BLG453E
Homework-2

04.11.2019

Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

Shamans who are angry with each other transform into bears, oxen and fight benefiting from many things (...). A male and a female shaman started one of their rituals. The female one asked "In which form should we test each other's strength?". "Fish form", answered the male one. The female one transformed into a fish form and jumped into the sea. The male one followed the same procedure and got to the female one in haste. The male one started to swallow the female one gasping her tail as a starter. However, she was not fitting for his mouth. After he slipped her out of his mouth, she jumped back just behind him. She grasped and swallowed him greedily, flew back to her home in the form of a seabird (...). She vomited the remains of the shaman out and threw them away.

Turkic Shaman Texts, Fuzuli Bayat

- You should write all your code in Python language.
- Cheating is highly discouraged.
- Ninova only stores files under 20 MB. If you could not upload your results, you can share them with me via Dropbox, or send me private YouTube video links for each part's results.
- In this homework, it is suggested to select your and a computer science veteran's photos. If you are not into using your own photo, you can use mine or Res. Asst. Enes Albay's.

1 - Part 1: Showing the landmark points (10 pts.)

In this homework, to beat our opponent, we will shapeshift into a dangerous and irresistible animal: cat. The dataset collected for an ECCV conference paper [1] by Zhang et al. will be very useful for this homework. Since the dataset is not officially available online anymore, I uploaded a part of it to my website. You can obtain the cat photos

[1] Zhang, W., Sun, J., Tang, X. 2008. Cat head detection-how to effectively exploit shape and texture features. In ECCV (pp. 802-816). Springer, Berlin, Heidelberg.

and respective landmark points under the given links [2]. For every cat image, there is a description file with extension **".cat"** which contains information of total point count, and positions of eyes, mouth and ears respectively. Some example cat images and their respective landmark points are given in Figure 1.

Figure 1: Some serious cats from the dataset and their landmark points

As a first step, you will obtain two photos containing human faces and another photo for a cat face, all of them having landmarks. Cat photo with landmarks can easily be obtained by reading a cat image and marking the pixels corresponding to the given points. For human faces, you can use *dlib*[3] library to obtain facial landmark points. It is necessary for the three photos to be of the same size. **Remember, if you crop or rescale a cat photo, the given landmark locations will also change.**

Dlib uses a very powerful face predictor[4] which uses a histogram of gradients (HOG) based model. By default, it outputs 68 landmark points from a face. An example skeleton code to find the landmarks is given below.

```
detector = dlib.get_frontal_face_detector()
#The detector object is used to detect the faces given in an image. It
    works generally better than OpenCV's default face detector.
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
#To predict the landmark points given a face image, a shape predictor with
    a ready-to-use model is created . The model can be found under "http://
    dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2"

image = cv2.imread("input1.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#The predictor only works on grayscale images.

rectangles = detector(gray)
```

[2] https://web.itu.edu.tr/sahinyu/CAT_00.zip. You can use filenames from *CAT_00* to *CAT_06* obtain every part of the set.

[3] http://dlib.net/

[4] C. Sagonas, E. Antonakos, G, Tzimiropoulos, S. Zafeiriou, M. Pantic. 2016. 300 faces In-the-wild challenge: Database and results. Image and Vision Computing (IMAVIS), Special Issue on Facial Landmark Localisation "In-The-Wild".

```
#Use detector to find a list of rectangles containing the faces in the
    image. The rectangles are represented by their xy coordinates.

#Check whether there is only one rectangle. Then, draw the rectangle on the
     image. To reach the values of the rectangle you can use functions
    given in the official documentation (http://dlib.net/python/index.html#
    dlib.rectangle). E.g. rectangles[0].bl_corner().x and rectangles[0].
    bl_corner().y will give one of the points. Show your work.


points = predictor(gray, rectangles[0])
#Points is a special structure which stores all the 68 points. By using
    points.part(i), we can reach ith landmark point. You should mark every
    point on the image. Show your work.
```

Write the code to obtain an image having all the three landmarked photos as given in Figure 2.

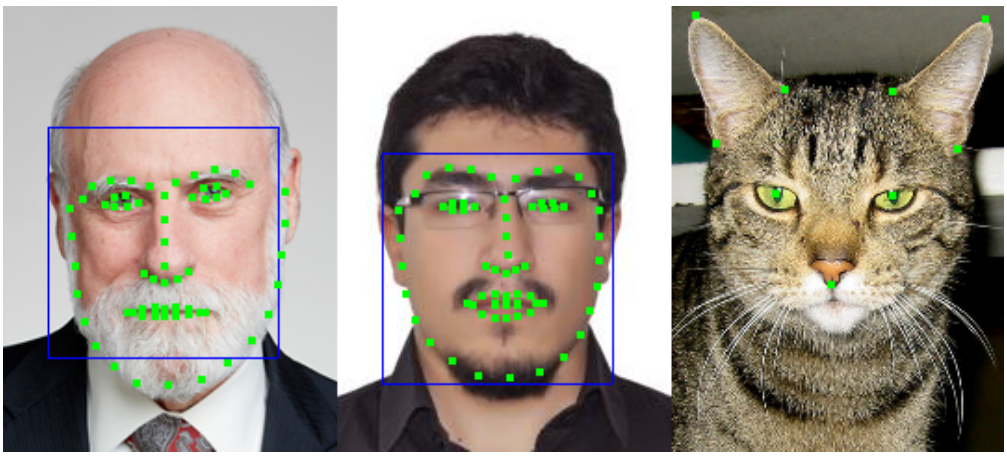

Figure 2: An example output image for Part 1

2 - Part 2: "Cat your life!" (10 pts.)

With the homework document, I also uploaded a numpy file named **"template points.npy"** which contains an xy template map for a frontal face image. The points and their ID correspondences can be seen in Figure 3.

To use a cat image in further operations, we should adapt the cat's original landmarks to the scheme with 68 landmarks. If the cat is looking straight, it is an easy process.

To basically do this;

- Let left eye center be in the middle of points 36 and 39. Similarly, you can use points 42 and 45 for right eye center. Calculate the distance between eyes in the

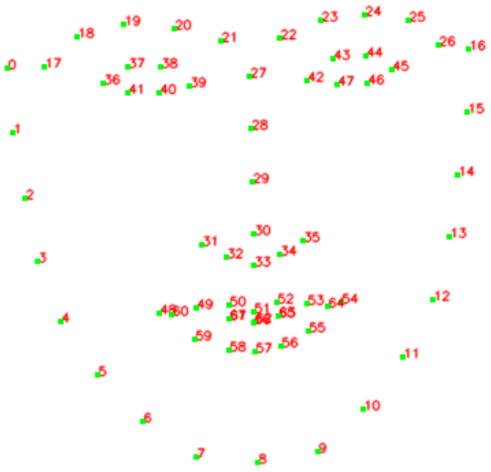

Figure 3: Facial landmarks and their ID correspondences

both images. Use ratio of these distances to scale the template points in horizontal direction.

- Find the point in the middle of two eyes in both the template and the cat image. For the template, let the mouth center be at the point 66. Use the ratio of distance between eyes and mouth to scale the template points in horizontal direction.
- After scaling operations, move the template on the cat's face using eyes and mouth of the cat as reference points. It will not fit perfectly but the result will be adequate for our implementation.

An example output for this part is shown in Figure 4.

3 - Part 3: Delaunay Triangulation (10 pts.)

In Delaunay Triangulation, triangles are created using points from a set. These triangles are formed so that, there are no intersection between any of them. Avoiding to delve deep into the theory of Delaunay Triangulation, we will benefit from the built-in functions of OpenCV for this task. You can start from the following skeleton code.

```
image = cv2.imread("image.png")
subdiv  = cv2.Subdiv2D((0,0,image.shape[0],image.shape[1]))
# Subdiv2D is an OpenCV object which performs Delaunay triangulation.

for i in range(68):
    subdiv.insert(...) #Each landmark point should be insterted into
    Subdiv2D object as a tuple.
```

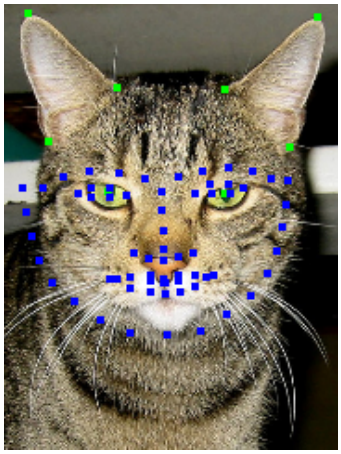

Figure 4: Coarsely applying the human facial landmarks on cat face

```
7
subdiv.insert((0,0))
9 subdiv.insert((0,image.shape[1]-1)) #Also to cover the whole image, 8
    points from the edges should be inserted. Show your work here.

11 triangles = subdiv.getTriangleList()
# Using getTriangleList function we can obtain the full list of triangles.
13
for i in range(len(triangles)):
15     sel_triangle = triangles[i].astype(np.int)
      #You can use cv2.line function to draw the triangles on the image. Show
       your work here.
```

The results for the example images are given in Figure 5.

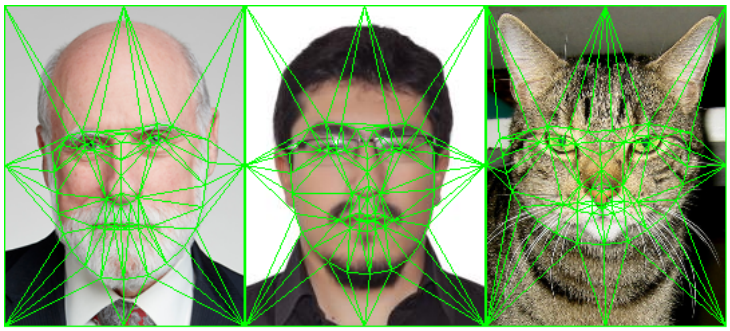

Figure 5: Delaunay triangles for the found landmarks.

4 - Part 4: Triangle Animation (20 pts.)

In this part of the homework we will do a simple animation which will be useful to understand the operations in the last part.

- Create an empty image to work on.
- Define two different triangles: source and target.
- Draw the source triangle on the blank image using the following lines.

```
cv2.polylines(image, [source_points], isClosed=True, color=(0, 0, 255)
    , thickness=1)
cv2.fillPoly(image, [source_points], color=(0, 0, 255))
```

- Find the transformation matrix which transforms the source triangle into the target triangle.
- Apply this transformation to the source triangle in 20 steps and keep each step's resulting image. (**Hint:** Remember that this is a linear operation.) Also change the color of the triangle so that at the last step it becomes pure blue.
- Using the *moviepy* library, save the animation.

Some example frames are given in Figure 6.

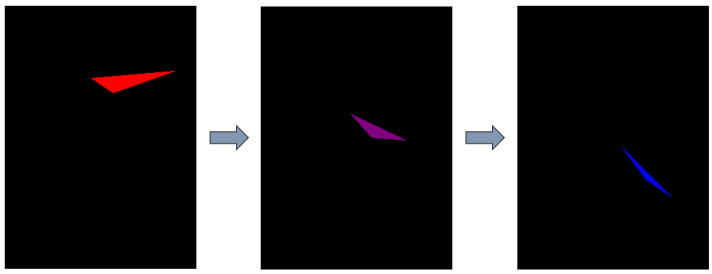

Figure 6: Some frames from the triangle animation.

5 - Part 5: Face Morphing (50 pts.)

All the parts above were nothing but some warm-up exercises for this last part. In this part, you are responsible to implement face morphing for all three permutations of images. The procedure is as given below.

- Obtain Delaunay triangles for all three images.

- To each triangle in the source image, assign a triangle in the target image. You can do this manually (Very time consuming but OK) or according to distances between triangles.
- The count of triangles will generally be different. If this kind of situation occurs you can do one of the following:
  - Divide some triangles into two in some underfilled regions.
  - Create fake and very small triangles which are filled with a constant color value at the start. They will not been seen initially since their edges are shorter than 1 pixel. You can select the initial color as the mean color value around the triangle.
- For each triangle couple, calculate the transformation matrices (Both from source to target and from target to source.). Split the operation into steps as you have done in the previous part.
- In each step, multiply the triangles with corresponding transformation matrices and then **blend** the coupled triangles.
- Display your resulting morphed images in between source and and target images for t=0, 0.5, and so on. You can define the number of steps according to the results. Using moviepy library, obtain the morphing videos.