Morph

February 19, 2020

[ ]:
```
import numpy
import matplotlib.pyplot as plt
from scipy.spatial import Delaunay
import glob as read
from scipy.spatial import tsearch
```

[ ]:
```
# this code is from assigenment 2 with few modification , its read the .asf␣
↪files and store the coords
def readFiles(typename):
    # decide how many files to read depend on gender
    if typename == "male":
        size = 33
        # defind arrays for x and y points . 58 points + 4 points for corners␣
↪and middle
        points = numpy.zeros((size,62,2))
    elif typename == "female":
        size = 7
        points = numpy.zeros((size,62,2))
    elif typename == "morphing":
        typename = "male"
        size = 6
        points = numpy.zeros((size, 62, 2))
    for filenumber in range(1, size + 1):
        filename = "files/" + typename + " (" + str(filenumber) + ").asf"
        f = open(filename, "r")
        lines = f.readlines()
        counter = 4
        addCorners(points,filenumber)

        # read specific lines to get the coords
        for i in range(16, 74):
            lines[i] = lines[i].replace('\n', '')
            lines[i] = lines[i].replace('\t', '')
            lines[i] = lines[i].replace(' ', '')
            x = int(float(lines[i][2:12]) * 640)
            y = int(float(lines[i][12:22]) * 480)
            points[filenumber - 1, counter, :] = x,y
```

```
            counter = counter + 1
            f.close()
    return points
```

[ ]:
```
#copy all the images on array
allThePicturesArray = numpy.zeros((480, 640, 3, 40))
allFiles = read.glob('files/images/*.jpg')
for each in range(len(allFiles)):
    allThePicturesArray[:, :, :, each] = plt.imread(allFiles[each]) / 255.
```

[ ]:
```
# a fundtion that find the average of an array
def averageShape(array):
    averageShape = numpy.average(array, axis=0)
    averageShape = averageShape.astype(int)
    return averageShape
```

[ ]:
```
# a function that add corners coords to the matrix
def addCorners(points, tempFileNumber):
    points[tempFileNumber - 1,0,:] = 0,0
    points[tempFileNumber - 1,1,:] = 640,0
    points[tempFileNumber - 1,2,:] = 0,480
    points[tempFileNumber - 1,3,:] = 640,480
    return points
```

[ ]:
```
#This function is from assignment 1 , it receive two triangles and calculate␣
 ↪the affine transformation
def calcAffineTransformation(tri1_pts, tri2_pts):
    A = numpy.float32([[tri1_pts[0, 0], tri1_pts[0, 1], 1.0, 0, 0, 0],
                       [tri1_pts[1, 0], tri1_pts[1, 1], 1.0, 0, 0, 0],
                       [tri1_pts[2, 0], tri1_pts[2, 1], 1.0, 0, 0, 0],
                       [0, 0, 0, tri1_pts[0, 0], tri1_pts[0, 1], 1.0]
                          , [0, 0, 0, tri1_pts[1, 0], tri1_pts[1, 1], 1.0],
                       [0, 0, 0, tri1_pts[2, 0], tri1_pts[2, 1], 1.0]])
    X = numpy.float32([tri2_pts[0, 0], tri2_pts[1, 0], tri2_pts[2, 0],␣
 ↪tri2_pts[0, 1], tri2_pts[1, 1], tri2_pts[2, 1]])

    transformMatrix = numpy.float32(numpy.linalg.inv(A).dot(X))
    transformMatrixArranged = numpy.float32([[transformMatrix[0],␣
 ↪transformMatrix[1], transformMatrix[2]],
                                             [transformMatrix[3],␣
 ↪transformMatrix[4], transformMatrix[5]],
                                             [0, 0, 1]])
    invsOfTransMatrix = numpy.array(numpy.linalg.inv(transformMatrixArranged))
    return invsOfTransMatrix
```

[ ]:
```
# this function receive a picture and its points and the average shape with the␣
 ↪edges of the average shape
# it extract the triangles of the average shape and the source image
# then it's calculate the affine transformation for each
#it calls the inverse wraping
def midface(source, average_Shape, edges, imagePoints):
    resultImg = numpy.zeros((480, 640, 3))
    listOfTransformationMatrix = numpy.zeros((3, 3, edges.shape[0]))
    for eachTri in range(0, edges.shape[0]):
        image_triangles = numpy.array([imagePoints[edges[eachTri][0]],␣
 ↪imagePoints[edges[eachTri][1]],
                                       imagePoints[edges[eachTri][2]]])
        avergeShape_triangles = numpy.array([average_Shape[edges[eachTri][0]],␣
 ↪average_Shape[edges[eachTri][1]],
                                             average_Shape[edges[eachTri][2]]])
        calculatedAffine = calcAffineTransformation(image_triangles,␣
 ↪avergeShape_triangles)
        listOfTransformationMatrix[:, :, eachTri] = calculatedAffine
    inversewarp(source, resultImg, listOfTransformationMatrix)
    return resultImg
```

[ ]:
```
# this code is from assignment 1 with a modification
# we add tsearch to find wich tri index
# we wrap the image
def inversewarp(source, result, totalTrans):
    for specH in range(0, 480):
        for specW in range(0, 640):
            triIndex = tsearch(trAvrShape,(specW,specH))
            x, y = findingCoords(specH, specW, totalTrans[:, :, triIndex])
            if insideTheLimits(x, y, 480, 640):
                result[specH, specW, 0] = source[int(x), int(y), 0]
                result[specH, specW, 1] = source[int(x), int(y), 1]
                result[specH, specW, 2] = source[int(x), int(y), 2]
    return result
```

[ ]:
```
#these parts of code are from assignment 1
def findingCoords(u, v, invMatrixCopy):
    x = v * invMatrixCopy.item(3) + u * invMatrixCopy.item(4) + invMatrixCopy.
 ↪item(5)
    y = v * invMatrixCopy.item(0) + u * invMatrixCopy.item(1) + invMatrixCopy.
 ↪item(2)
    return x, y


def insideTheLimits(cor1, cor2, nRows, nCols):
    return (cor1 >= 0 and cor1 < nRows and cor2 >= 0 and cor2 < nCols)
```

```
[ ]: #this code is from assignment 2 , it calculates the new shape according to␣
     ↪alpha value
     def interpolation(firstPic,secPic,alpha):
         averageTwoShapes = numpy.zeros((62, 2))
         for i in range(0, 62):
             averageTwoShapes[i,:] = (1 - alpha) * firstPic[i, :] + alpha *␣
     ↪secPic[i, :]
         averageTwoShapes = averageTwoShapes.astype(int)
         return averageTwoShapes
```

1 Morphing

```
[ ]: #this is the morphing function it use wrapfrac to find the new shape
     # it create two pictures and wrap them
     # then we use the dissovle fraction for the color
     def morphing(firstImg, secImg, firstImgPoints, secImgPoints, edges, warpFrac,␣
     ↪dissolveFrac):
         avg_shape = ((1 - warpFrac ) * firstImgPoints) + (warpFrac*(secImgPoints))
         resultFirstImg = midface(firstImg,avg_shape, edges,firstImgPoints)
         resultSecImg = midface(secImg,avg_shape, edges,secImgPoints)
         finalResult = ((1 - dissolveFrac) * resultFirstImg) +␣
     ↪(dissolveFrac*(resultSecImg))
         return finalResult

     #here we defind the number of frames , then we loop the number of frames␣
     ↪finding the warpFrac and dissolve frac
     frames = 45
     for i in range(0,frames+1):
         warpFrac = (1./frames)*i
         dissolveFrac = (1./frames)*i
         # morphes the 5th and 6th faces on the dataset
         morphed_im = morphing(allThePicturesArray[:, :, :, 4], allThePicturesArray[:
     ↪, :, :, 5], totalPoints[4,:,:], totalPoints[5,:,:], edgesAverageShape,␣
     ↪warpFrac, dissolveFrac)
         plt.figure(figsize=(8, 6))
         plt.imshow(morphed_im)
         plt.gca().axes.get_yaxis().set_visible(False)
         plt.gca().axes.get_xaxis().set_visible(False)
         #save each image
         plt.savefig('morphed/' + str(i) + '.png')
         print("frame saved" + str(i))
```

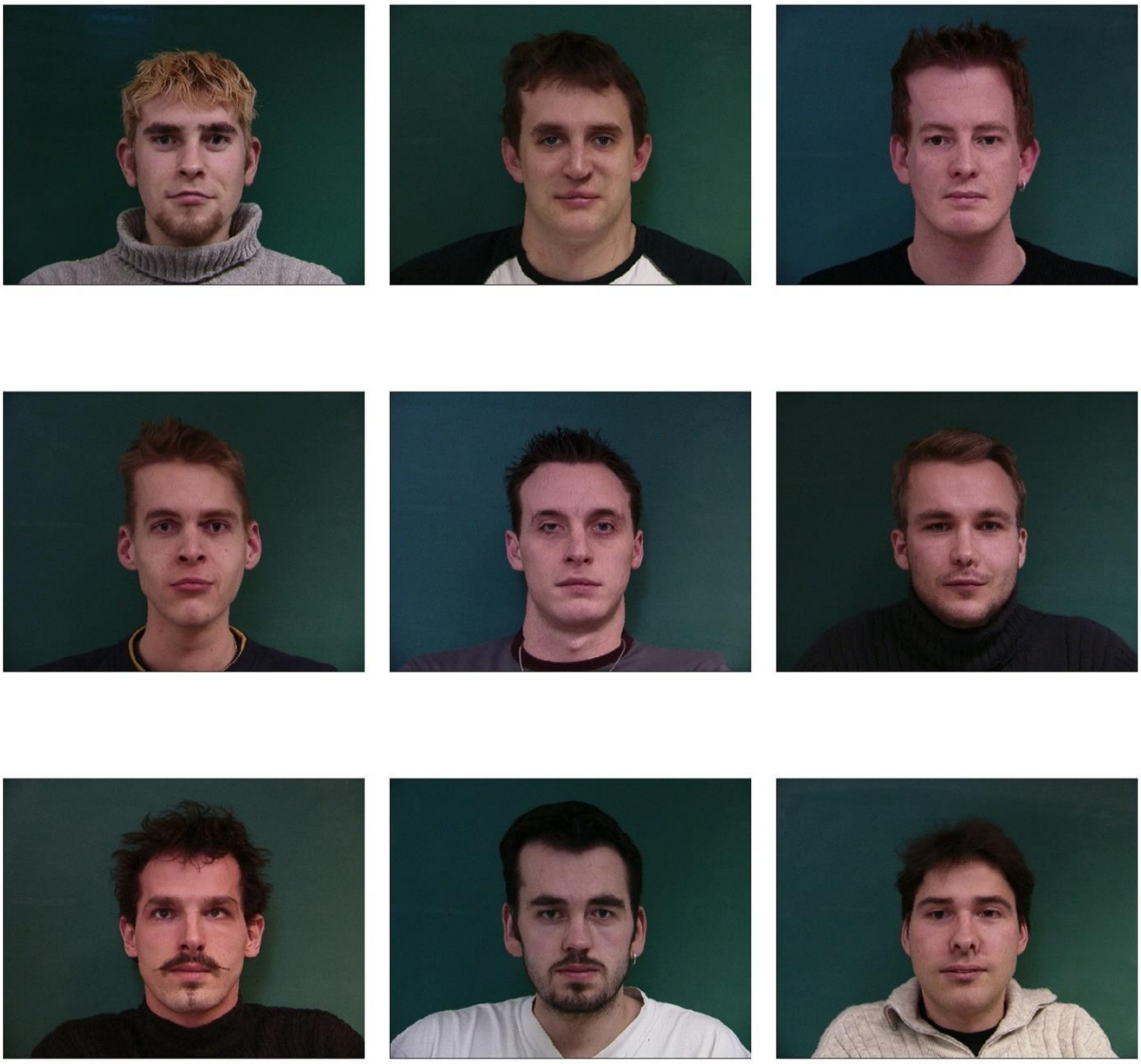
2 mean of a population of faces

[ ]:
```
#read the male population
totalPoints= readFiles("male")
#find the average male shape
avrShape = averageShape(totalPoints)
#trianglation
trAvrShape = Delaunay(avrShape)
edgesAverageShape = trAvrShape.simplices.copy()
#two matrix one for the result and another to store each morphed face
result = numpy.zeros((480, 640, 3))
total = numpy.zeros((480,640,3,33))
#loop 33 time (number of facese) calcuationg the mid face
for i in range (0, totalPoints.shape[0]):
    total[:,:,:,i] = midface(allThePicturesArray[:, :, :, i], avrShape,␣
 ↪edgesAverageShape, totalPoints[i,:,:])
# add a face to the result
    result = result + total[:, :, :, i]
    #find the average face by dividing the result by the number of facese
result = result / 33


#show and save the result
plt.figure(figsize=(8, 6))
plt.imshow(result)
plt.gca().axes.get_yaxis().set_visible(False)
plt.gca().axes.get_xaxis().set_visible(False)
plt.savefig('results/meanface.png')
print("total  saved")
```

I morphed all the males' faces into the average face

Here are some examples:



Then I calculated the mean male face, and the result is:

Morphing:

I choose these two faces for morphing.


First I computed the midface

Then I morphed the two faces using 45 frames, and here are some frames: