

Cmpe-537
Computer Vision
Assignment-2

Abdullah Atakan Guney
2018700069

November 4, 2018

Contents

1	Problem	3
2	Image Stitching Procedure	3
2.1	Select Common Points on Images	3
2.2	Homography Estimation	3
2.3	Image Warping	4
2.4	Image Blending	4
3	Experiments	4
3.1	Number of Points	4
3.2	Point Selection	6
3.2.1	Without Normalization	6
3.2.2	With Normalization	7
3.3	Noisy Points	9
3.3.1	Normalized	9
3.3.2	Unnormalized	10
3.4	Stitching All	10

1 Problem

In this assignment, I developed a image stitching system which can stitch multiple images in order to create a single panoramic image. Program implemented in Python 3.

2 Image Stitching Procedure

2.1 Select Common Points on Images

As it is told in the description, I have used matplotlib.pyplot.ginput to get corresponding points between images.

2.2 Homography Estimation

I calculated H matrix by calculating SVD of matrix A which is given below:

$$A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & u_1x_1 & u_1y_1 & u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & v_1x_1 & v_1y_1 & v_1 \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & u_Nx_N & u_Ny_N & u_N \\ 0 & 0 & 0 & x_N & y_N & 1 & v_Nx_N & v_Ny_N & v_N \end{bmatrix}$$

Matrix A holds the equality:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & u_1x_1 & u_1y_1 & u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & v_1x_1 & v_1y_1 & v_1 \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & u_Nx_N & u_Ny_N & u_N \\ 0 & 0 & 0 & x_N & y_N & 1 & v_Nx_N & v_Ny_N & v_N \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{N-1} \\ h_N \end{bmatrix} = \mathbf{0}$$

The estimation of h given by singular vector which corresponds to least singular value, so $h \approx V_N$.

I have used numpy's svd algorithm to calculate h . Given h ,

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Additionally, to calculate a robust transformation matrix, I have also normalized all input points before calculate H , such that average length of points is $\sqrt{2}$ and average of points is $(0, 0)$.

By given transformation matrix T_1 and T_2 such that

$$\begin{aligned} T_2x' &= H'T_1x \\ x' &= T_2^{-1}H'T_1x \\ H &= T_2^{-1}H'T_1 \end{aligned}$$

2.3 Image Warping

Image warping is changing the domain of the image function, rather than filtering. Let say we have transformation matrix H , which transforms domain of image-1 to the domain of image-2, then warping corresponds to finding $g(x')$. So,

$$g(x') = f(H(x))$$

where f is the input image of warping function and g is the output function. There are 2 major methods to calculate output image: Forward and backward transformation. I used backward transformation, which aims to find output image's pixels by transforming its index back to input image by using inverse transformation matrix H^{-1} .

$$H^{-1}x' = x$$

After finding all corresponding x , we can set $g(x)$. But result of inverse transformation can be placed between pixels on input image. So, I have used nearest-neighbour interpolation from "scipy" library.

In this assignment, first I have calculated borders for output image by forward transformation of corner points of input image. Then, I have calculated range for output image. By using H^{-1} , I have calculated all the corresponding points of the points in the range calculated, then I have assigned points of output image to corresponding input image pixel.

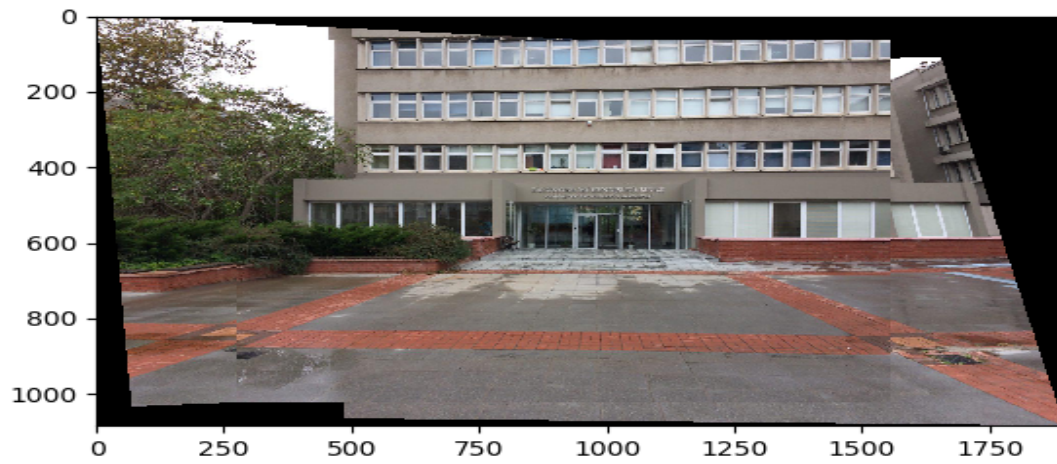
2.4 Image Blending

For blending all the images, first I have transformed the domain of all the images into middle one's domain. Then by warping all these images, I have calculated range for all of these. After all, I merged them into new big image. On the overlapping points, I have used a basic strategy that if overlapping pixel is zero in of the intersecting images, I have used the other one, and if both are non-zero, I used the average of them.

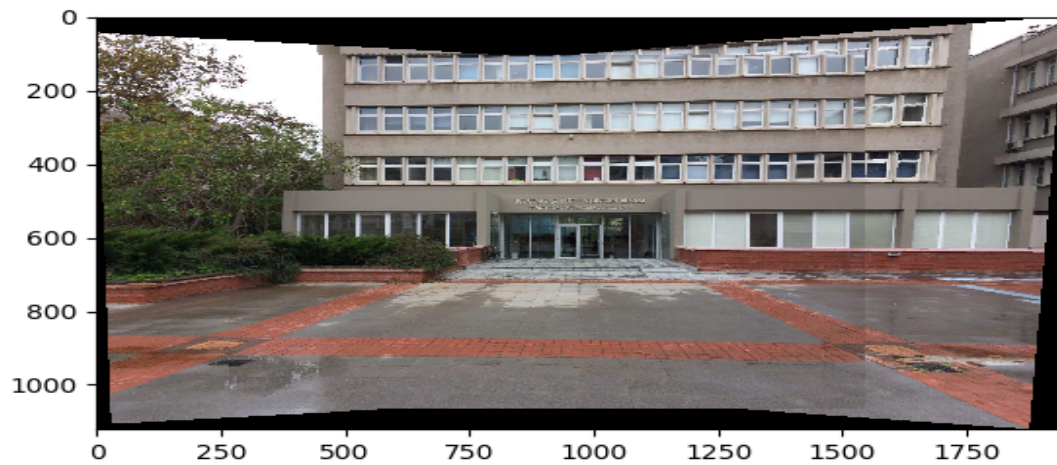
3 Experiments

3.1 Number of Points

I have experimented on both 5 and 12 corresponding points.



(a) Result for 5 correspondence points



(b) Result for 12 correspondence points

As we can observe that choosing more correspondence points results better. It is obvious that for above equation 5 points would be enough but more points are resulting better.

3.2 Point Selection

3.2.1 Without Normalization

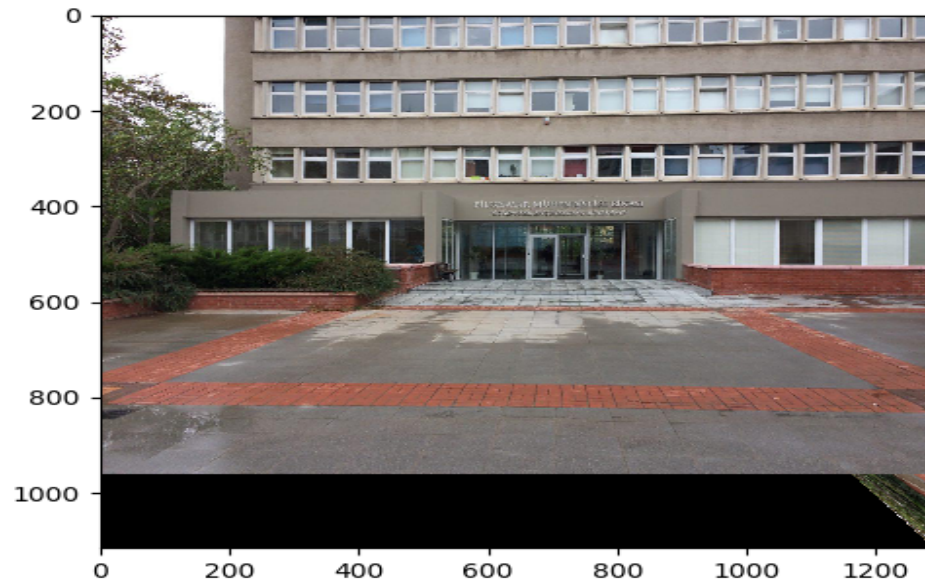


Figure 2: 3 wrong points are selected out of 12

3.2.2 With Normalization

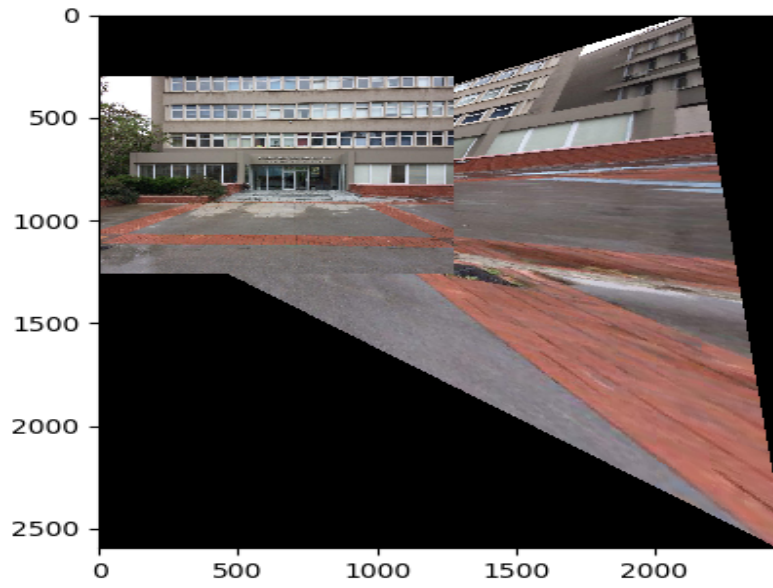


Figure 3: 3 wrong points are selected out of 12

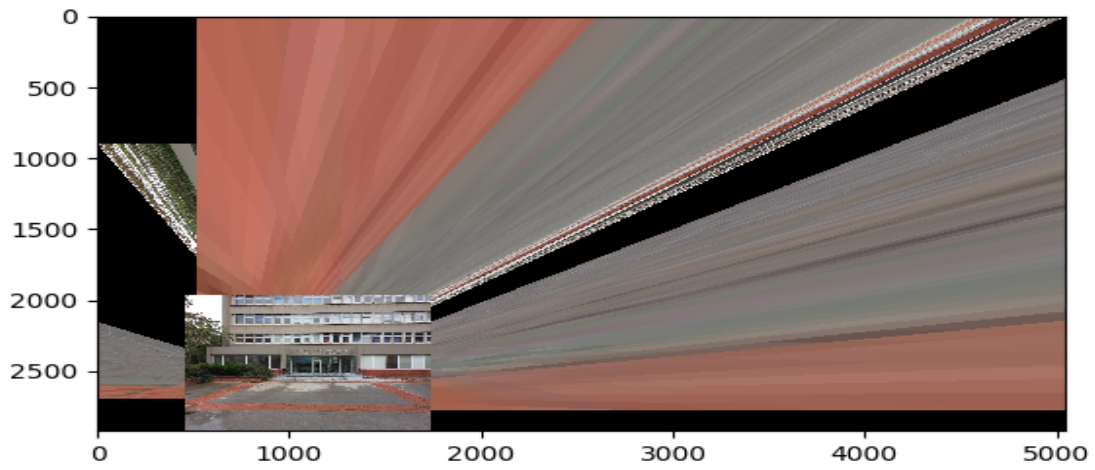
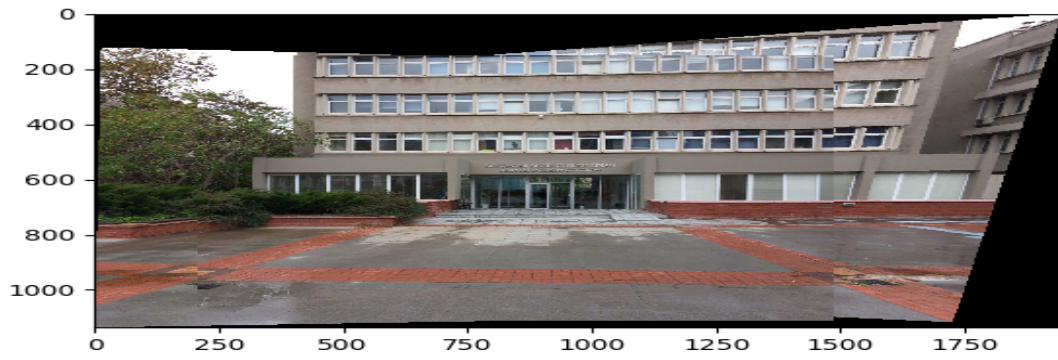


Figure 4: 5 wrong points are selected out of 12

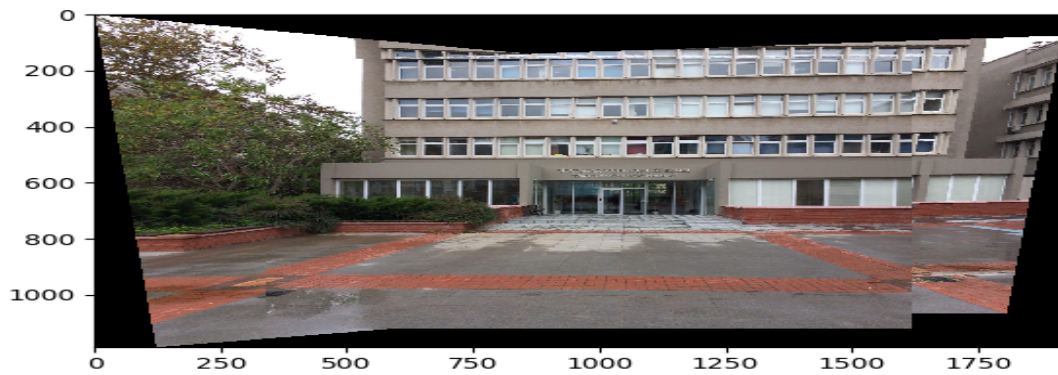
As we can see easier, choosing wrong points cause problem, and without normalization, transformation has almost no effect when we select points wrong.

3.3 Noisy Points

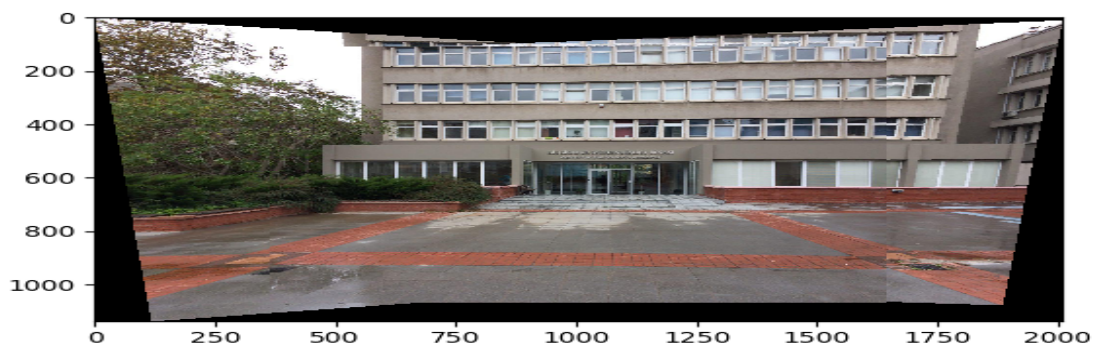
3.3.1 Normalized



(a) Gaussian Noise with variance 1



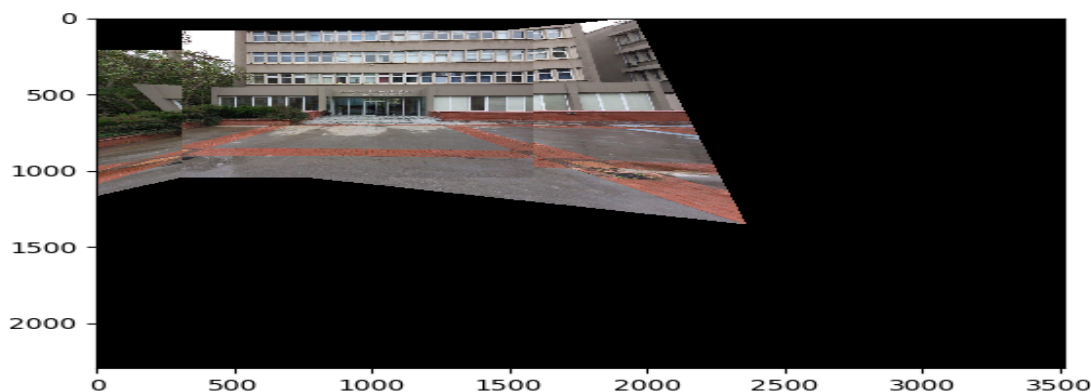
(b) Gaussian Noise with variance 5



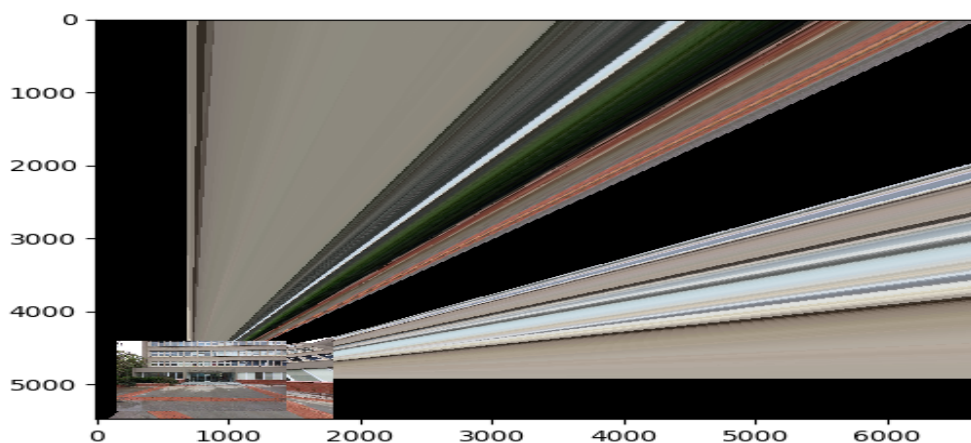
(c) Gaussian Noise with variance 10

So, as we can observe since normalizing is used, gaussian noise won't be so problematic.

3.3.2 Unnormalized



(a) Gaussian Noise with variance 1

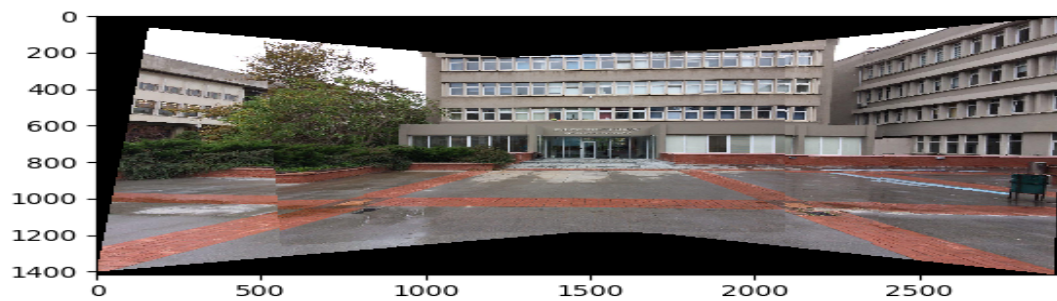


(b) Gaussian Noise with variance 5

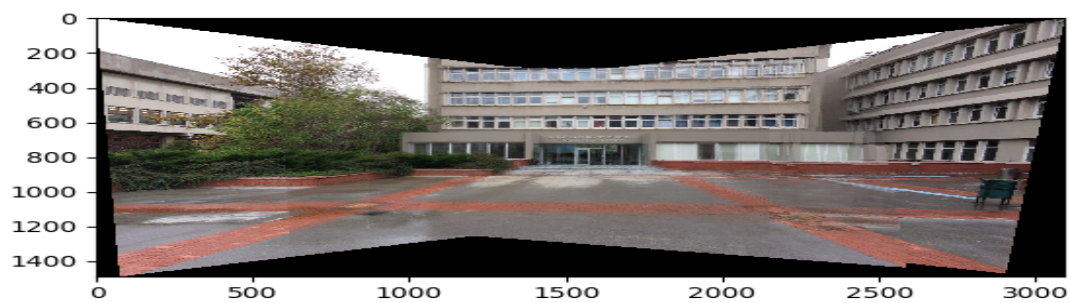
In unnormalized version, Gaussian noise would effect results even with little noise.

3.4 Stitching All

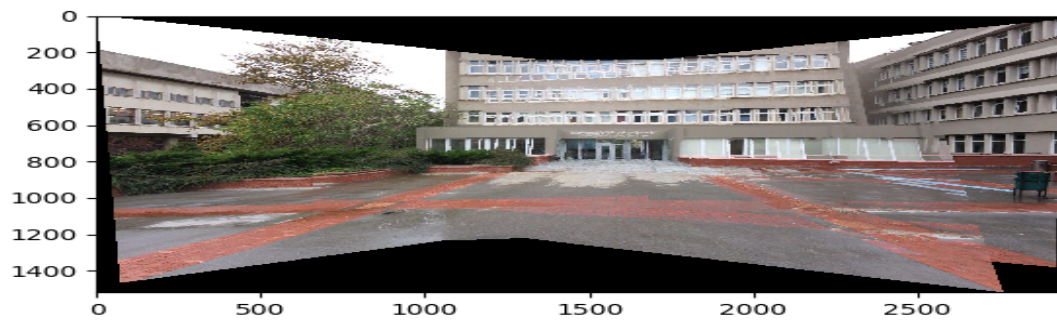
I have tried 3 different approach to handle overlapping pixels when blending all five images. In one of them I added images directly but in a natural way that gives priority to the middle image, in the second one, I averaged overlapped pixels, and lastly I set maximum intensity for each color channel. Results follows,



(a) All images blended directly top of each other



(b) Overlapping pixels are averaged



(c) Set it to max intensities