- Who did what in your project?

We both equally contributed to most aspects of this project. We would meet on zoom whenever we wanted to work on the project. After writing all of the code together, Vir took some pictures for our “above and beyond” aspect of piecing together more than two images. He was able to capture 4 images and piece them all together. We both tried to capture a group of pictures from our dorms to piece together, but they didn’t turn out all too well as they were hard to find good matching points in each image.

- Describe your process for developing the program and how you alternated between editing code and generating data.

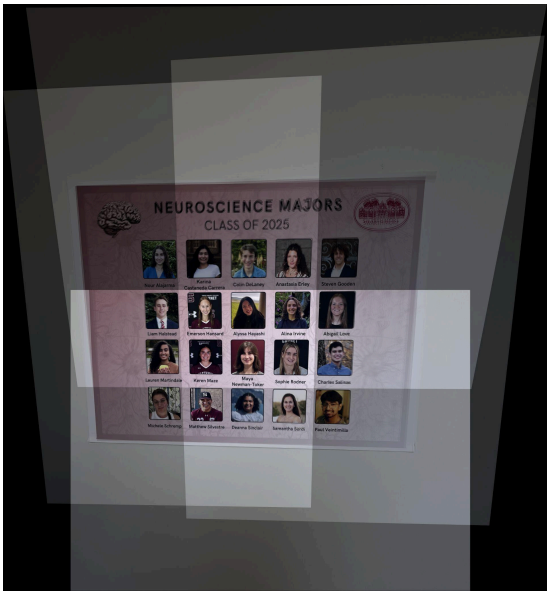
To write the code for the project, we closely emulated the steps outlined in the project instructions. We took what was instructed and turned it into a code version. This was relatively easy. The difficult part came when we tried to piece together a variable amount of images for our “above and beyond” aspect. Our overall procedure for both aspects of this project involved essentially writing what we thought would accurately stitch together the images, then we would debug any errors that occurred, and if the images did not stitch together properly we would try to identify why. For the base aspect of the project, we only needed a few iterations to get it to work solidly, then when adapting the code to go further, we would apply a similar procedure, but if the stitching wasn’t successful, we would test it on the “weird1” and “weird2” images to see if it was an image/point picking problem or a code problem. Since we knew the weird images worked, it served as a good standard to identify where the issues were occurring.

- What difficulties did you encounter along the way, and how did you overcome them?

We encountered two significant difficulties along the way. The first was a problem when we were trying to adapt our code to work for a variable amount of images. We were running into an issue with boundingRect where we weren’t passing in a parameter of the correct shape. We originally used vstack in the base phase of the project, but we didn’t think that would work for our adaptation, so we tried using the reshape function to fix it. That still failed as we were unable to successfully determine what shape was needed exactly and where we were going wrong. Vir had a good idea to just try vstack which ended up working to our surprise. The other issue we encountered was something we assumed had to be related to our point picking. The original images we used, we struggled to find good points to pick so when we ran the images through the stitcher program, the output was not ideal. It resulted in some weird warping. So, we decided to resolve this issue with a better set of images with more clear, distinct points and it resulted in a great output.

- Include images of your program outputs for at least two datasets in your PDF.

Output image, stitching 2 images together



Output image, stitching 4 images together

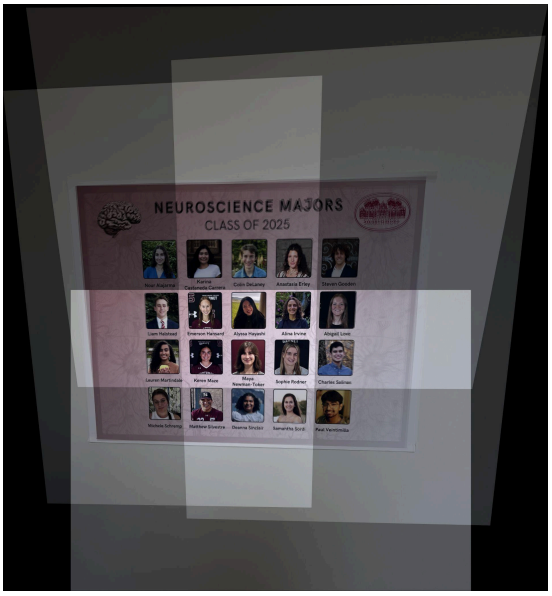
- Describe the work you did for the “going further” aspect of the project, including telling me how to use any additional programs you wrote. Include program output images in your writeup if appropriate.

For the "going further" aspect of this project, we extended the original stitcher.py program we had written to support stitching more than two images together. This way we could reconstruct objects from multiple partial views as long as they had some overlap.

Our implementation maintains much of the same logic we developed for two-image stitching, except we had to implement the algorithm inside a loop whose total iterations were dictated by the number of images input by the user. We chose to use the first image as a reference frame. All other images were warped to align with this reference. We tweaked the code so that for each additional image, our code computes a new homography that transforms it to the reference image's coordinate system. And to determine the dimensions of the final output, we transformed the corners of all images into the reference frame and computed a bounding box that contains all content. Finally, each image is warped according to its homography combined with a translation to ensure all content fits in the output.

The execution remains identical, the only difference is that you are free to provide any number of image files as command-line arguments:

eg - python stitcher.py image1.jpg image2.jpg image3.jpg



Output image, stitching 4 images together