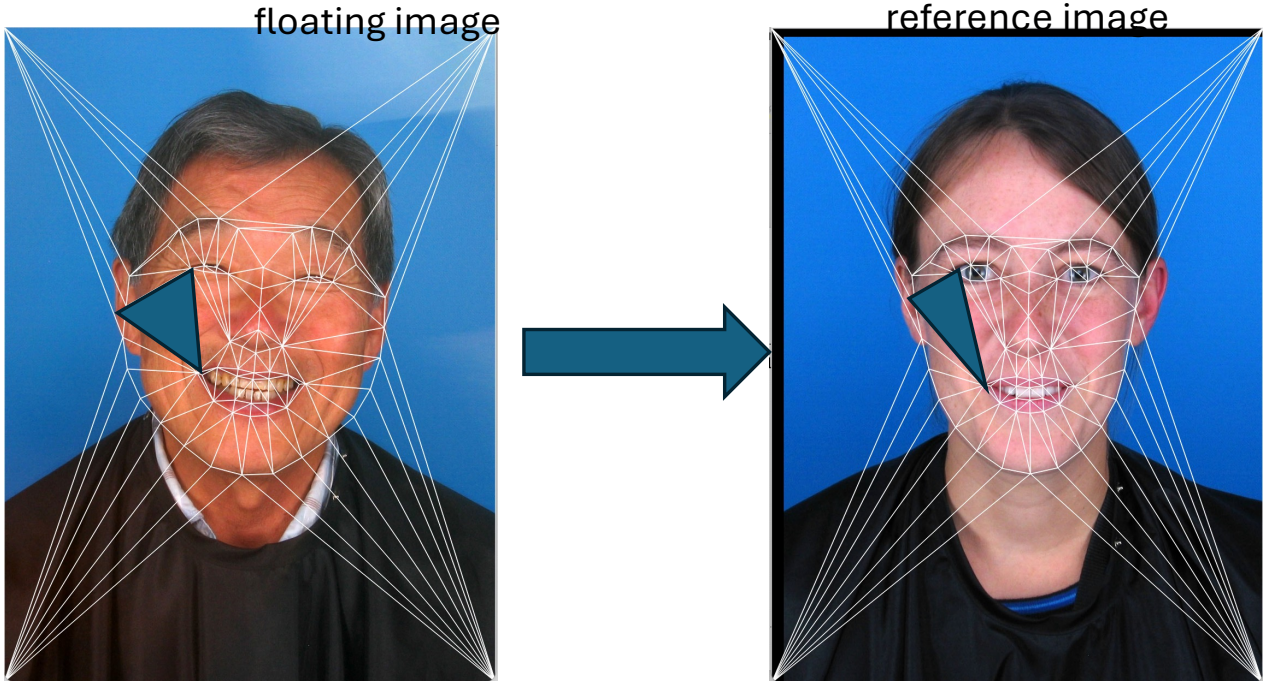
Geometrical Transformation

- We need to **transform a triangle of the floating image into a triangle of the reference image**
- Given the corresponding triangle vertices we can:
  - find an **Affine Transformation** (see lecture!)
  - **transform the pixel coordinates** of one triangle into the other
- Once this is done for each triangles pair

we can **use inverse mapping** to

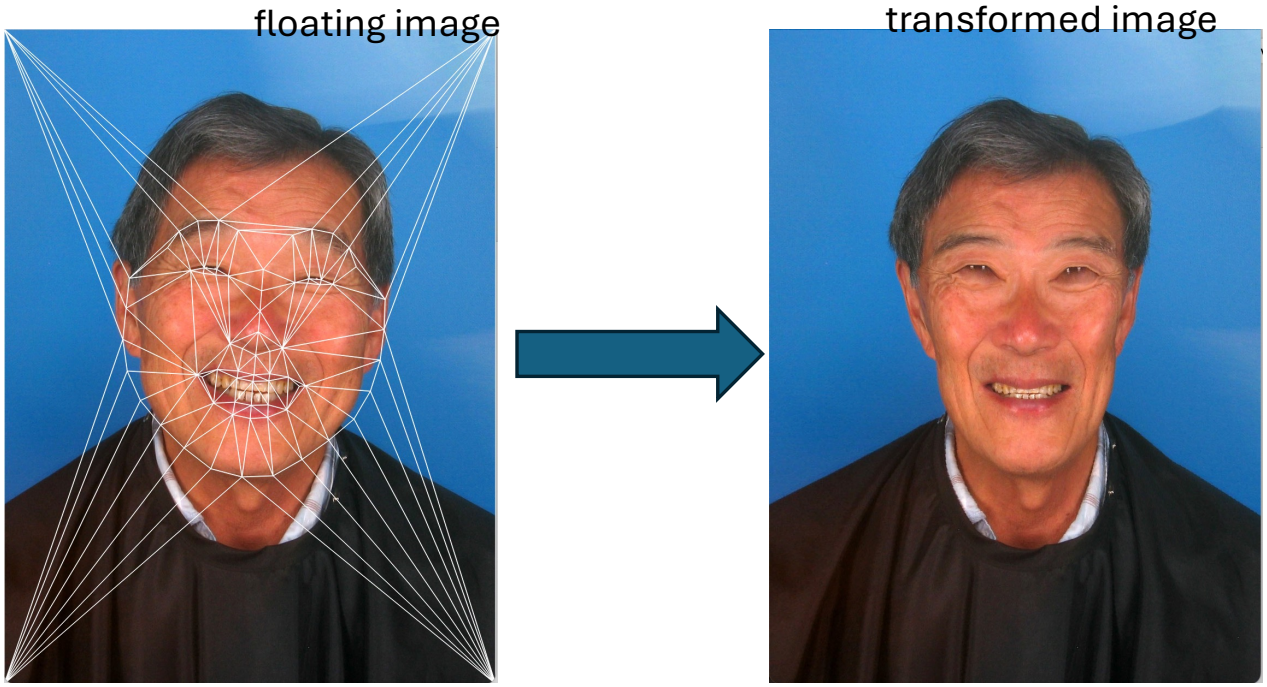
find the transformed image (see lecture!)

Let's call L_f the landmarks of the floating image and L_r the landmarks of the reference image, the piecewise affine transformation transform the floating image in such a way that the landmarks of the transformed image are aligned with L_r



Geometrical Transformation

- After applying the piecewise affine transformation by inverse mapping, we get an image whose face landmarks are aligned to the ones of the reference image



Geometrical Transformation

- **To make the transformation gradual**, we can **use a parameter *t*** *in [0, 1]* to modify the landmarks to be used in the transformation
- When applying the piecewise affine transformation, we can transform the floating image to map L_f into intermediate landmarks L_I

$$L_I = (1 - t)\, L_f + t\, L_r$$

- When t = 0, $L_I = L_f$ and the piecewise transformation returns the floating image
- When t = 1, $L_I = L_r$ and the piecewise transformation returns the transformed image whose landmarks are aligned with the one of the reference image (see image in previous slide)

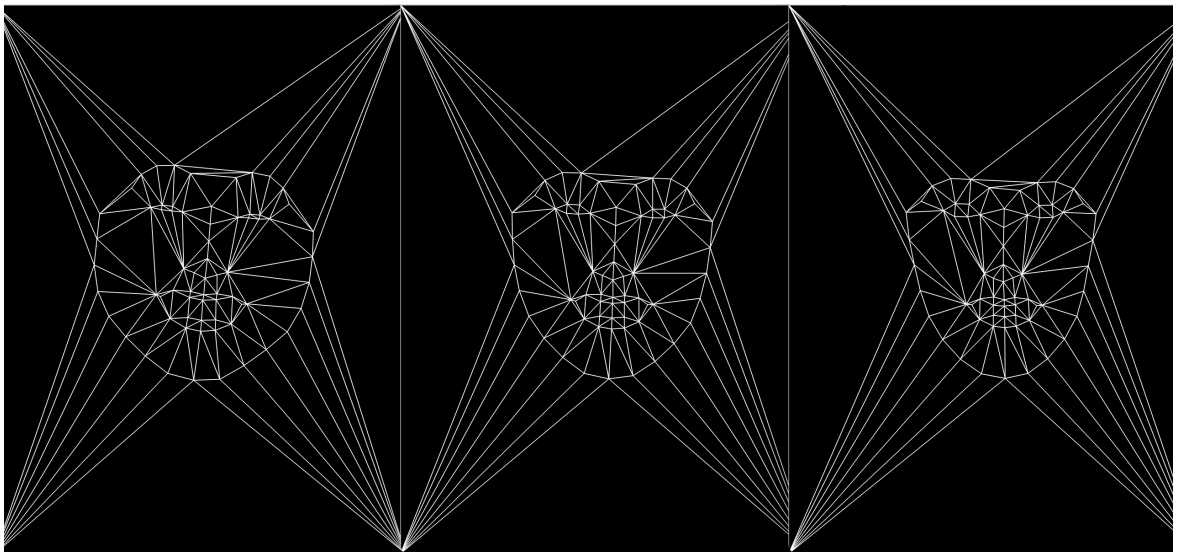
- By varying t in [0, 1] we get a sequence of transformed images

Face Morphing – slide 12

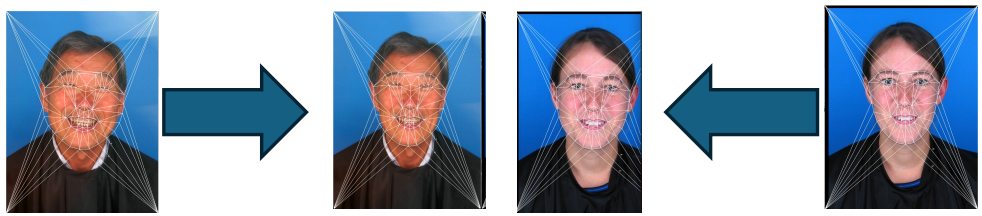
Delaunay Triangulation with landmarks L_f

Delaunay Triangulation with landmarks L_l

Delaunay Triangulation with landmarks



The first transformation aligns L_f to L_l



The second transformation aligns L_r to L_l

Blending – slide 13

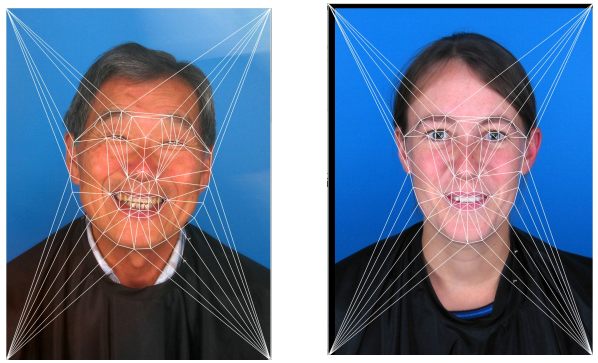
- Blending two images can be done easily by computing a weighted sum of the transformed images. The weights can vary along the sequence and should sum 1

(1 –t) *  + t *  = 

- Please, note that both the images are transformed!!

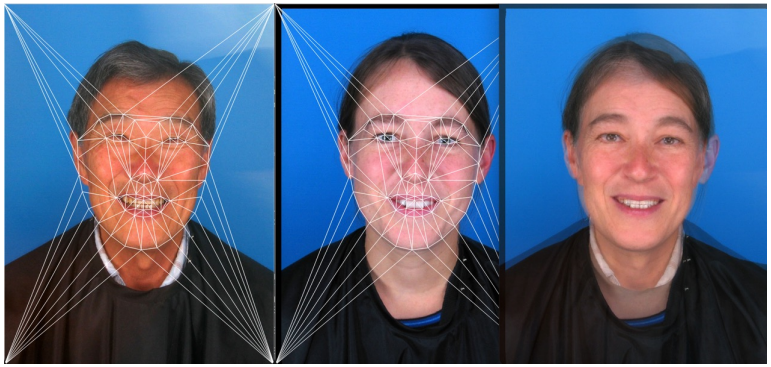
Face Morphing – Intermediate results

Initial floating and reference images and corresponding triangles

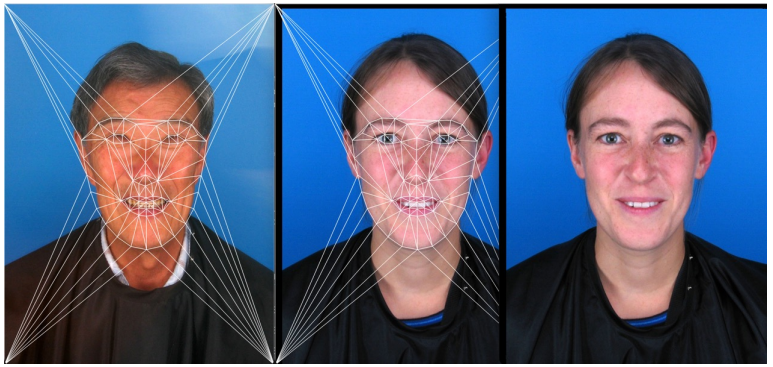


transformed floating, reference, blended with t=0.3

transformed floating, reference, blended with t=0.6



transformed floating, reference, blended with t=1

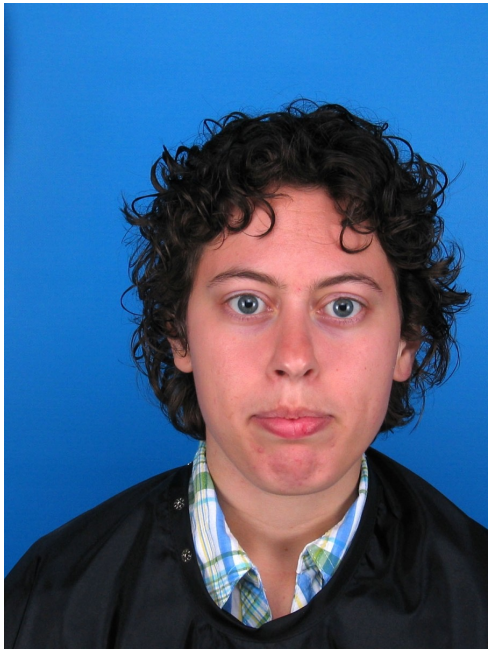


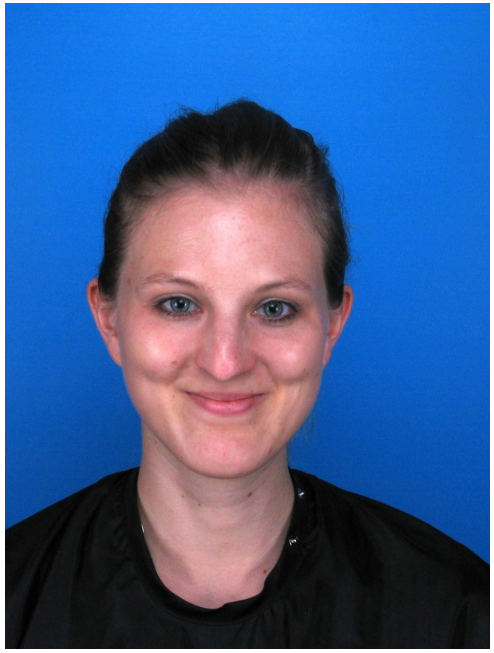
Face Morphing - sketch

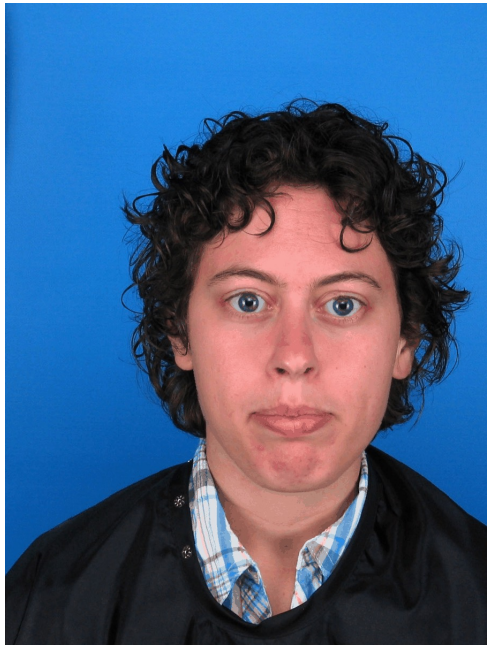
1. load the 2 images and ensure they have the same size
2. detect the faces and use them to align the images (translation + scale) such that the center of the faces in the two images overlap
   from openCV use: the faceDetector (*cv2.CascadeClassifier*) + transformation functions (*warpAffine*)
3. detect face landmarks by dlib *(Attention! dlib must be installed first! This is time consuming but it is worthy. You may need to install cmake (if you don't already have) and build the library locally on your machine. Follow the instructions on the library website (http://dlib.net)!)*
   from dlib, to detect the landmarks, use the model: *dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")*
4. use openCV to compute the delaunay triangulation (*cv2.Subdiv2D*). You will probably need to add some other points to get good results. *(How many times do you need to create a delaunay triangulation??)*
5. For varying values of t, compute intermediate landmarks and corresponding triangles
6. use **your own piecewise affine transformation function** to transform both floating and reference images considering the intermediate landmarks (slide 12). (*useful functions: cv2.fillConvexPoly, cv2.remap, np.meshgrid. Do NOT use functions of other librerie, implement your own version!*)
7. blend the two transformed images as detailed in slide 13 (*be careful to data types + opencv can have a useful function for doing it*)
8. Collect the blended images computed for varying values of t and create a gif *(I suggest imageio library because openCV does not support gif)*

Result 1

Result 2







Result 3