Saatvik Billa CS180 Project 4A: Image Warping and Mosaicing

Part 1: Shooting the Pictures

I chose three sets of pictures, the first set being of my living room, the second set being of some tapestries in my room, and third set being of my desk.

While taking the photos, there were a couple of things to keep in mind, the main one being to keep the center of projection (COP) the same. This meant that when I was taking the second photo, the only thing I could do was move the phone to the right or left; I couldn't change the angle at which I was taking the photo. I'm well aware that image processing techniques exist where you can stitch two photos that are taken at different angles, but that was out of the scope of this project ;)

My Living Room

Picture #1

Picture #2

Car Tapestries and Posters


It's named 964.
But it feels like 69
all the way.
There is still only one car that looks, feels, and performs like a Porsche 911:
a Porsche 911. It is the one sportscar that manages to be both timeless
and ahead of its time. And we're now making it very affordable for you to
drive one. After all, we know how many decades you've waited.
PORSCHE
LEXUS LFA
Beginning production in December 2010, just 500 examples of the Lexus LFA were hand-built.
Powered by a 4.8 litre V10, making 552 horsepower at a screaming 8,700 rpm, the Lexus LFA is
one of the most iconic supercars in automotive history.

Picture #1

Picture #2

My Desk & Bed


San Francisco
SOUVENIR
"THE CITY BY THE GOLDEN GATE"
GOLDEN GATE BRIDGE
Chinatown
OAKLAND BAY BRIDGE
COIT TOWER
PALACE OF FINE ARTS
FISHERMAN'S WHARF
GOLDEN GATE PARK
ALCATRAZ
Mosaicing

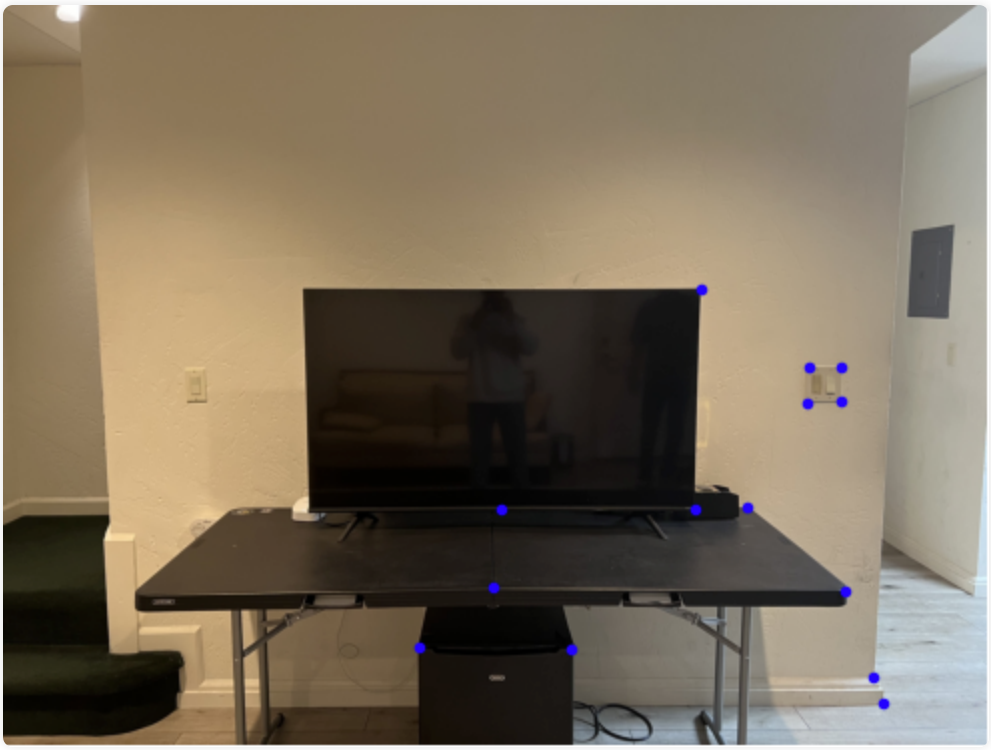
Picture #1

Picture #2

Part 2: Recover the Homographies

For each pair of images, I used the online tool provided by the course staff to pick out correspondence points pertainining to relevant features. Once I got the list of these points, I iterated through each pair of correspondences and formulated a linear equation relating the two points. The goal was to come up with a set of coefficients, known as the Homography matrix, that would provide a mapping between the points in the first image and those in the second. Since there are 8 coefficients we have to solve for (the 9th is assumed to be 1.0), we would need 8 equations to solve it normally, but since we have way more than 8 equations, the system becomes overdetermined, leading us to use least squares to find the best possible set of coefficients.

Living Room



Picture #1 with Reference Points

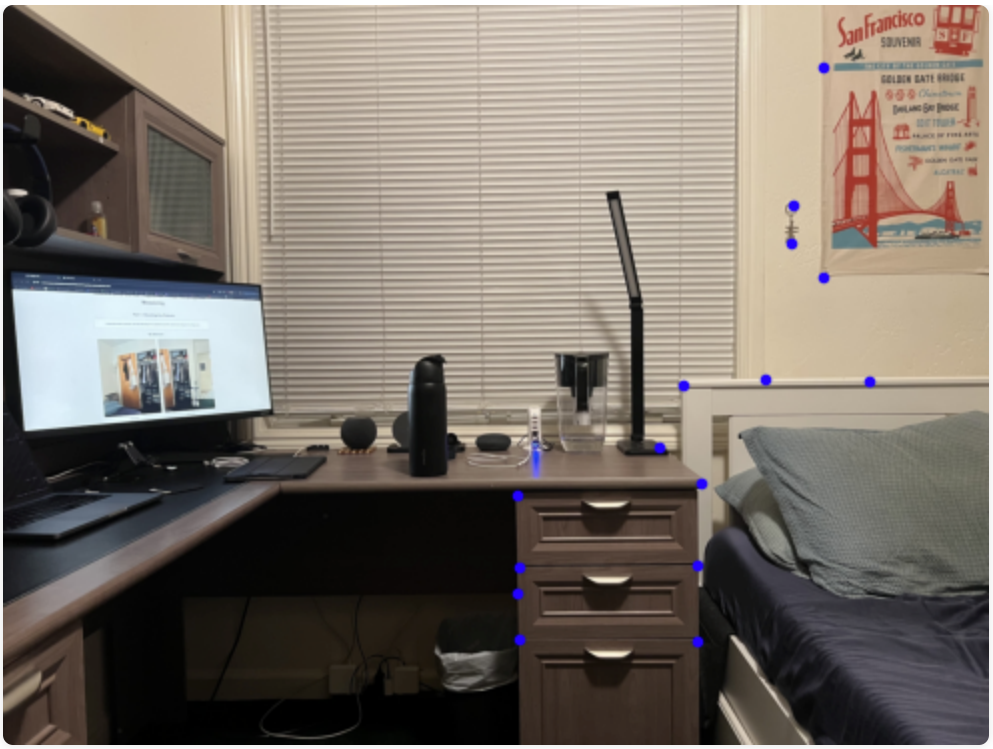
Picture #2 with Reference Points

Car Tapestries and Posters

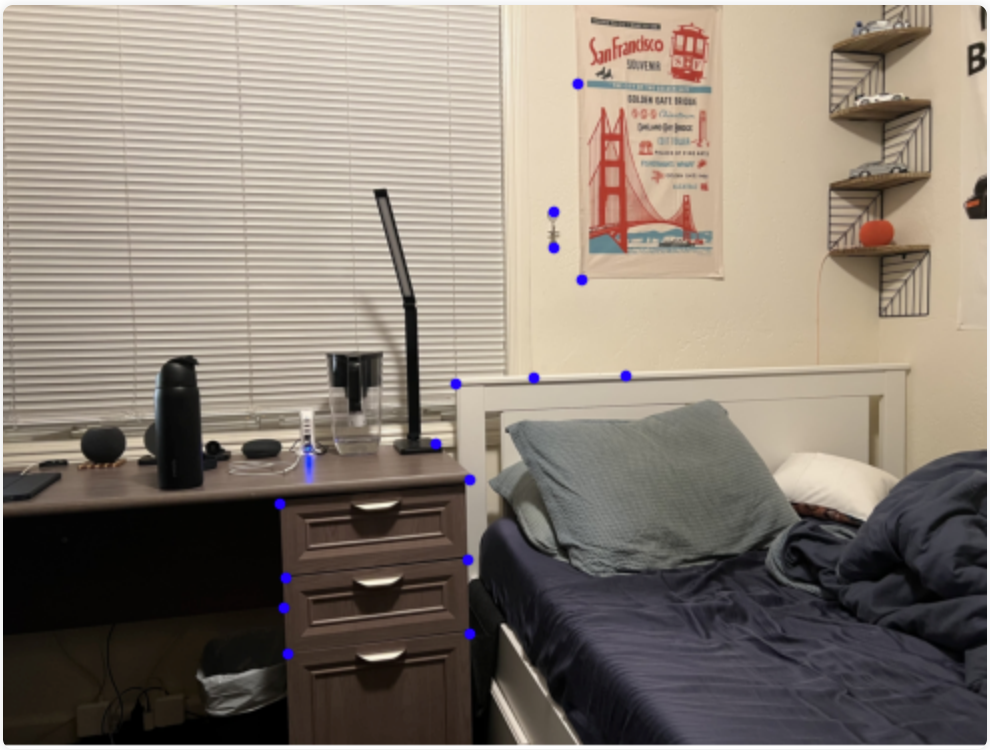
Picture #1 with Reference Points

Picture #2 with Reference Points

Desk & Bed



Picture #1 with Reference Points



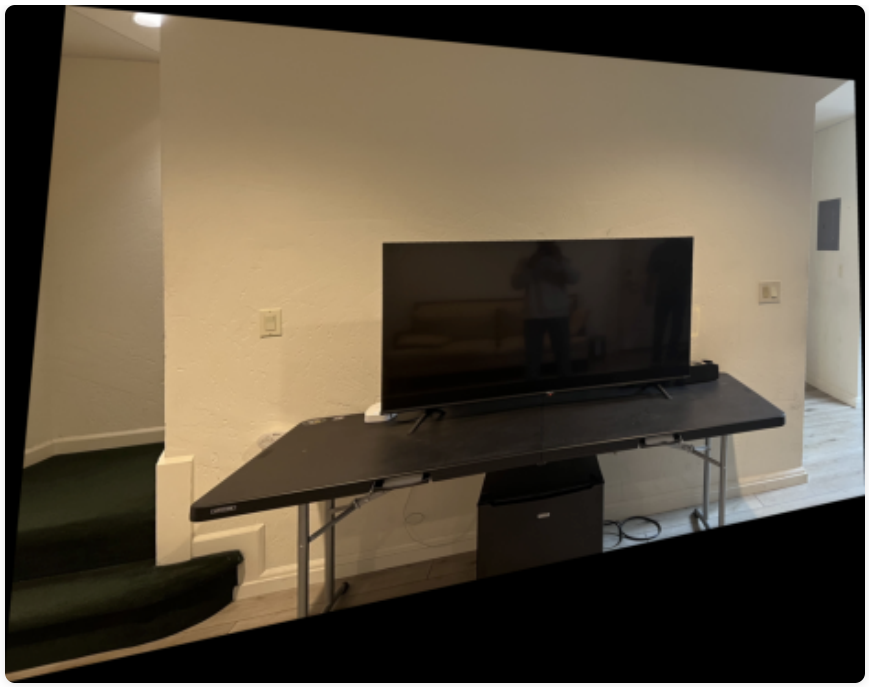
Picture #2 with Reference Points

Part 3: Warp the Images

This is where the mosaic process starts to take off. Now that I have the homography matrix that I computed in the previous part, I can map the entirety of the first image to the second image's space. I start by warping the corners of the image so that I can get an idea of the "bounding box" formed by the warped image. Then I iterate through all the points within this range, use the inverse of the homography matrix to get the pixel value in the original image, and if that point exists within the original image's range, I then set the pixel value of the new point to the pixel value I just retrieved. If the point doesn't exist, it remains black, which is why in the images below, you see some parts of the image are just dark.

Living Room

Original Picture #2



Picture #1 warped to Picture #2's Homography

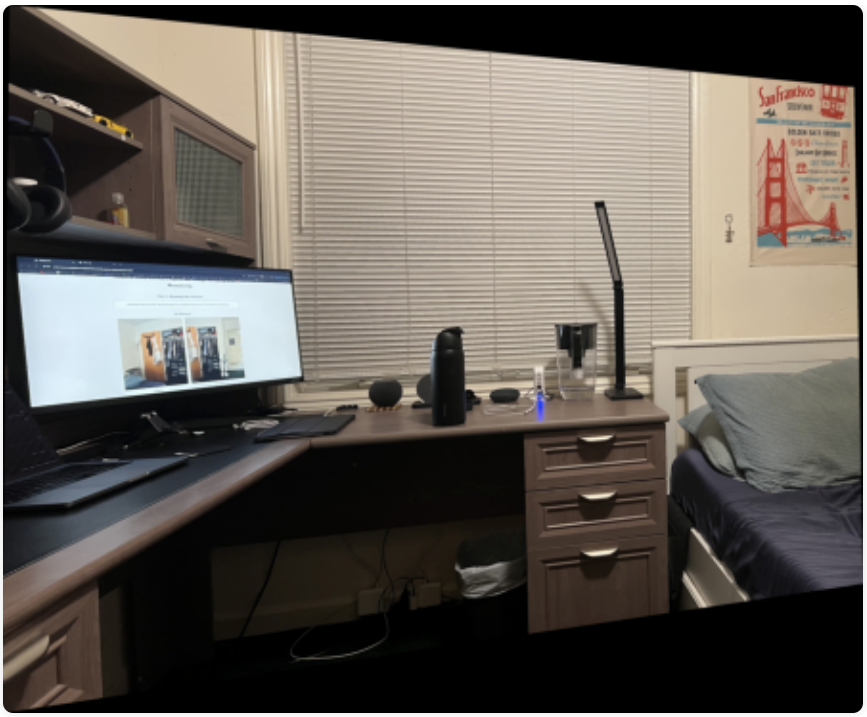
Car Tapestries and Posters

Original Picture #2

Picture #1 warped to Picture #2's Homography

Desk & Bed

Original Picture #2



Picture #1 warped to Picture #2's Homography

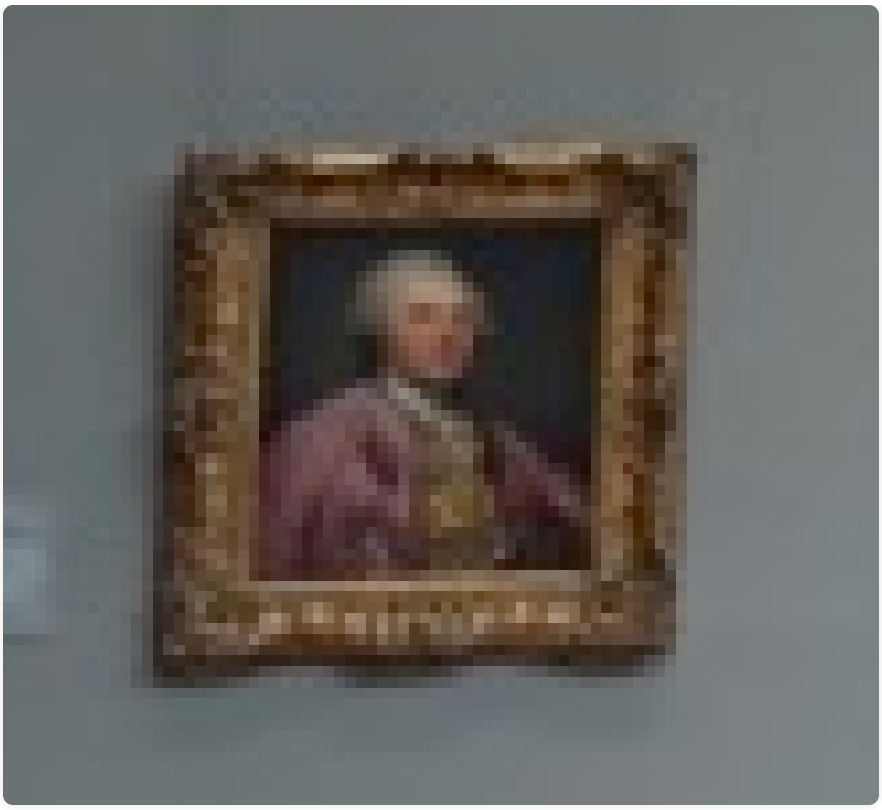
Part 4: Image Rectification

The purpose of this section was to make sure all the code I had written till this point was working properly, and so to test that, I took an image with multiple slanted objects and warped them to the homography of a polygon with parallel lines (i.e. square and/or rectangle). As correspondence points, I selected the four corners of the painting and I mapped them to a the following rectangle that I just arbitrarily defined: [0,0],[335,0],[0,200], [335,200].

Museum Painting

Original Picture

Rectified Picture #1



Rectified Picture #2

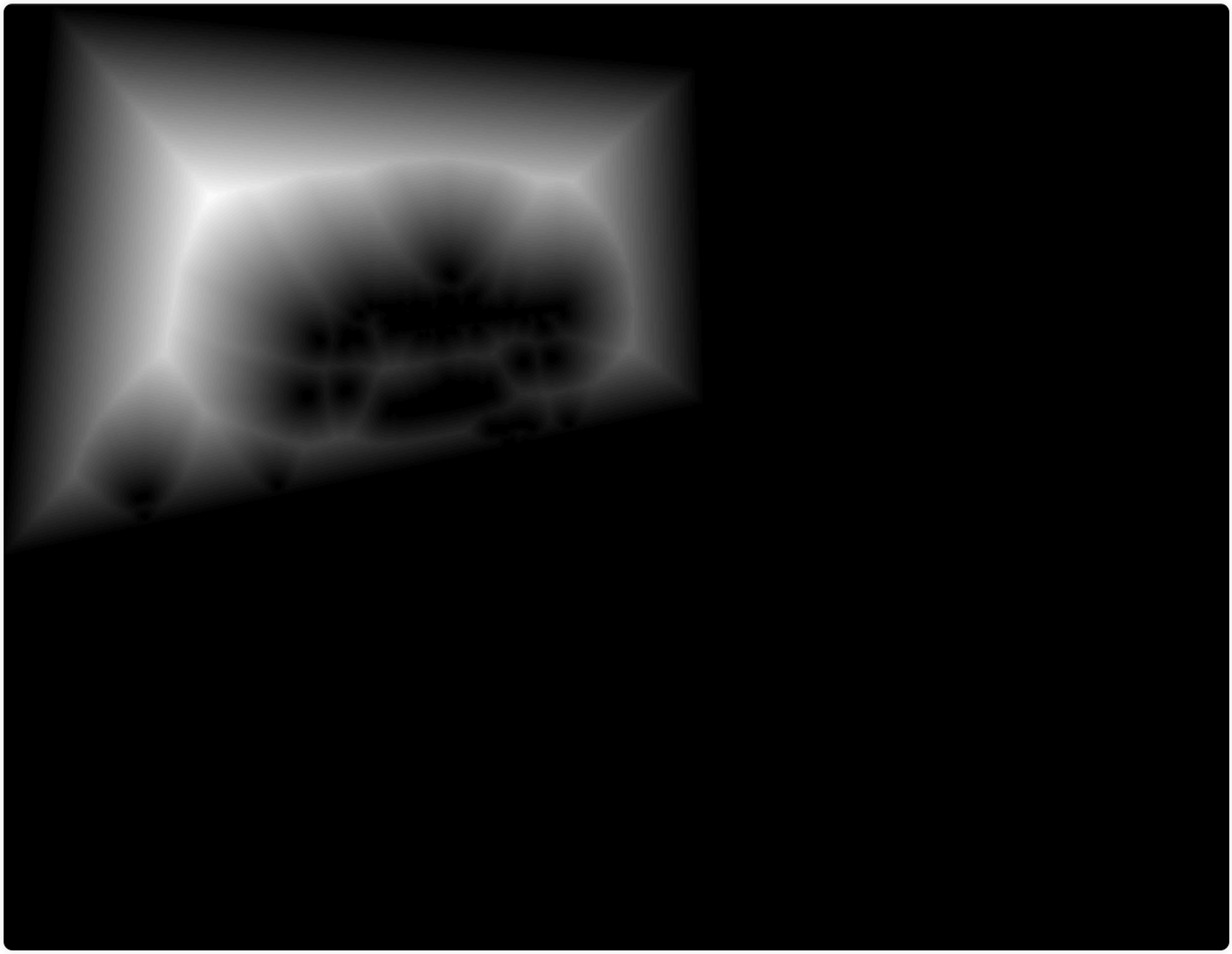
Part 5: Blending Images into a Mosaic

This was probably the heftiest part of the project. I started off by setting up a canvas large enough to accommodate both images side by side. I also calculated the necessary positions to place each image on the canvas.

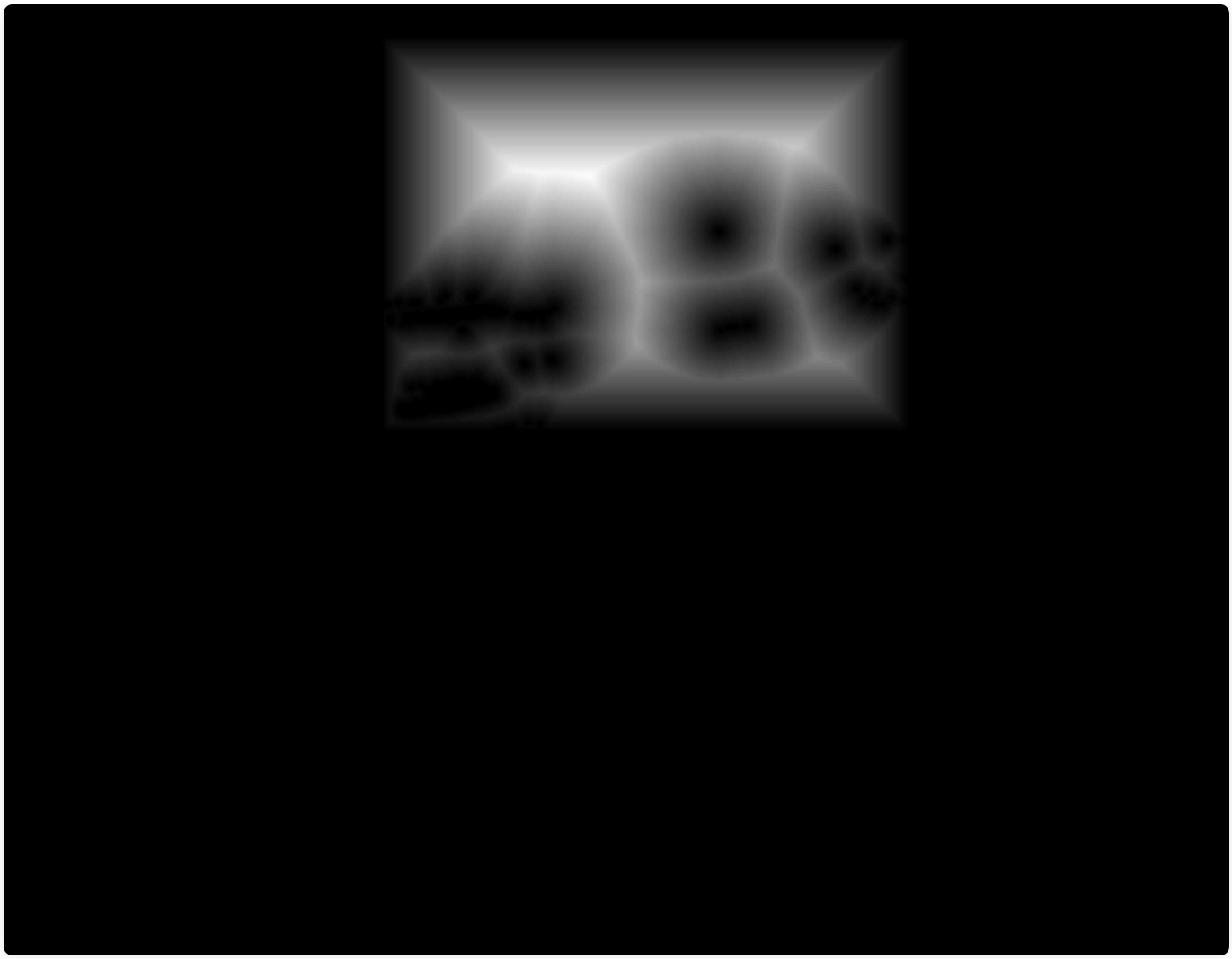
To handle the blending, I generated distance transforms to create masks for each image, which helped identify the transition zones between them. These masks were normalized and used to blend the overlapping parts of the images smoothly. I then applied two-band blending by separating each image into low and high-frequency components. To generate the former, I simply defined a 2-D gaussian kernel with size 15 x 15 and standard deviation 4, and convolved the image with it. To get the latter, I took this lower-resolution image, subtracted it from the original image, and then multiplied it by a binary mask to get rid of any noise. The low-frequency components were blended using weighted averages based on the values at a specific coordinate in the distance transform arrays, while the high-frequency components were blended based on which image had a stronger presence at each pixel, meaning that if image A had a higher presence, I would take the value at the high-pass version of image A.

Finally, I combined the low and high-frequency blended components to produce the final composite image, maintaining both smooth transitions and sharp details across the seam.

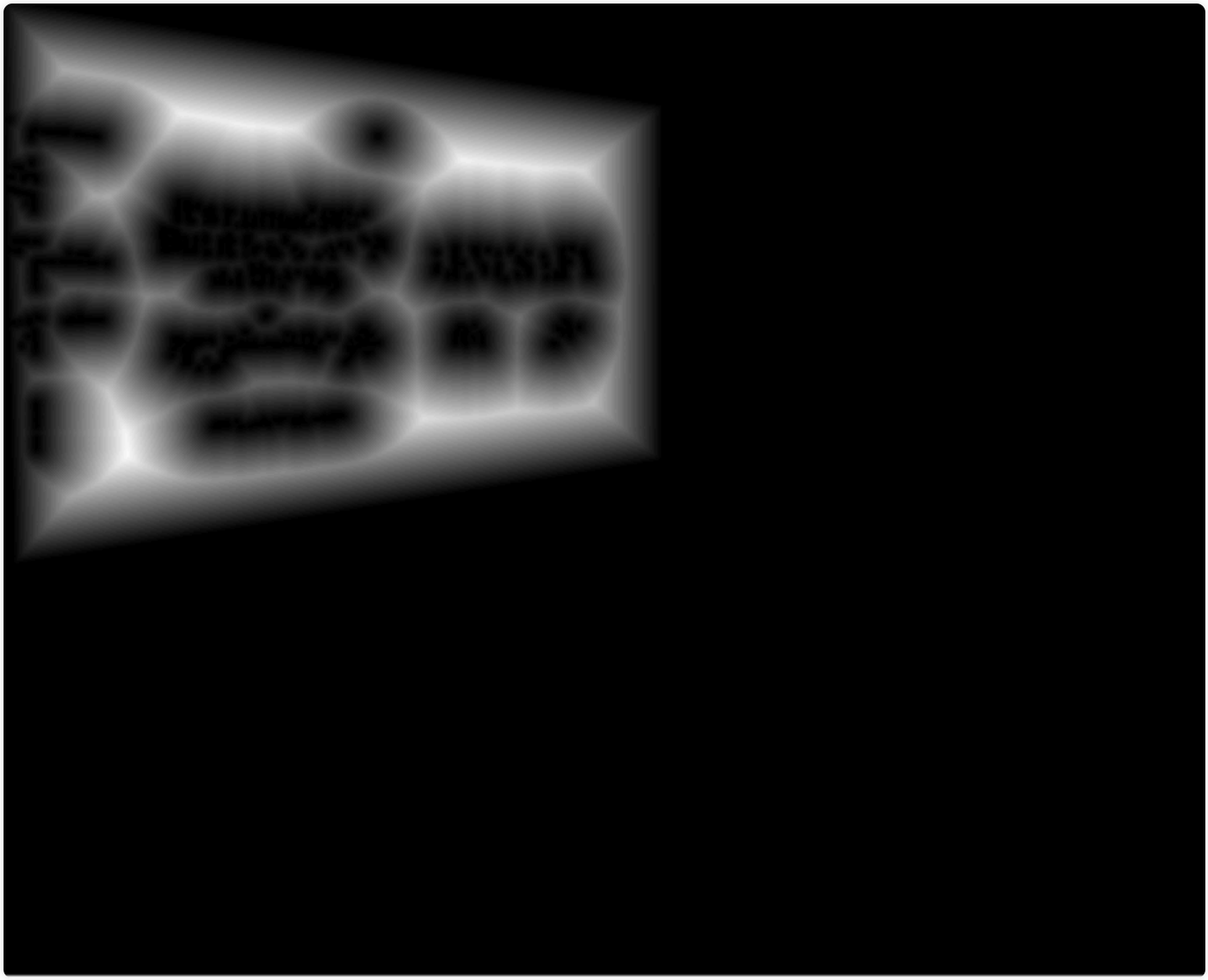
Distance Transforms



Distance Transform for warped Living Room 1

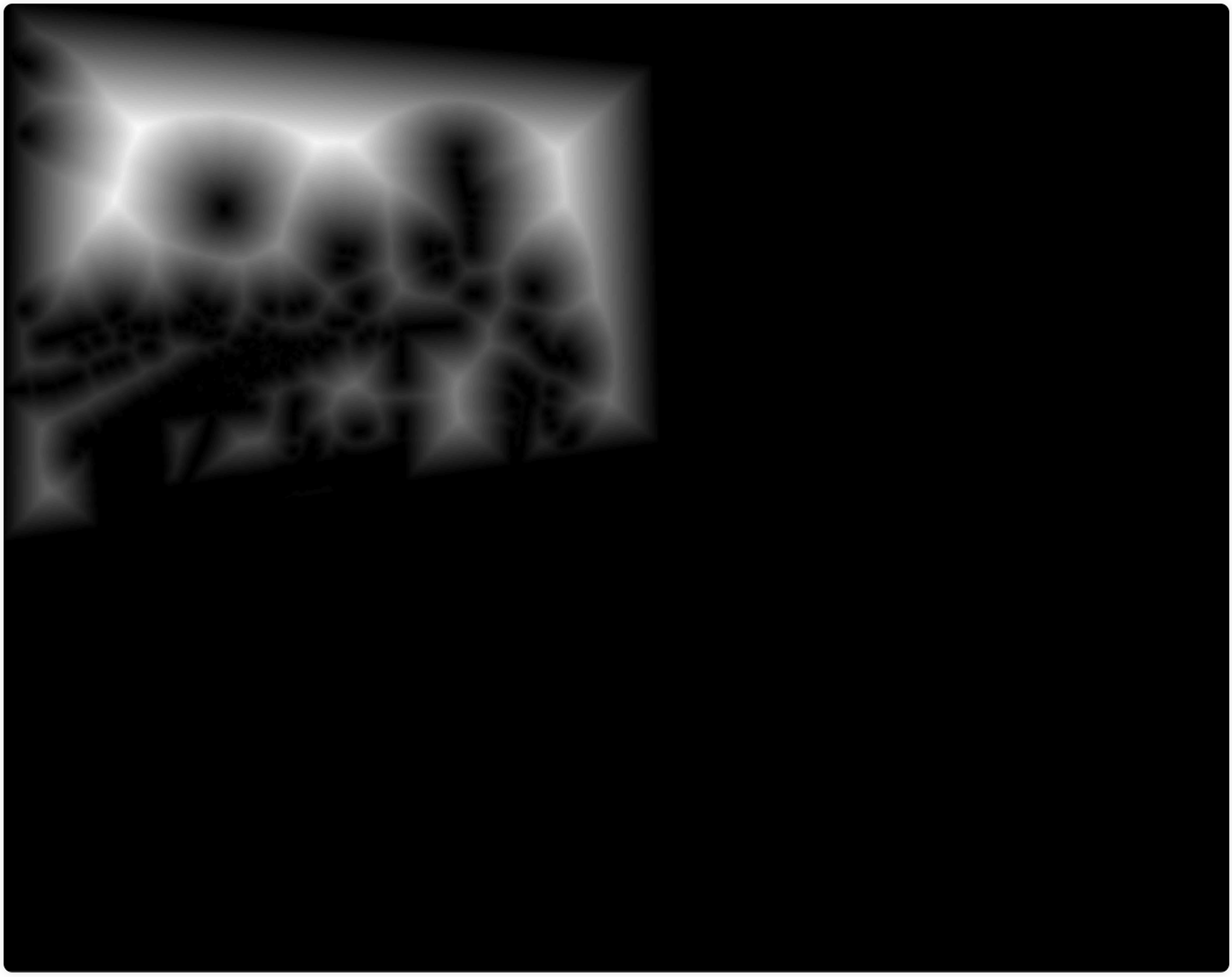


Distance Transform for Living Room 2



Distance Transform for warped Car Posters 1

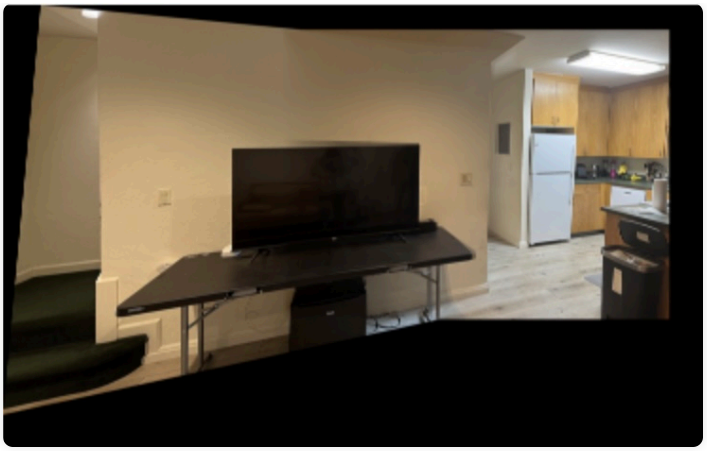
Distance Transform for Car Posters 2



Distance Transform for warped Desk 1

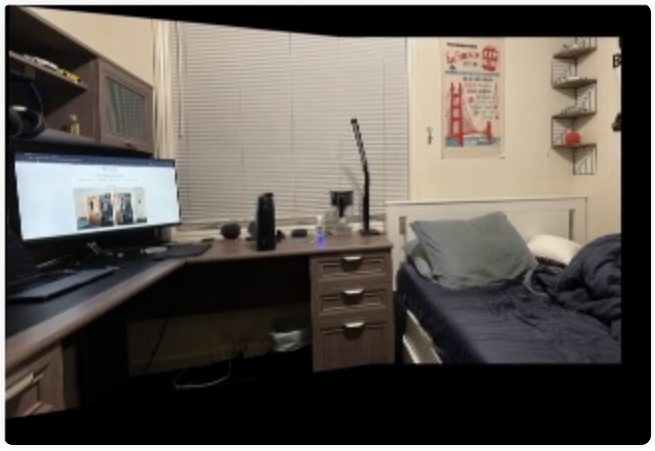
Distance Transform for warped Desk 2

Final Mosaics



Living Room

Car Tapestries & Posters



Desk & Bedroom

Saatvik Billa CS180 Project 4B: Feature Matching for Autostitching

Step 1: Harris Interest Point Detector

We start off the process by detecting prominent features in each of the images, namely corners. The Harris Detector takes care of this by looking at how the image gradients change in different directions for each pixel, and evaluating it as a corner if it sees that there is a great change in pixel intensity in multiple corners. As you can see on the following initial images, the detector detects a boatload of points on each, not guaranteeing a one-to-one correspondence between the images as the process is undergone separately on both images. We will see in later steps how you take care of this concern

Image 1 with Harris Points

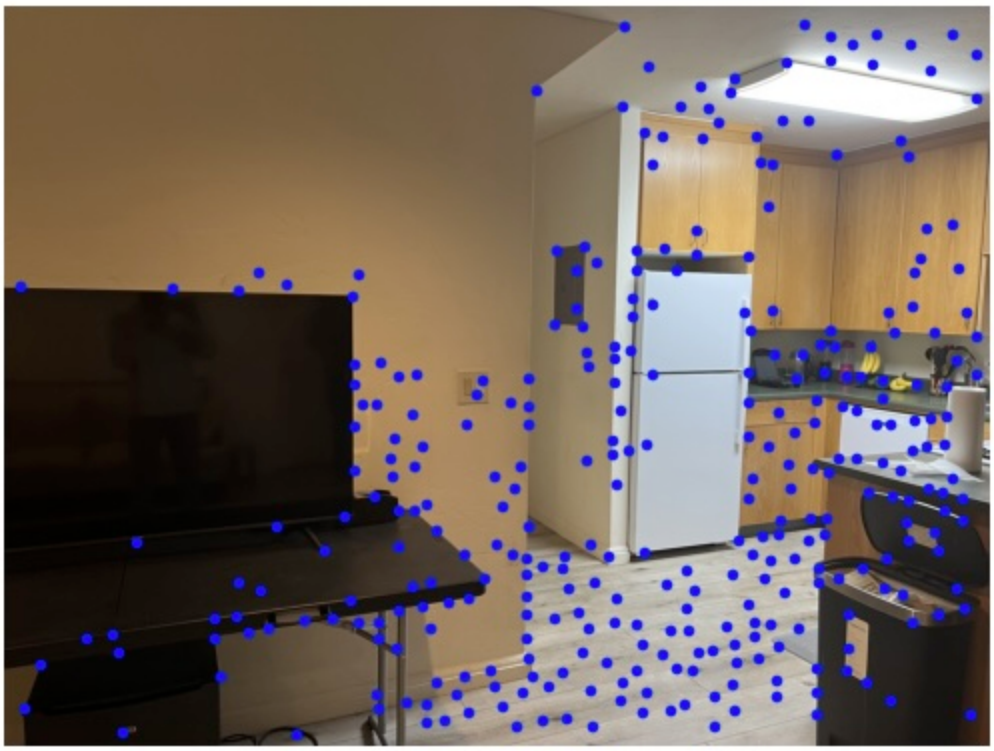


Image 2 with Harris Points

Step 2: Adaptive Non-Maximal Suppression

Adaptive Non-Maximal Suppression, or ANMS for short, is an algorithm designed to remove "redundant" information returned by Harris Detector, meaning that it attempts remove clusters of points and spread out all the significant points. To do this, I basically took the list of coordinates returned by the Harris Detector, and I tracked the minimum distance for each to another coordinate that had a higher corner strength score. I then took the list of all these distances, sorted them by farthest distance to closest distance, and then took the top 100 of these points. This way, I get 100 points that are spread out on the image. As you see on the following two images, the points are now much more spread out and less cluttered.

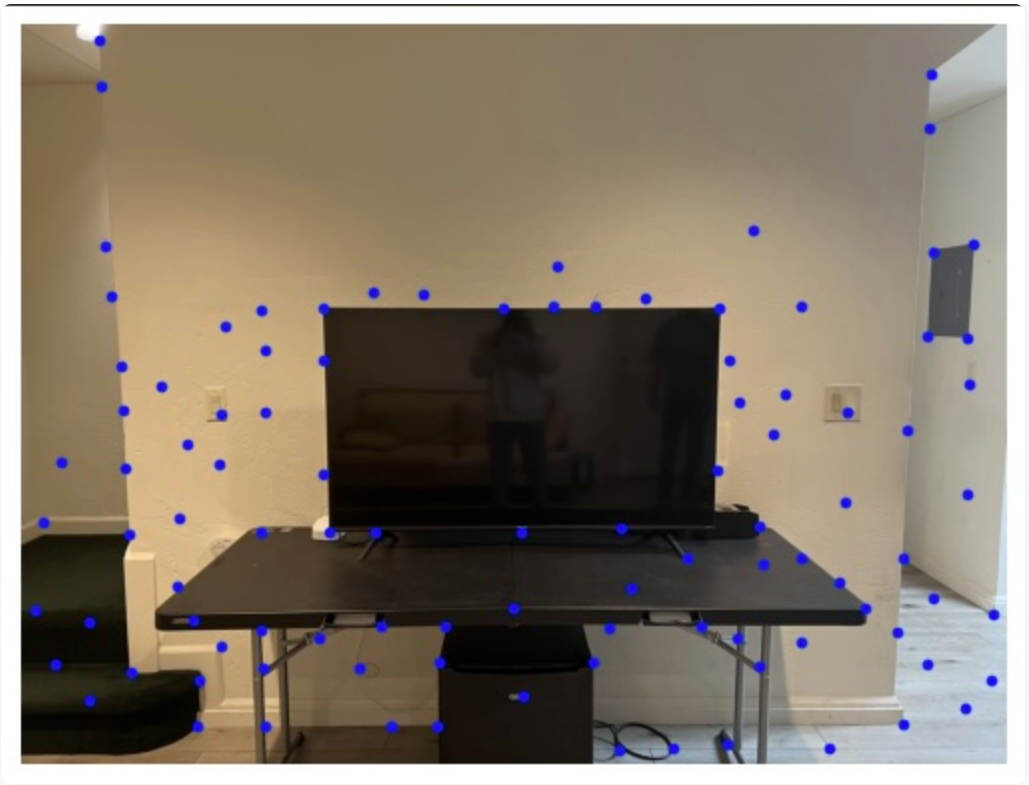


Image 1 with ANMS

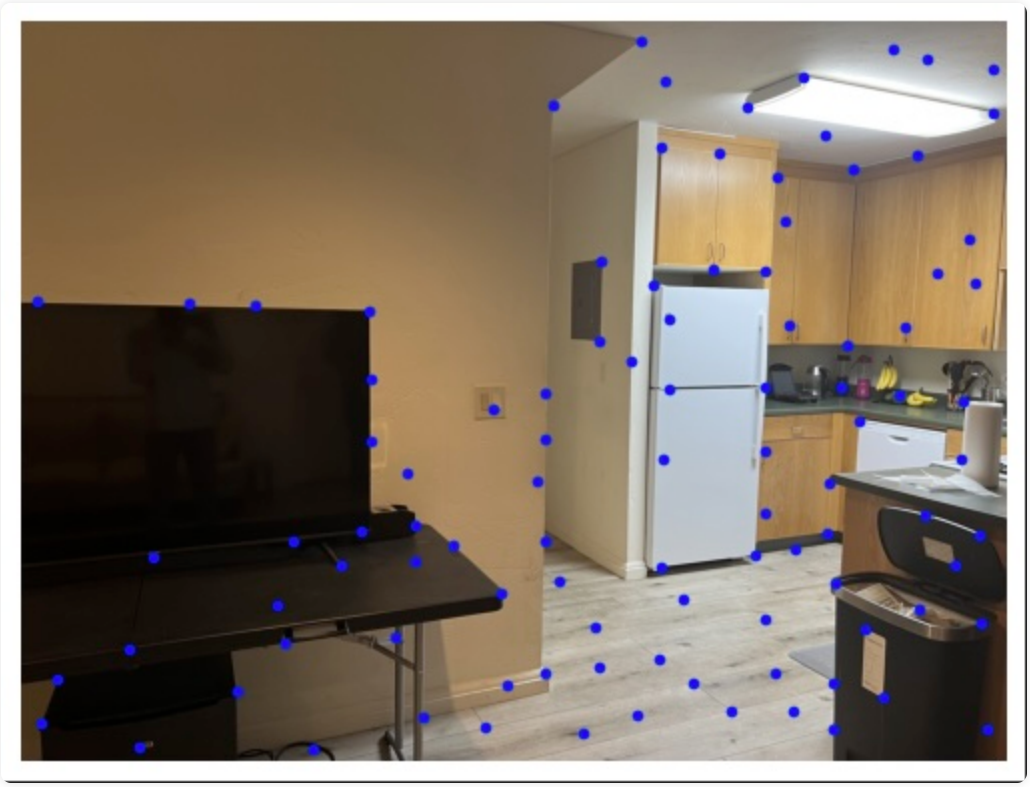


Image 2 with ANMS

Step 3: Feature Descriptor Extraction & Feature Matching

Onto the third step, where we want to attribute a unique identifier to each point such that we can use it to match points from one image to another later on. Up till this point, the set of selected features on either image have no relation to each other as both Steps 1 & 2 were conducted separately on the image, so extracting the features is the first step towards matching them.

To get the feature descriptors, I took the points returned by the ANMS algorithm, and for each one, I extracted a 40x40 pixel patch from the image surrounding that point. Then I scaled it down to an 8x8 patch using sk.transform.rescale, with a scale factor of 0.2, and proceeded to normalize this patch by subtracting it's mean from it and dividing the result by the standard deviation. I then flattened this patch into a 1-D array and eventually created a matrix of size (100, 64), where each row was one point and each column represented a pixel value from the 8x8 unique patch.

Once I got the descriptor matrix for both images, I used the dist2 function provided by the course staff to get my hands on a distance matrix of size (100, 100), where the i,jth entry of the resultant matrix was the Euclidean distance between the ith descriptor vector of image 1 and the jth descriptor vector of image 2.

Then what I did was I looped through each descriptor vector in image 1 and found two things: the descriptor vector of image 2 that it was closest too and second closest too. I then applied the concept of Lowe's ratio to check if the ratio of the former to the latter was below 0.8, and if so, I would go ahead and append the coordinate in image 1 and it's nearest neighbor as a feature match. Lowe's ratio is employed to eliminate matches where the closest and second-closest matches are almost equally similar.

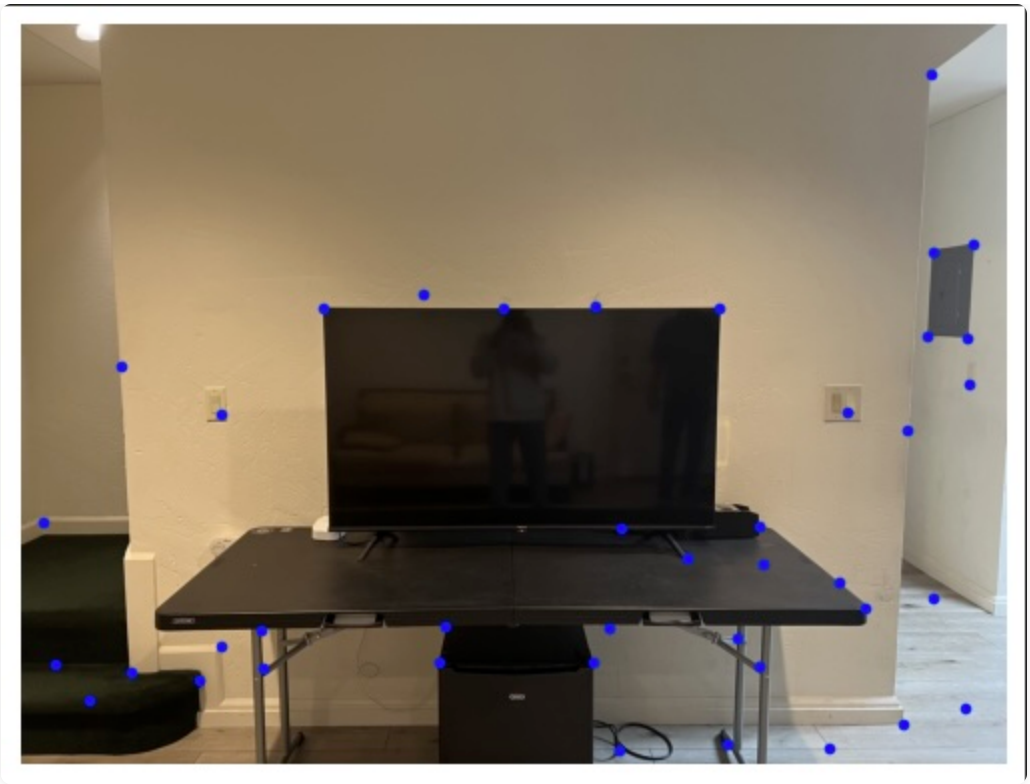


Image 1 with Initial Correspondences

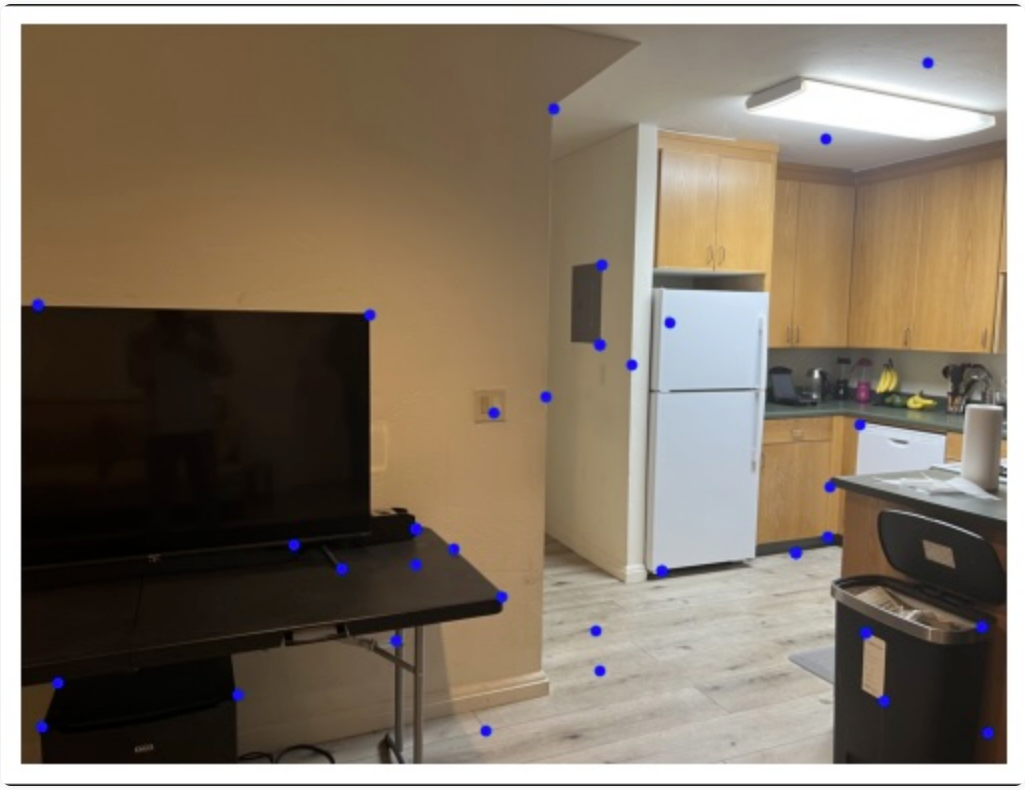


Image 2 with Initial Correspondences

Step 4: 4-point RANSAC

Now that we have our correspondences, the point of RANSAC is to compute the best homography by picking samples of points, using them to compute H, and then applying that H to each point in image A's correspondences to get the projected points in image B's space. If the projected point was within a distance of whatever's defined by the value inlier_threshold, which in my case was 400 pixels since my images were big, then I would add it to my set of final correspondences. I did this proceess 500 times before I got my final list of correspondences. The first two images show the final correspondences on each image and the third one shows the projected points from image 1 besides their matched pairs in image 2. For the most part, they are pretty accurate. The final homography matrix I use for creating the mosaic is the one that returns the most inlier points.

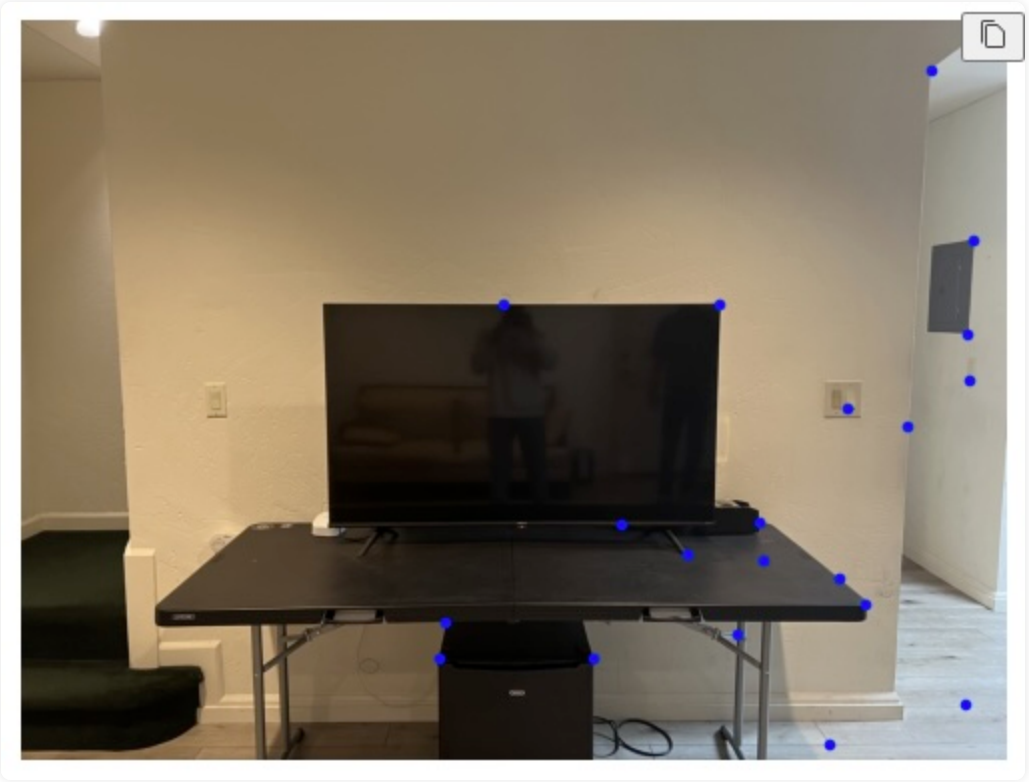


Image 1 with Final Correspondences

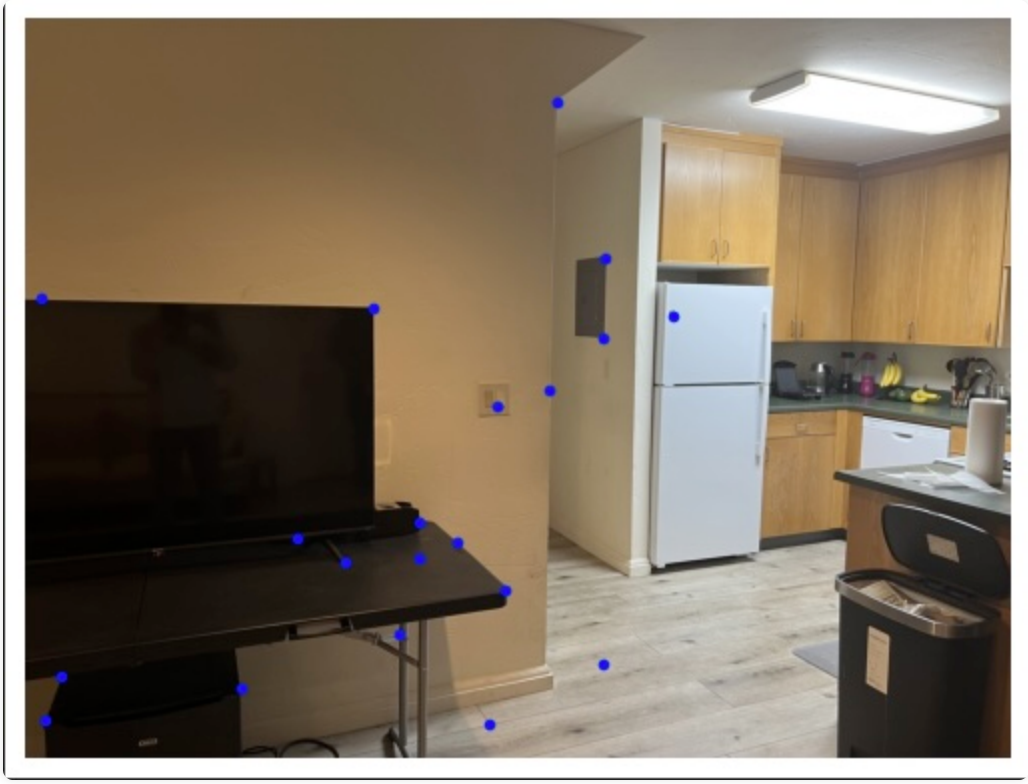
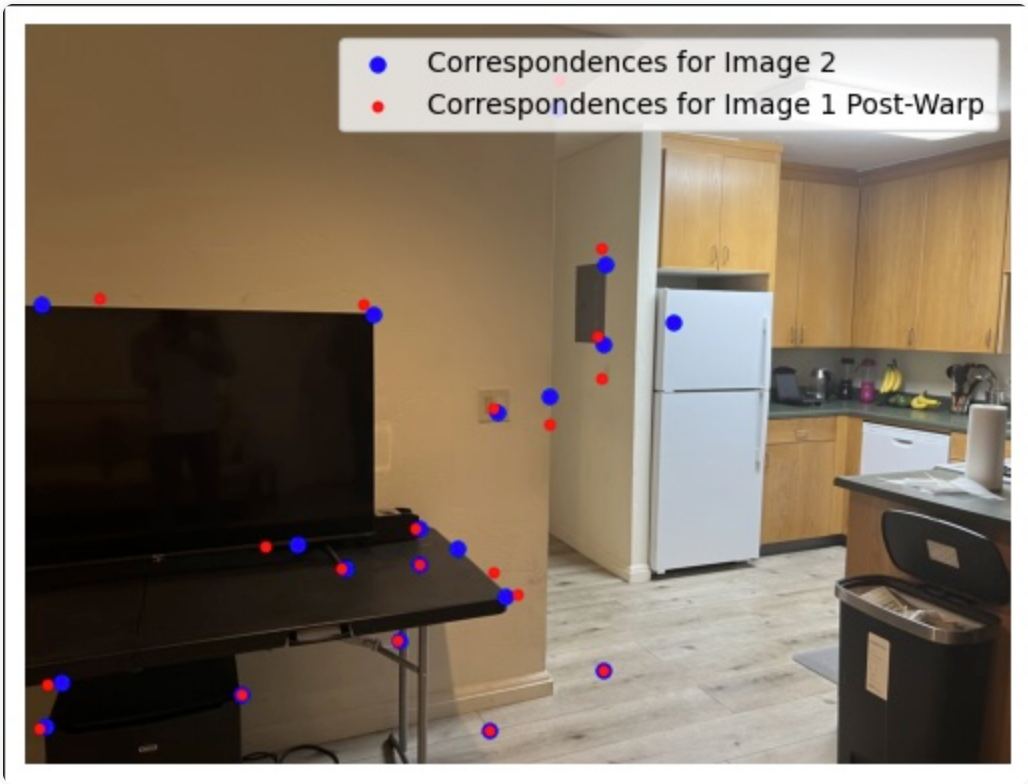


Image 2 with Final Correspondences

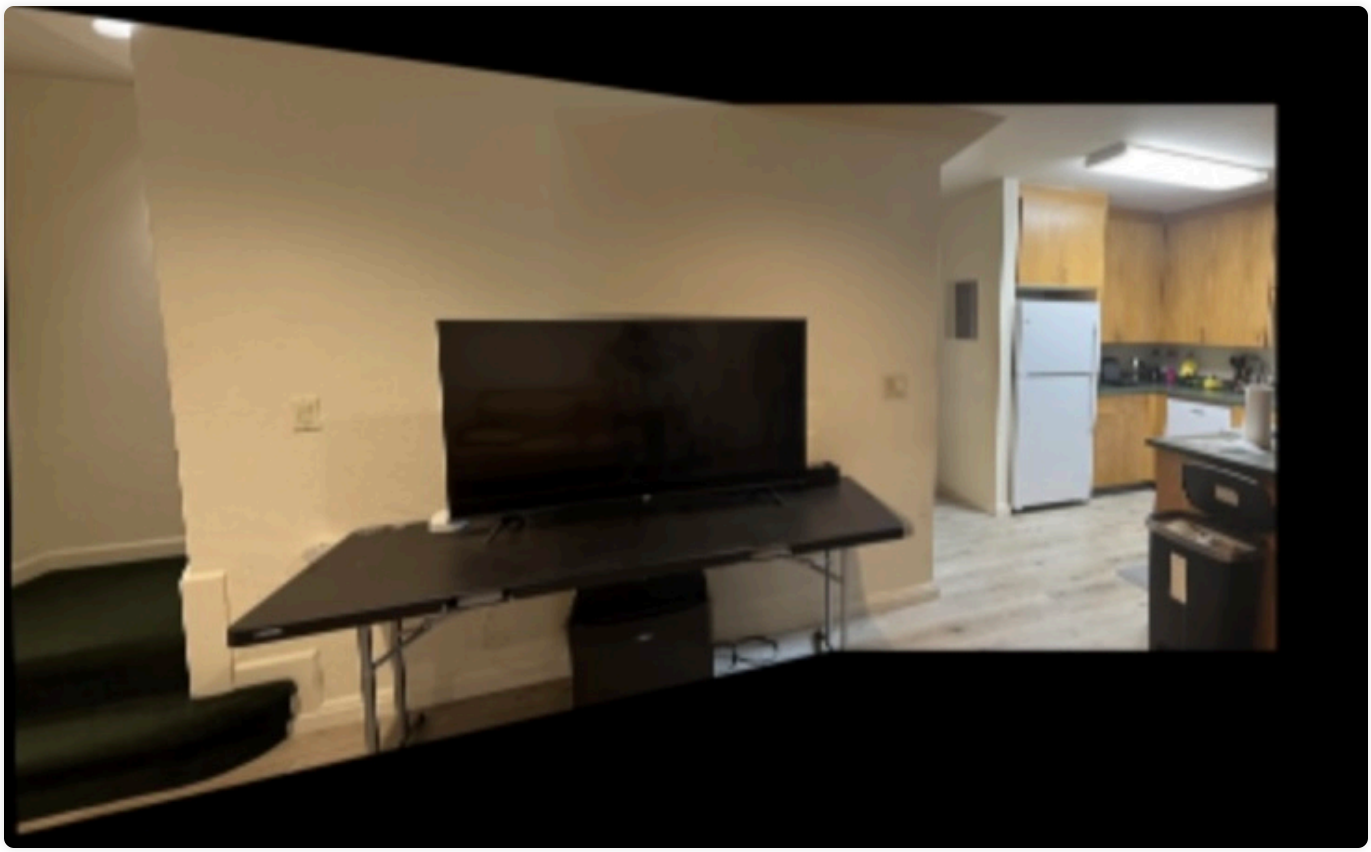


Projected Image 1's points alongside Image 2's final Correspondences

Final Mosaics

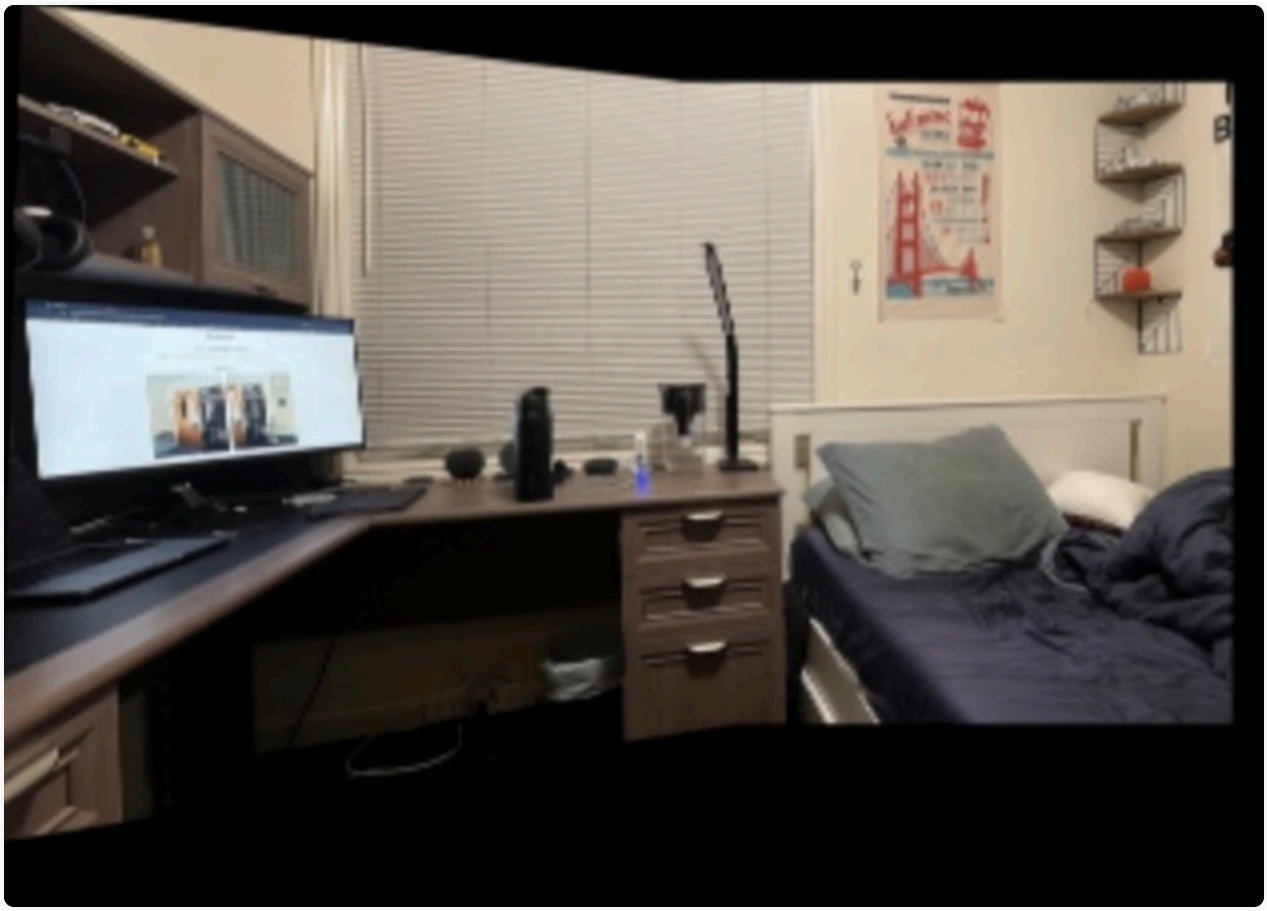
Now that I had the correspondence points for each image, where there was a one to one mapping between images, I utilitzed the functions I implemented in part A to warp the first image to the homography of the second, and then used both stitching and color blending methods to create the mosaic.

Here are the final mosaics returned by the Auto-stitching process! As you can see, they are pretty similar to the ones created manually, with a few differences here and there. Regardless, I think they all look great!

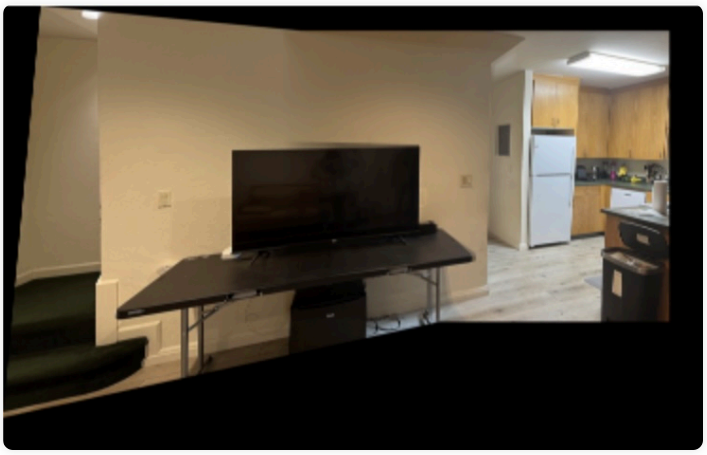


Living Room AUTO

Car Tapestries & Posters AUTO

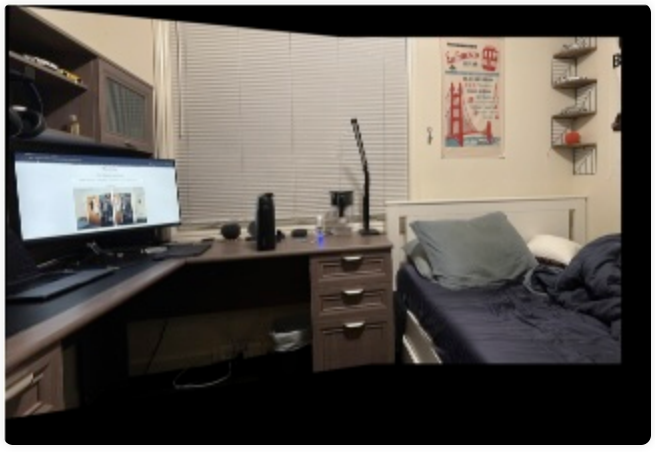


Desk & Bedroom AUTO



Living Room MANUAL

Car Tapestries & Posters MANUAL



Desk & Bedroom MANUAL

Takeaways

I really enjoyed this project as it gave me great insight into what goes on behind the scenes in making those cool panoramas on our phones! Doing the process manually first and then implementing an automatic way to do it was really sick and really re-inforced the concepts for my learning!