University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Alexandra Diaconu
Bogdan Alexe
MSC in Computer Science

Computer Vision - Laboratory class 5

Image Morphing

Objective

The goal of this exercise is to present a simple method for image morphing. Image morphing is defined as the animated transformation of one image into another. In this laboratory class you will create an animation that transforms one face (Bogdan's face) into another face (George Clooney's face).

We will work with pairs of images (see Figure 1) containing faces which are annotated with facial landmarks. The goal is to morph (transform) a face into another other face by creating a smooth transition between the source image and the destination image. The transition contains fake images obtained using geometric and photometric interpolation based on regions (triangles) selected from the two images. The method can be easily extended to other class of objects (different than faces).

In achieving our goal, we will follow the next steps:

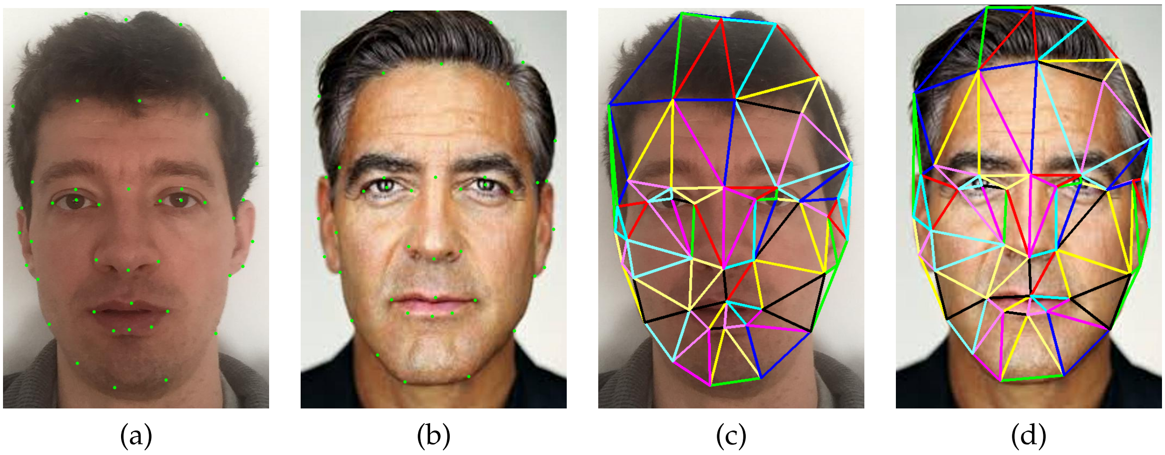
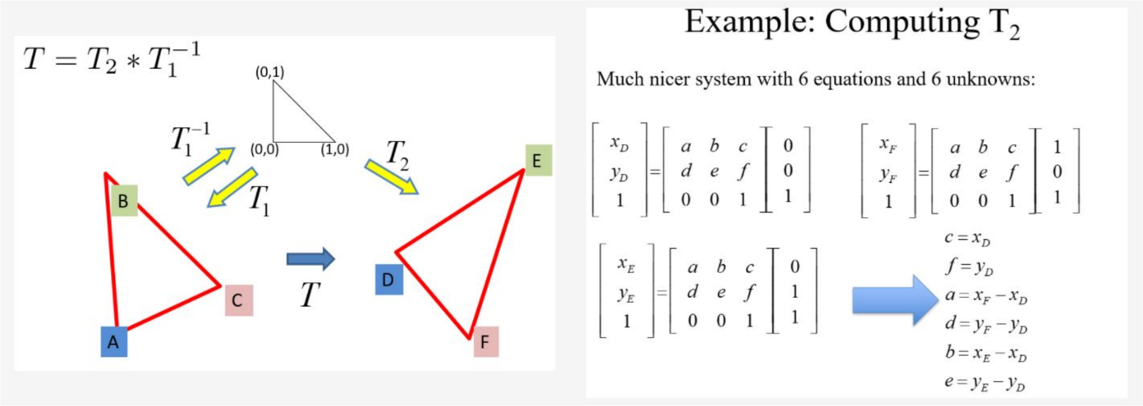

(a) (b) (c) (d)

Figure 1: *(a) source image (Bogdan's face) with facial landmark annotations shown; (b) destination image (George Clooney's face) with facial landmark annotations shown; (c) obtained Delauney triangulation for the source image; (d) transferred Delauney triangulation to the destination image.*



Example: Computing T_2

Much nicer system with 6 equations and 6 unknowns:

$$\begin{bmatrix} x_D \\ y_D \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x_F \\ y_F \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_E \\ y_E \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$c = x_D \quad f = y_D \quad a = x_F - x_D \quad d = y_F - y_D \quad b = x_E - x_D \quad e = y_E - y_D$$

Figure 2: *Computing the affine transformation that maps triangle ABC to triangle DEF. Using the intermediate triangle with points (0,0), (0,1) and (1,0) we obtain a very simple system with 6 equations and 6 unknown that can we solve immediately.*

a) compute the Delauney triangulation based on the facial landmarks in the source image, partitioning the face region in triangles. We transfer this triangulation to the destination image for corresponding facial landmarks;

b) compute the affine transformation between pairs of corresponding triangles from the two images (see Figure 2);

c) create fake triangles based on geometric and photometric interpolation of triangles selected from the source and destination images;

We compute the Delauney triangulation using existing function in OpenCV.

We can compute the affine transformation T between two triangles ABC and DEF using the formula derived in Lecture 9. We use the proxy triangle with vertices (0,1), (0,0) and (1,0) to get affine transformations T_1 and T_2. Then, we have that $T = T_2 * T_1^{-1}$.

We create fake triangles in our animation by:

- interpolating fake triangles at the geometric level wrt to the corresponding source and destination triangle. We do this for all triangle contained in the Delauney triangulation;
- for all pixels in the fake triangle we obtained their color by interpolating the RGB values of the corresponding pixels in the source and destination triangles/images.

Just interpolating triangles at the geometric and photometric level is not enough to obtain a nice animation. We also have to take into account pixels in the background. How would you address this problem?

Figure 3 shows the source and destination images (first and last frames in the animation) and intermediary fake images in the transitions.

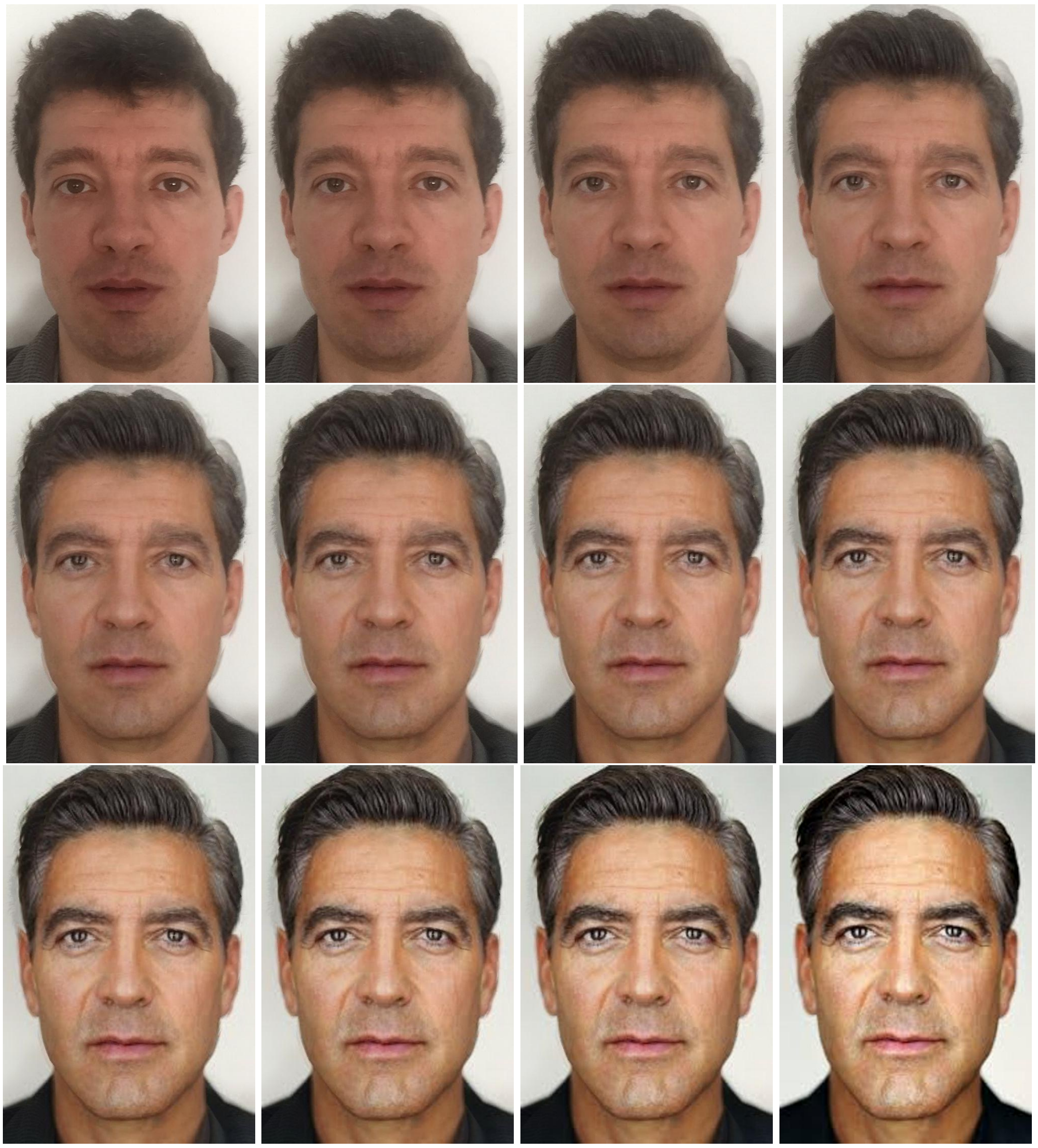


Figure 3: *We show selected frames from the obtained animation: first and last frames are real (source and destination images) while the rest of the frames are fake.*