Image Morphing

Digital Image Processing



**Divjot Singh Rawla**

20.02.18

Assignment 2

# INTRODUCTION

The program implements morphing of two images given the tie points using delaunay triangulation using OpenCV with C++.

# HOW TO RUN

1. Open terminal and type the following command to compile the code :

*“g++ a2\_Divjot\_2016CSB1039.cpp -o output `pkg-config --cflags --libs opencv`”*

2. A file named “output” will be created in the same directory. Run the program on terminal.

3. Choose from the alternatives available to see two images getting morphed or a morphing of an image with it’s affine transformation.

4. The implementation requires the size of both the images to be the same. The case is also true for the affine transformation, otherwise the results are not very good.

5. You also need to provide the tie points for the two images in a separate file and mention the number of tie points in the program when asked. The instructions will be asked for while running the program as and when required.

# OUTPUT

The output of the program are image files stored on your system at the place you specify. The name of the files are in alphabetical order using two alphabets of the english language, i.e. the first image produced will have the name photoaa.jpg, followed by photoab.jpg, and so on.

# ALGORITHMS USED

1. **Detection of triangle under which a point falls:** It may be the case that the different triangles need different transformation for which, it was necessary to find the triangle in which a point lies. This was done in the following way:  
   *A line( y = mx + c ) divides the plane into two parts, one where y > mx + c and the other where y < mx + c. Along with this, the fact that the centroid of a triangle is always inside a triangle was used. The position of any candidate point was tested using centroid of the triangle as the base.*
2. **Triangulation:** The inbuilt function of getTriangleList was used for achieving the delaunay triangles for the given tie points. The function is for the object of a Subdiv2 class which contains information about the surrounding area of the image and the tie points for triangulation. The function took as input a vector of Vec6f as input and stores the triangle points in this.  
     
   The method took the points (-1800,-1800), (1800,0) and (0,1800) as its base. This base was used by me to say that the rest of the changes are taking place with respect to these. And the triangle that was covering the whole image was removed.
3. **Calculation of the Transformation Matrices:** The matrices of transformation for all triangles were calculated by solving the matrix multiplication and applying the end result there.
4. **Modified Median Filter:** In the end, the result obtained had some black lines because of forward mapping implemented by me. The last image obtained doing this was :-  
     
      
     
     
     
   Although the image is correctly triangled and is correct in terms of the morphing, it has got these lines which look bad. These lines are present because I did forward mapping. I could have done backward mapping but that would have been a tedious job. So, I decided to implement Median Filtering and this was not very helpful. The final image is below:-  
     
     
      
     
     
   This is better but the problem is still persistent so, I decided to use Median Filtering and using the newly obtained pixels for the median finding. That is,   
   Rather than creating a different image for finding median and updating, I used the same image that I updated for the finding of next pixels’ median. This gave very good results:-  
     
      
    

**Almost no inbuilt function of the OpenCV library was used except for Triangulation**

# MORPHING HILLARY TO TRUMP:

The video file has been attached in the program submitted before.

# REFERENCES

I learnt about the methodology to create a morph from the weblink:  
 “*https://www.learnopencv.com/face-morph-using-opencv-cpp-python/”*