# Image Processing: Homework 5

Due on February 1, 2020 at 10:00pm

**Yamin Sarcheshmehpour**
94109827

# Problem 1

**Poisson Image Blending Solution:**
suppose that the source image pixels are called $s$, the target image pixels are called $t$ and the result pixels are called $v$. In poisson blending we should find a $\hat{v}$ such that:

$$\hat{v} = argmin_v \sum_{i \in S, j \in S \cap N_i} [(v_j - v_i) - (s_i - s_j)]^2 + \sum_{i \in S, j \in S \cap N_i} [(v_i - t_j) - (s_i - s_j)]^2$$

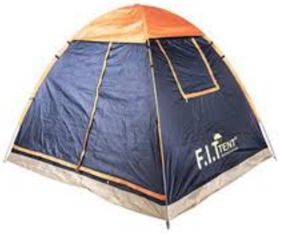
So for each $i$ we should have:

$$\sum_{j \in S \cap N_i} [(v_j - v_i) - (s_i - s_j)]^2 + \sum_{j \in S \cap N_i} [(v_i - t_j) - (s_i - s_j)]^2 =$$

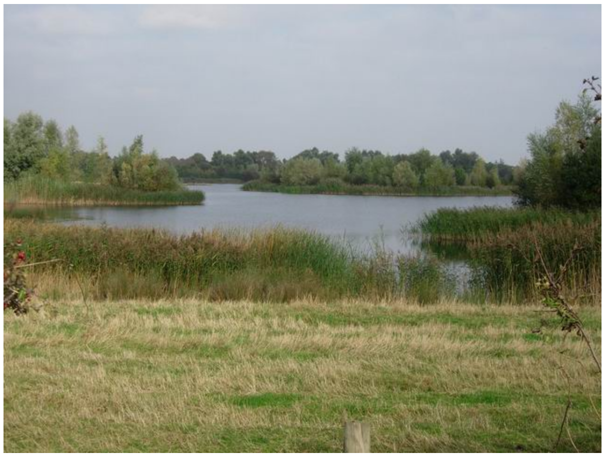
$$[(v_U - v_i) - (s_i - s_U)]^2 + [(v_i - t_U) - (s_i - s_U)]^2 + [(v_D - v_i) - (s_i - s_D)]^2 + [(v_i - t_D) - (s_i - s_D)]^2 +$$
$$[(v_R - v_i) - (s_i - s_R)]^2 + [(v_i - t_R) - (s_i - s_R)]^2 + [(v_L - v_i) - (s_i - s_L)]^2 + [(v_i - t_L) - (s_i - s_L)]^2$$

$$\rightarrow \frac{\partial f}{\partial v_i} = -2(v_U - v_i - s_i + s_j) + 2(v_i - t_U - s_i + s_j) - 2(v_D - v_i - s_i + s_j) + 2(v_i - t_D - s_i + s_j)$$
$$-2(v_R - v_i - s_i + s_j) + 2(v_i - t_R - s_i + s_j) - 2(v_L - v_i - s_i + s_j) + 2(v_i - t_L - s_i + s_j)$$
$$\rightarrow 4v_i - v_U - v_D - v_R - v_L = b$$
$$\rightarrow [4, -1, -1, -1][v_i, v_U, v_D, v_R, v_L] = b$$

So we can find $v$ by solving such $Ax = b$.
The *source* image is:

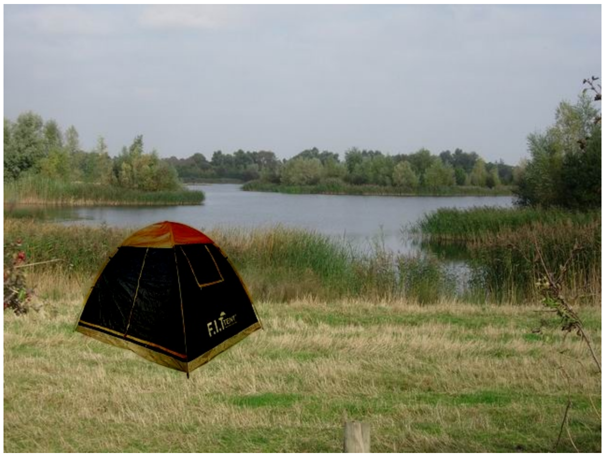


The *target* image is:

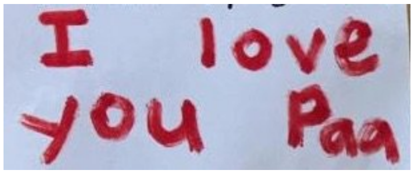


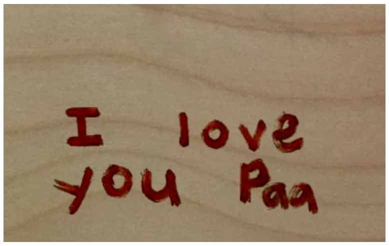
Problem 1 continued on next page. . .

And the *result* is:



Another example is as follows:





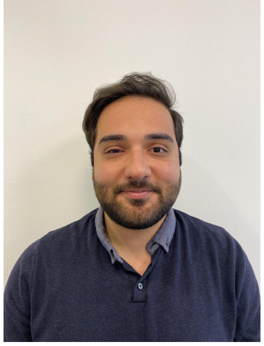
### Describe functions

1. *_reshape_img*: This function changes source image size the same as target image by considering offset.
2. *_get_mask_indices*: This function returns the mask pixels coordinates.
3. *_get_A_and_b*: This function calculates the $A$ and $b$ matrices for solving $Ax = b$ which has described above.
4. *_fix_target*: This function generates the result image by the result of $Ax = b$.

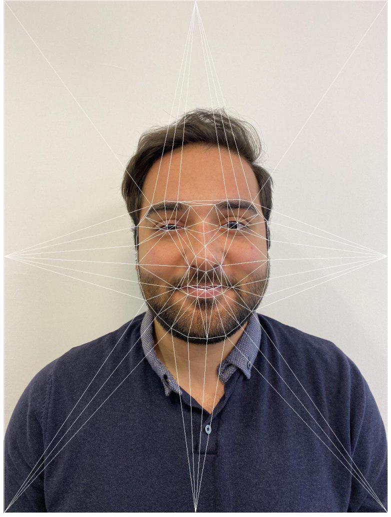
The code is in *Q1.py*, for runing the code please run *python Q1.py*.

# Problem 2

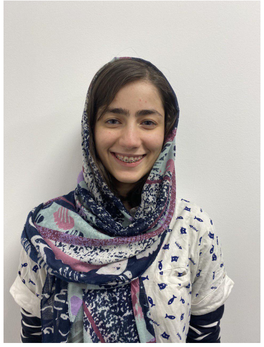
**Image Morphing Solution:**
For doing this we should convert the source image to the target image by dividing both images into triangles and match the corresponding triangles, then from the first step to the last one we try to change each triangle in the source image to its corresponding triangle in the target image.
The source image is:



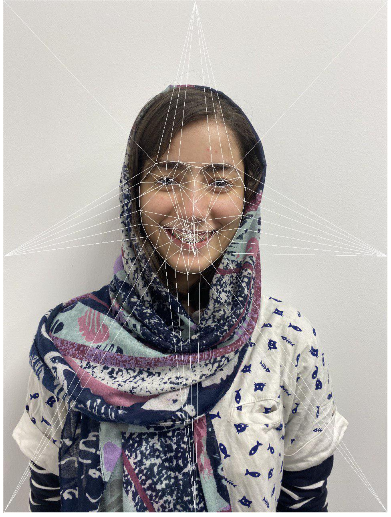
And its triangle divided image is:



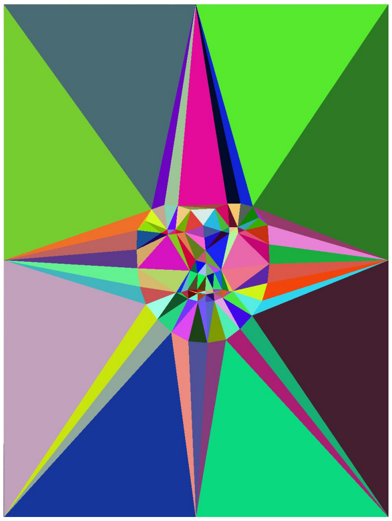
The target image is:

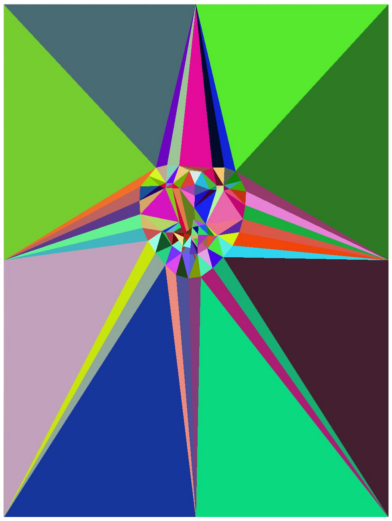


And its triangle divided image is:



For dividing images to triangles I have used *dlib* package and I have used *Delaunay* from *scipy* package. The matched triangles have the same color in the following images, the first one is for the source image and the second one is for the target one.





The final result has saved with the name of *morphing.mp4*.
**Describe functions**

1. *get_facial_points*: This function uses *dlib* and return facial key points of images.

2. *_get_cropped_triangle*: This function wraps a triangle of its input image to a given trangle.

3. *morph* This function wraps the triangle1 of the source image and its corresponding triangle of the target image to a given trangle.

Befor running the code please download *shape_predictor_68_face_landmarks.dat* from this link and then put it in the code directory.
The code is in *Q2.py*, for runing the code please run *python Q2.py*.