Assignment 2
COL 783: Digital Image Processing

Gonçalo Alves[2022VST9500]

Indian Institute of Technology Delhi
info@iitd.ac.in
https://home.iitd.ac.in

1 Introduction

This report is meant to inform about the implementation of image morphing, pyramids (Paper1, Paper2) and stylization

2 Image Morphing

In this assignment, we were tasked with implementing image morphing using triangle method. as discussed in the class.

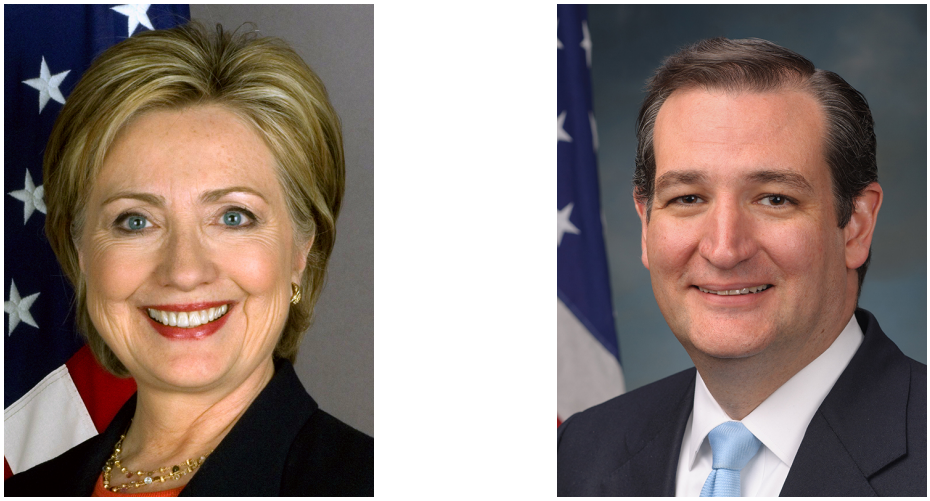

Fig. 1. Test images

The first task is to detect the faces, so that we can later get facial landmarks. For that, we used the in-built function of OpenCV, with a Cascade Classifier, and got the area of the face, for later use.

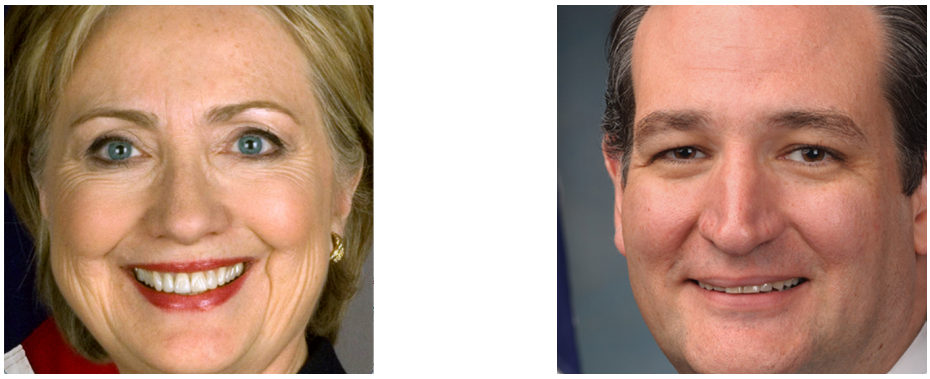

Fig. 2. Faces obtained

After, we retrieved 68 points from each face, **landmarks**, with the help of the **dlib** library. Additionally, we added 8 more points, corresponding to the corners of the image and the middle points between these corners, so that the morphing result was more accurate.

With these points, we can calculate the position of the points in the final image, by computing an average between the two sets. With these new points, we are then able to apply the Delaunay triangulation method. Additionally, there was a need to map these points with the help of a dictionary, in order to later use them fot the triangle warping.

Fig. 3. Results from triangulation

After, we warp each triangle and build the final warped image.

Fig. 4. Original images with result from triangulation

3 Image Pyramids

In this assignment, we were tasked with implementing image pyramid (Gaussian and Laplacian).

Fig. 5. Test images

The first task is to build the Gaussian pyramids of both images. For this, we first apply the following Gaussian matrix filter to the image:

$$G = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 1 & 4 \\ 4 & 16 & 24 & 4 & 16 \\ 6 & 24 & 36 & 6 & 24 \\ 4 & 16 & 24 & 4 & 16 \\ 1 & 4 & 6 & 1 & 4 \end{bmatrix}$$

And then, we downsample the image, by removing every even-numbered row and column. Next, we generate the Laplacian pyramids by upsizing the image to twice the original in each dimension, with the new even rows, performing a convolution with the same kernel shown above (multiplied by 4) to approximate the values of the "missing pixels" and then subtracting this new Gaussian with the corresponding one in our previously generated Gaussian pyramid.

We then add one halve of each image, at each level, and reconstruct the final image from our "joint" Laplacian pyramid and from again applying a Gaussian filter and upsampling the image.

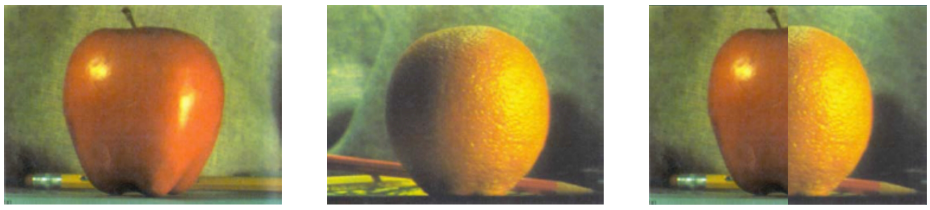

Fig. 6. Test images with simple halves addition result

Fig. 7. Test images with image pyramid result

4 Image Stylization

In this assignment, we were tasked with implementing the xDoG stylization contained in the paper, until section 2.5 (excluding).

Fig. 8. Original Image

For this, we simply calculated the difference of Gaussians of the image and then applied a thresholding function, as described in the paper:

$$D_{\sigma,k,\tau} = G_\sigma(x) - \tau \cdot G_{k,\sigma}(x) \tag{1}$$

$$T_{\epsilon,\varphi}(u) = \begin{cases} 1 & u \geq \epsilon \\ 1 + \tanh(\varphi \cdot (u - \epsilon)) & otherwise \end{cases} \tag{2}$$

Fig. 9. Test image with xDoG result. Parameters used: $\sigma = 0.5, k = 200, \tau = 0.98, \epsilon = 0.01, \varphi = 10$