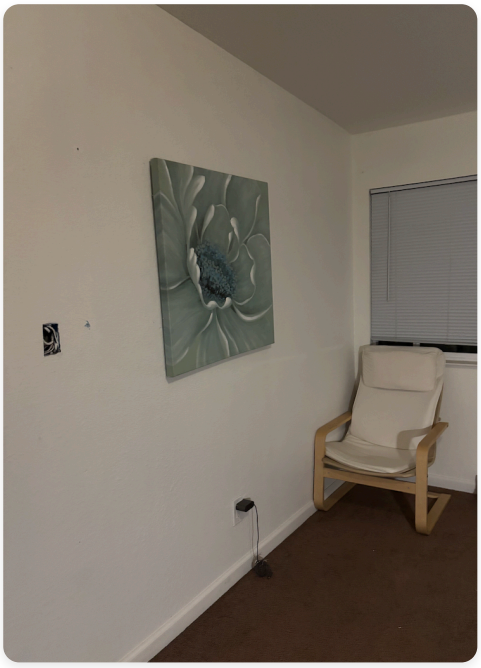
Part A: IMAGE WARPING and MOSAICING

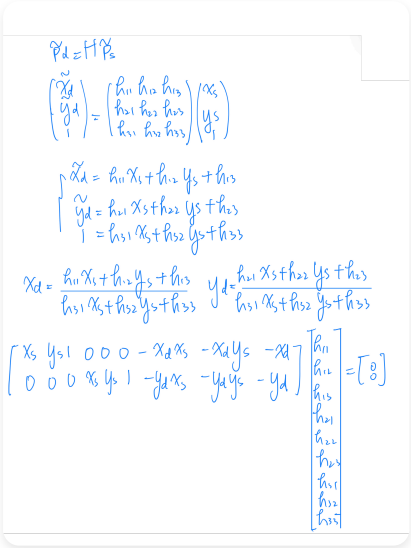
1. Shoot and Digitize Pictures

For image warping and mosaicing, I took three pictures, one outside of Moffit, one inside Moffit and one in my room.

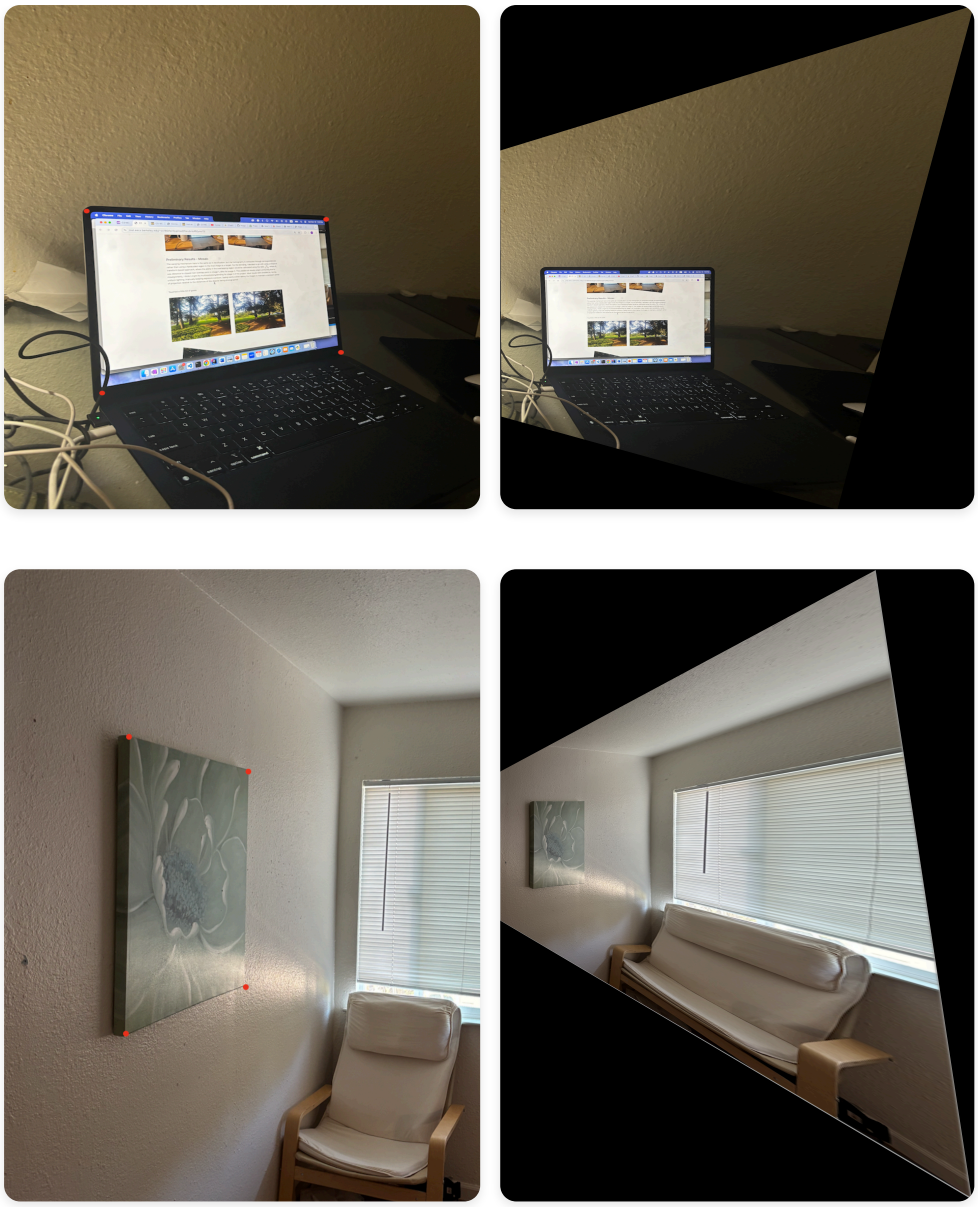
2. Recover Homographies

To compute the entries of the homography matrix H, I set up a linear system of equations in the form Ah = b, where h is a vector containing the eight unknown entries of H. By selecting more than four corresponding points, I ensured that the system was overdetermined, which enhances stability and reduces sensitivity to noise. I then utilized Singular Value Decomposition (SVD) to solve the system, taking the last row of V^T as the solution for h. After deriving the formula, I use the tool from Project 3 to select the points from each image.



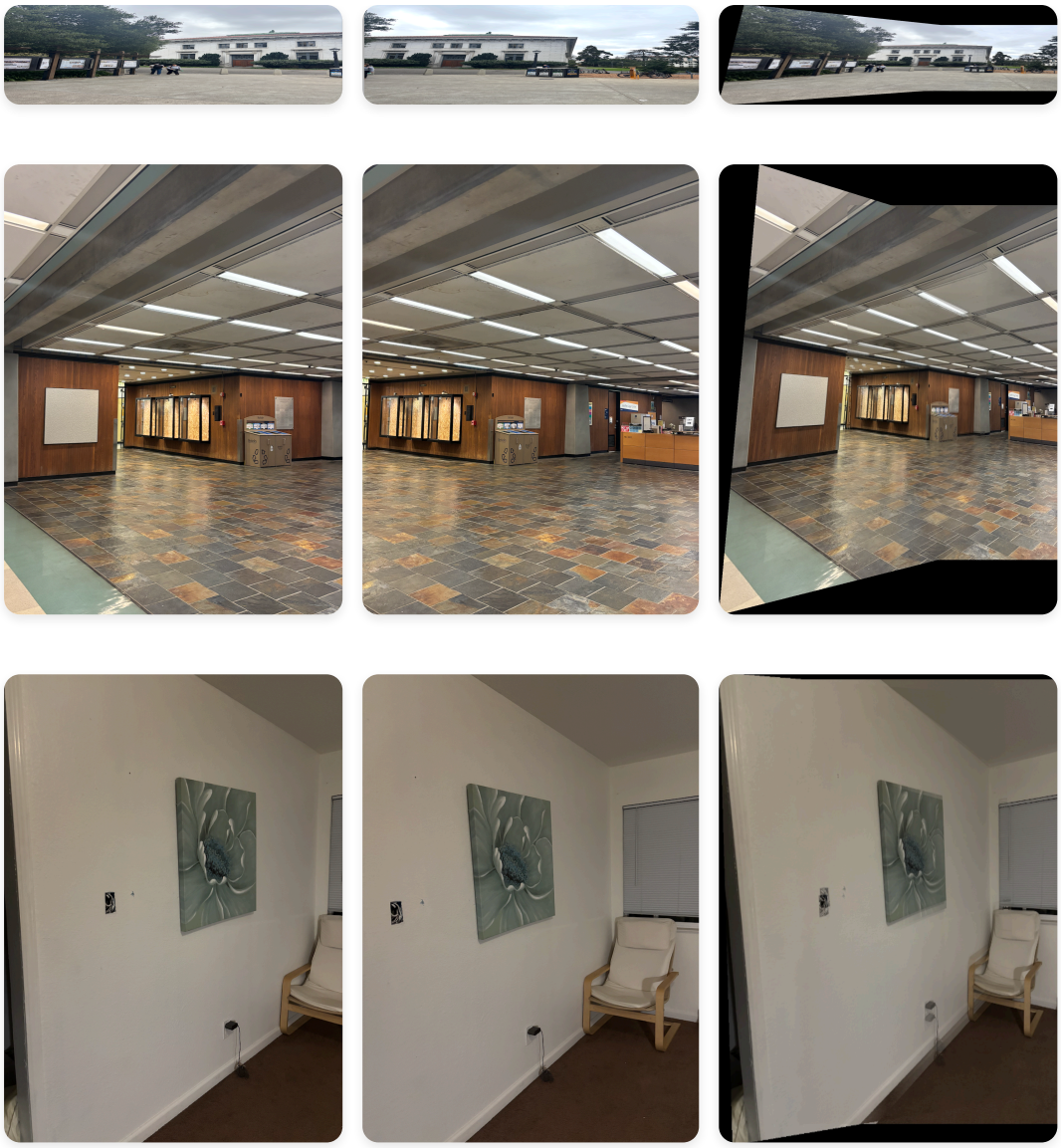
3. Image Rectification

To test my code, I need to perform image rectification on a photograph containing known rectangular objects, such as paintings or posters, using homography. This step ensures that my homography and warping functions work correctly before proceeding further.



4. Wrap and Blend Images into a Mosaic

In this process, I developed a method to create an image mosaic by warping multiple images to align them correctly in a common projection. Instead of allowing one image to overwrite another, which can result in noticeable edge artifacts, I employed weighted averaging to blend the images seamlessly. By determining the final mosaic size first, I warped all images into this predefined size, effectively stacking them together. I then applied a feathering technique to smoothly blend the overlapping regions of the images, allowing for a cohesive final image that minimizes artifacts and enhances the overall visual quality of the mosaic.



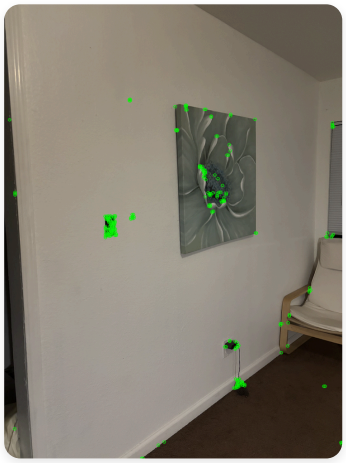
Part B: FEATURE MATCHING for AUTOSTITCHING

1. Harris Interest Point Detector

I implemented the Harris Interest Point Detector using the provided sample code in harris.py, focusing on identifying corner point points at a single scale without sub-pixel accuracy. By leveraging the existing code, I efficiently detected Harris corners in an image and created a visual representation that overlays the detected corners on the original image to highlight the points of interest.

2. Implement Adaptive Non-Maximal Suppression

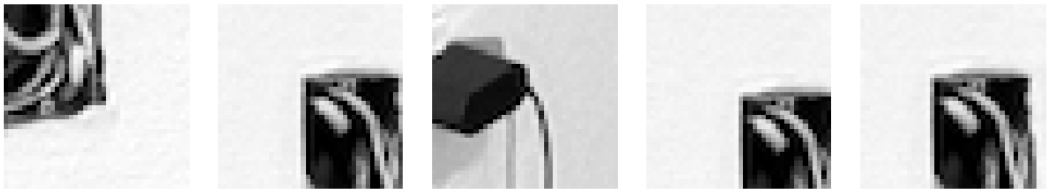
I implemented Adaptive Non-maximal Suppression (ANMS) to address the clustering of Harris interest points, which often leads to redundant information in feature matching. ANMS aims to evenly distribute these interest points across the image for more effective matching. To achieve a target number of interest points, I utilized a binary search approach to determine the optimal radius needed to filter down to this desired quantity. My algorithmic methods are based on the techniques described in the paper "Efficient adaptive non-maximal suppression algorithms for homogeneous spatial keypoint distribution" by Bailo et al.



3. Feature Descriptor extraction

I implemented feature descriptor extraction by focusing on axis-aligned 8x8 patches from interest points. To ensure the descriptors are robust, I sampled these patches from a larger 40x40 window, which allows for a well-blurred descriptor. I applied bias and gain normalization to the extracted descriptors to enhance their effectiveness. I ignored the wavelet transform section of the original method. In this process, I sampled points from a patch of a specified radius with designated spacing, taking care to use a low-pass filtered image (equivalent to a Gaussian blur) to prevent aliasing. After sampling, I flattened and normalized the features to ensure they have the desired properties.

I inlcuded some visualized Feature Descriptor



4. Implement Feature Matching

I implemented feature matching by taking a list of feature descriptors from multiple images and applying Lowe's ratio test to identify good matches. Using the k-nearest neighbor approach, I filtered matches based on the distance ratio between the closest and second-closest matches, with a default threshold of 0.75. I also calculated the average distance of good matches and filtered out those exceeding this average to enhance match quality.

RANSAC

I implemented a robust homography estimation using the 4-point RANSAC algorithm. The process starts by randomly sampling four unique points from two sets of matched keypoints, im1Points and im2Points. For each sample, I computed the homography matrix H H that relates these points. After calculating the reprojection error by applying the homography to the original points and measuring the Euclidean distance to the corresponding points, I identified inliers based on a predefined error threshold. The algorithm iterates for a specified number of times, keeping track of the homography that yields the maximum number of inliers. Finally, I returned the best homography matrix along with the inlier points that were used in the estimation, ensuring a robust alignment of the two sets of points despite potential noise and outliers in the data.

manuallyauto