Cmpe537: Assignment 2 - Color Image Stitching

In this assignment we were asked to stitch the given images using homography between them in order to create a panoramic mosaic of the given images. It was a very challenging assignment for me and I couldn't complete the assignment even though I worked really hard for it.

Normalization:

In the normalization step, we needed to normalize the point in a way that their center would move to (0,0) and the distance would be sqrt(2) from the center. I implemented my own normalization algorithm with the help of this StackOverFlow question. I tried to use my normalization method when I am calculating homography between two images and I was able to calculate it well. However, when trying to apply this homography matrix to the input images, I saw that the results are not good. I think the reason for this mistake is that I am not normalizing the pictures when I am applying homography matrix that I got from normalized points. And normalizing the picture was too hard so I didn't use normalization during my calculations.

Homography Matrix:

To warp images, we needed a homography matrix. I calculated the homography matrix the way that we saw at the class. However, I needed a little help for solving the matrix equations. After my research, I learned that we can not solve the matrix equation with the inverse of the matrix because some matrices may not have inverses so we need to use singular value decomposition. For the implementation I got inspired from this question on Reddit. My homography matrix works mostly correctly.

Image Warping:

To warp the image, I at first, traverse the image to calculate the corresponding points with forward transformation. Then I do interpolation when I am traversing the black pixels to do backward transformation. This way there is no hole in the final image.

I wrote my own simple interpolation function. I take the nearest pixel value when the point is in between pixels when I am doing backward transformation. At first I tried to use the interpolation function of the scipy library, but I couldn't use it because my result needed to be an array with size 3 as they are the rgb values. The I tried to write a more complex interpolation function where I would get the average of the pixel values of the surrounding pixels but it didn't work so I decided to stick with the nearest pixel. I used this implementation on my implementation.

Blending Images:

After wapring the image, I blend the images by using the offset value array that I return from my warp function. I used matplotlib library's paste function with the masks that I created myself. I wrote a function for creating the mask function. My mask is basically an image with the increasing or decreasing transparency on horizontal axis. I used this article as resource when I am creating my mask.

I have two functions for stitching images:

stitch_left gets the left most image name and the image name at the right in this order as the parameters and stitches the left image to the right and saves the result as result.png in the same directory.

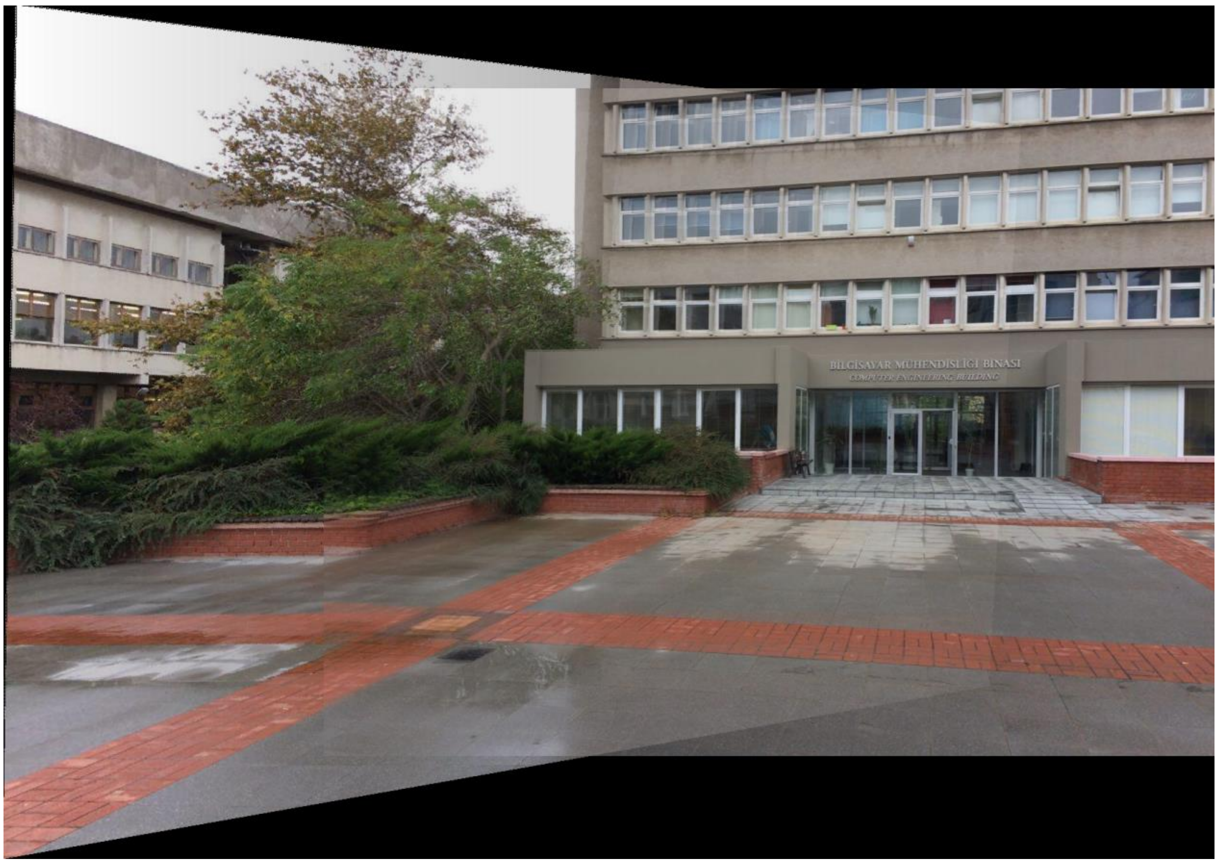
Stitch_right gets the right image name and left image name and stitches the right image to the left one and saves the result as result2.png in the same directory.

I use this two methods sequentially to stitch three images but it doesn’t always work greatly although they work great separately.

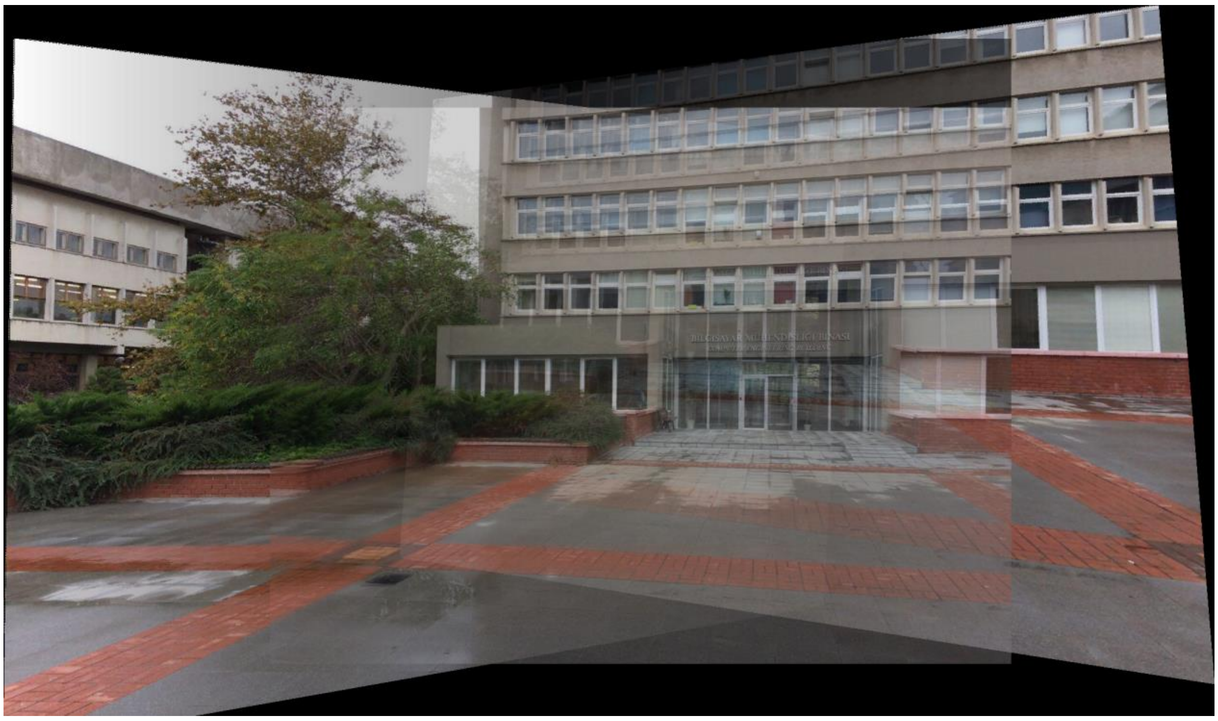
Experiments:

I couldn’t stitch 5 images but I have stitched three images and the algorithm work the same for 5 images. I don’t use normalization as it doesn’t work well.

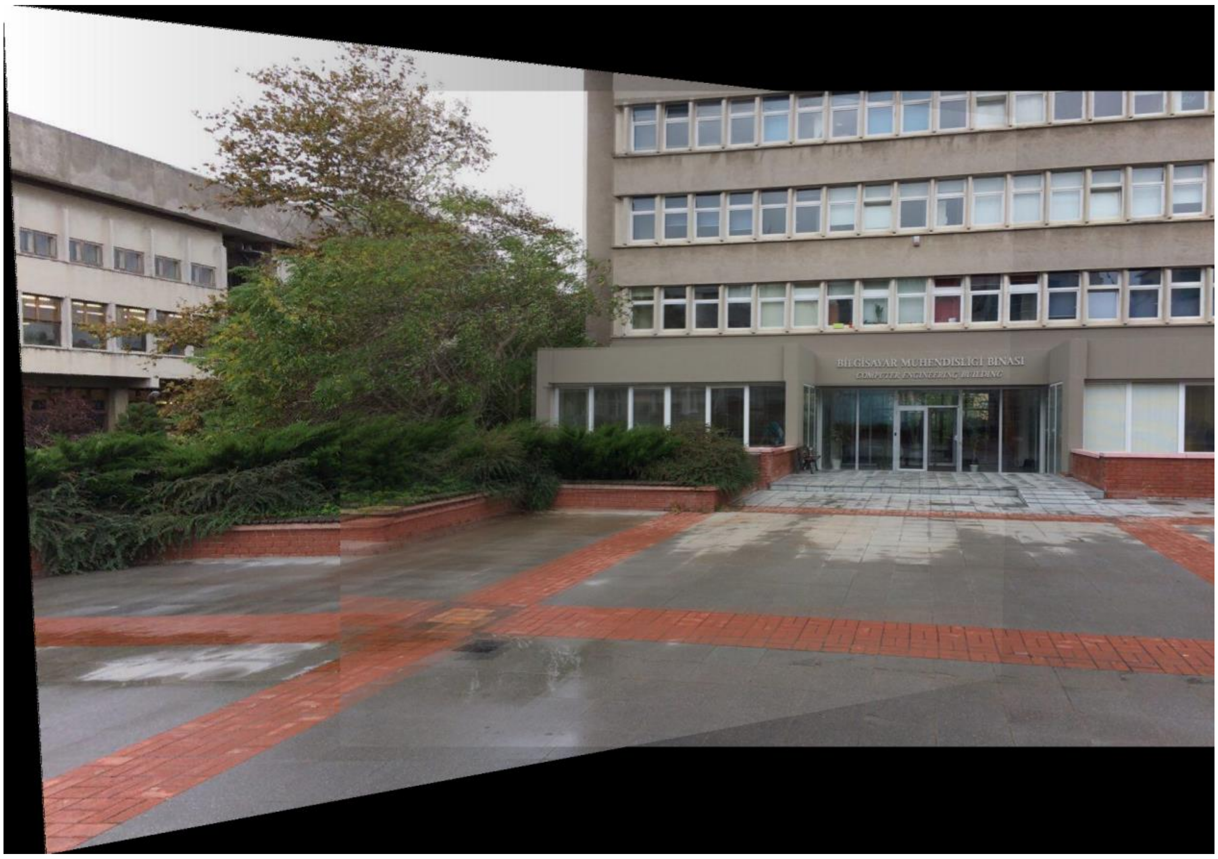
With 5 corresponding points(left-1 and left-2):



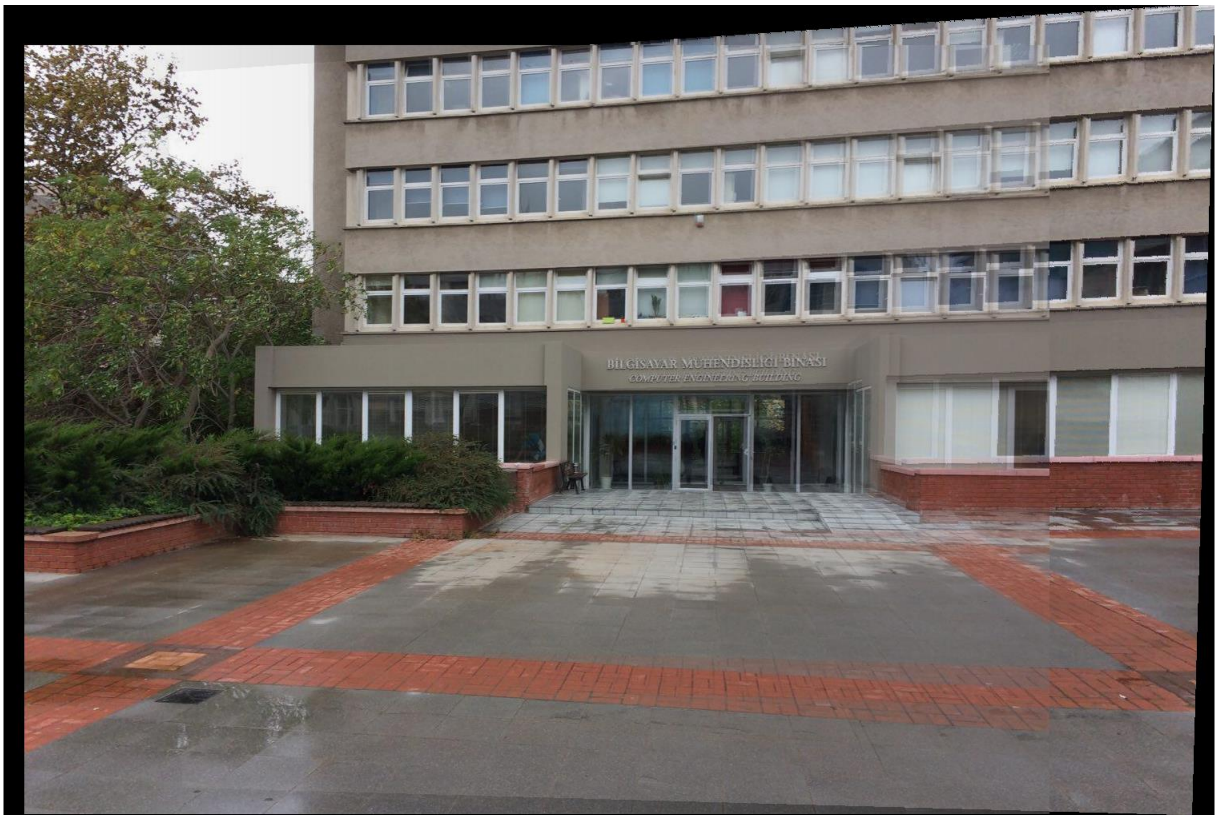
Stitching 3 images (left1-left2-middle) with 5 correct points:



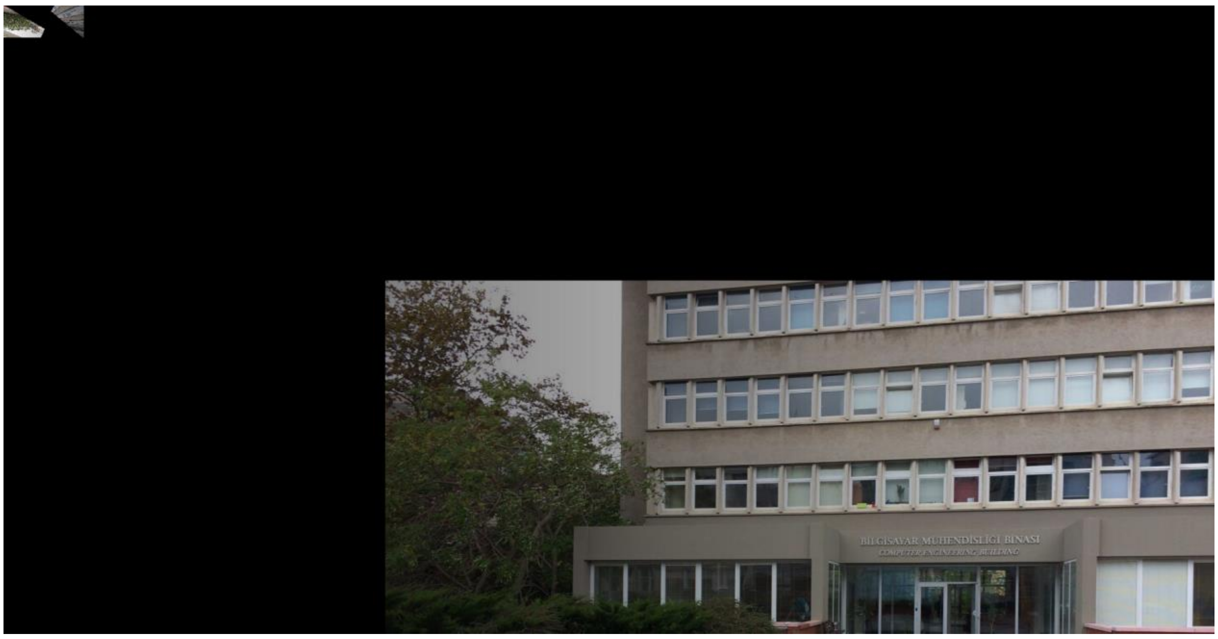
With 12 corresponding points (left-1 and left-2):



With 12 corresponding points (left-1 and middle):

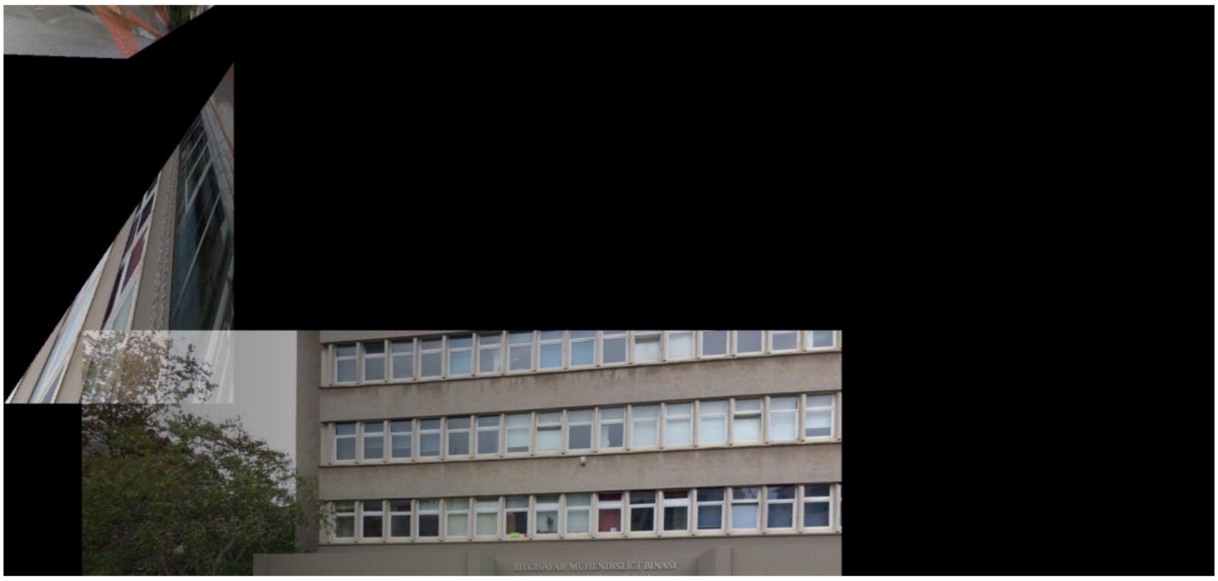


With 12 corresponding points with 3 wrong points (left-1 and left-2):

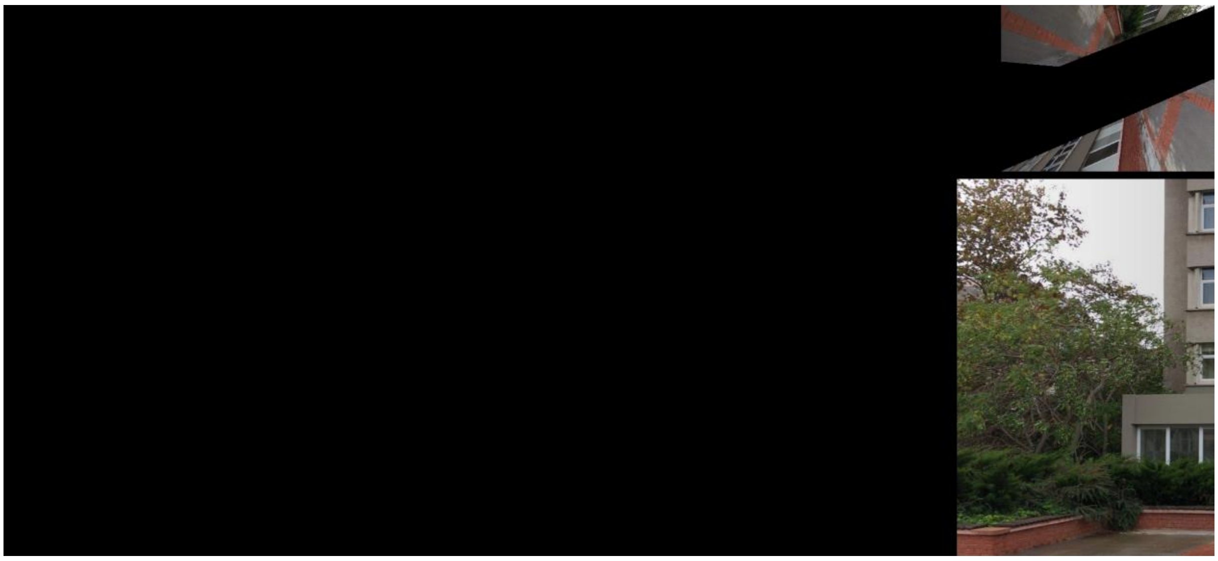


With 12 corresponding points with 3 wrong points (left-1 and middle): Gives error ☹

With 12 corresponding points with 5 wrong points (left-1 and left-2):

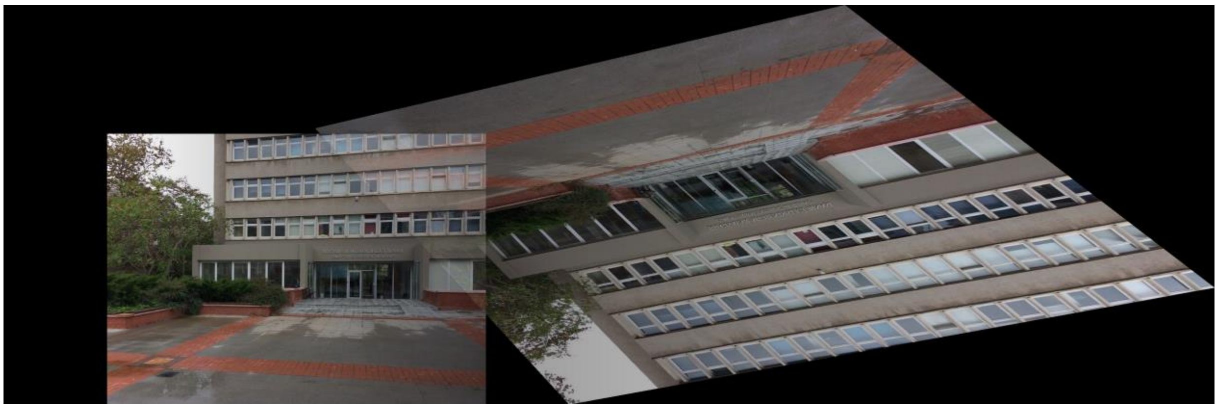


With 12 corresponding points with 5 wrong points (left-1 and middle):

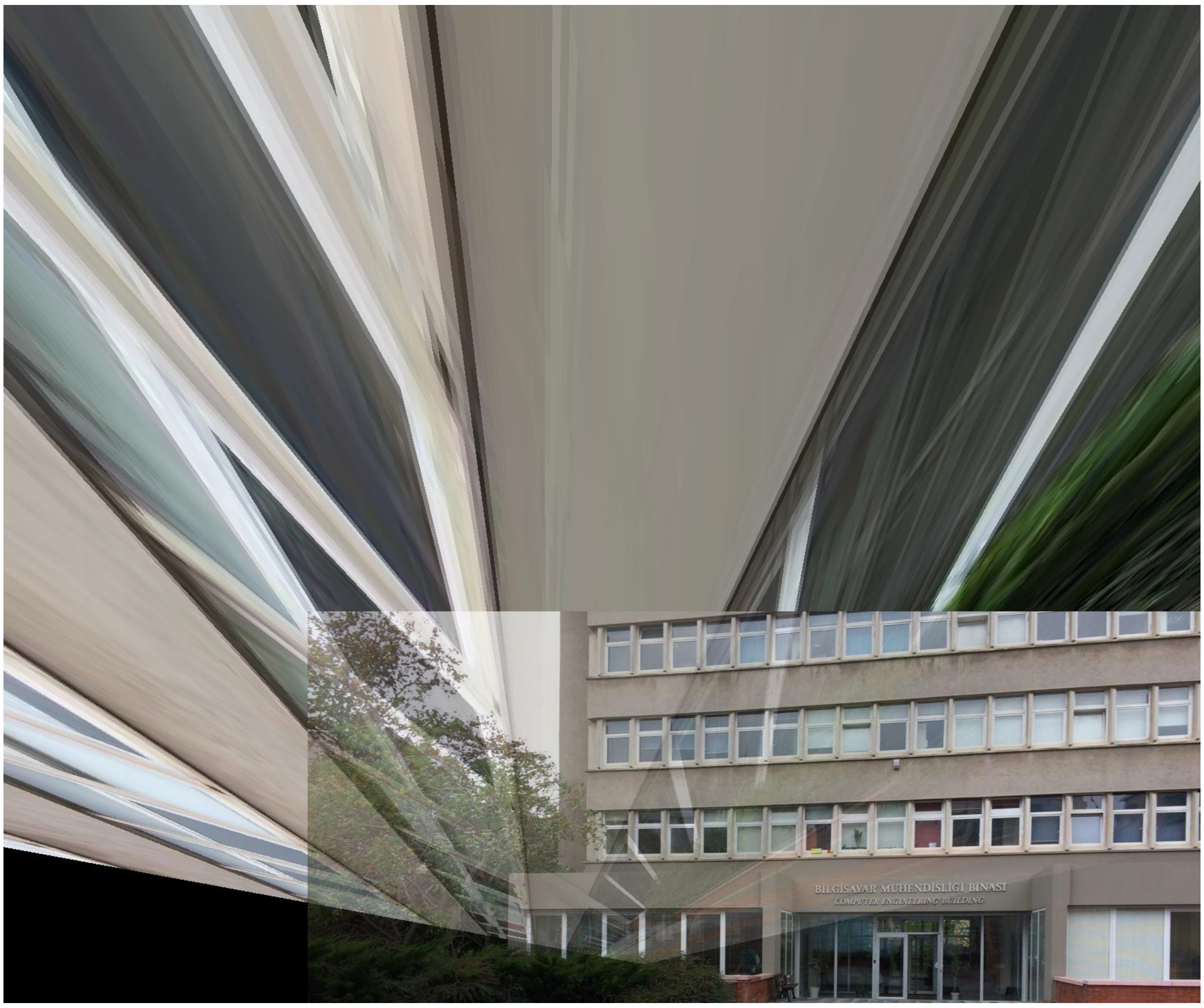


With 5 corresponding points with 3 wrong points (left-1 and left-2): Takes too long to run ☹

With 5 corresponding points with 3 wrong points (left-1 and middle):



With 5 corresponding points with 5 wrong points (left-1 and left-2):



With 5 corresponding points with 5 wrong points (left-1 and middle): Takes too long to run ☹

Conclusion:

I think this homework was harder than it should be. I tried my best to do it by spending more than 10 days on it and sleeping late for a few days but I couldn't go further. I believe I understood the theory of homography but the implementation was really hard and it was really hard to find resources. Even the lecture book suggested us to use cv2 library for this type of work where it was expected from us to write every function.