CS 180 Project 4: Stitching Photo Mosaics

By Ethan Jagoda

Table of Contents

Introduction

This project explores how to use homographic transformations to stitch together a photo mosaic. As well, we use homographic transformations to warp images to rectify shapes to seem like they are from a different point of view.

Part A: Image Warping and Mosaic Creation

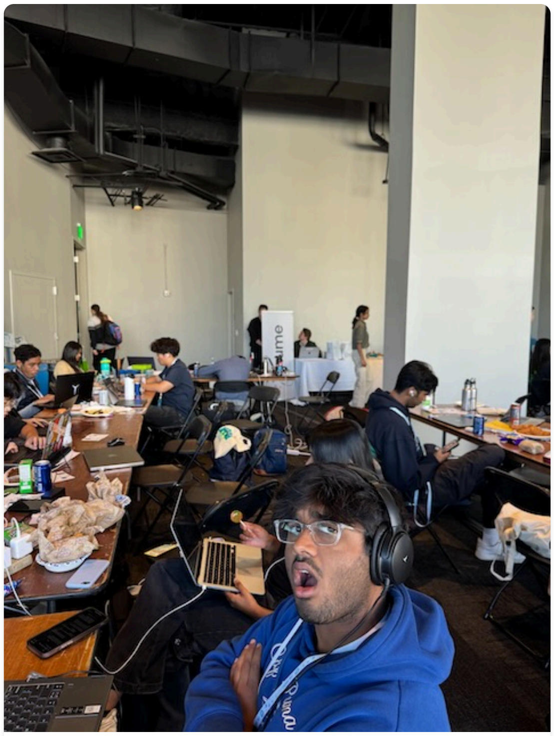
Raw Images Used

These are some of the raw images I used in this project.

Berkeley Way West Bottom

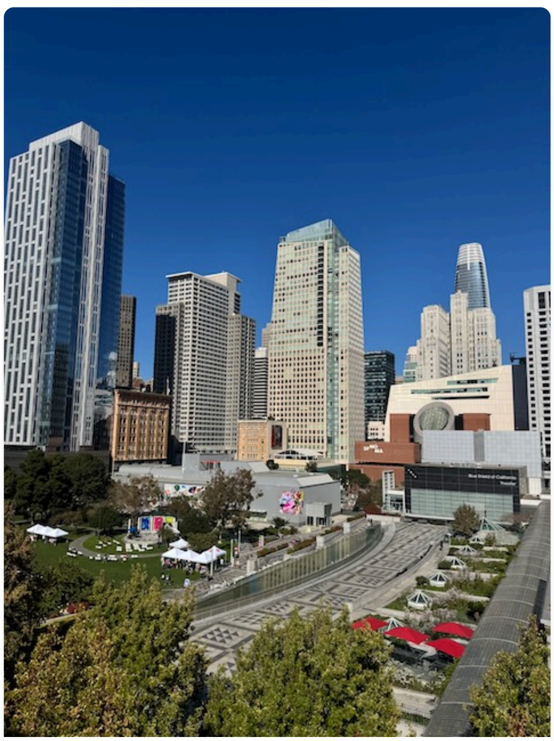
Berkeley Way West Top

Table of Contents

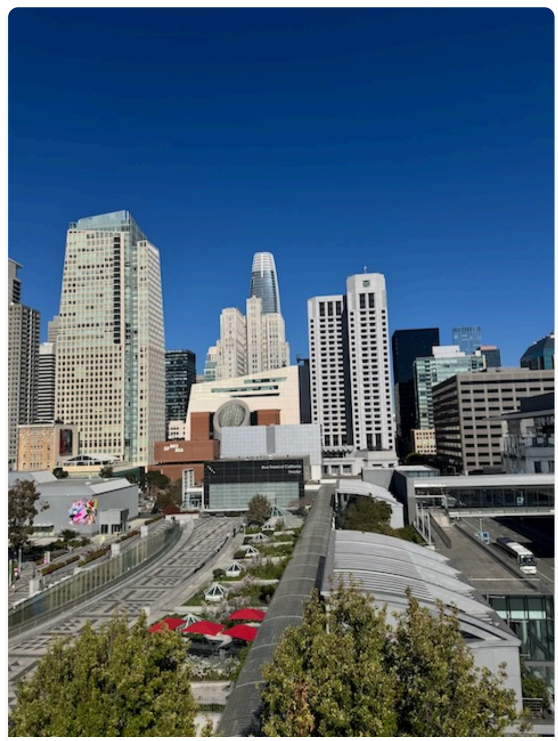


Pranav Left

Pranav Right



SF Skyline Left



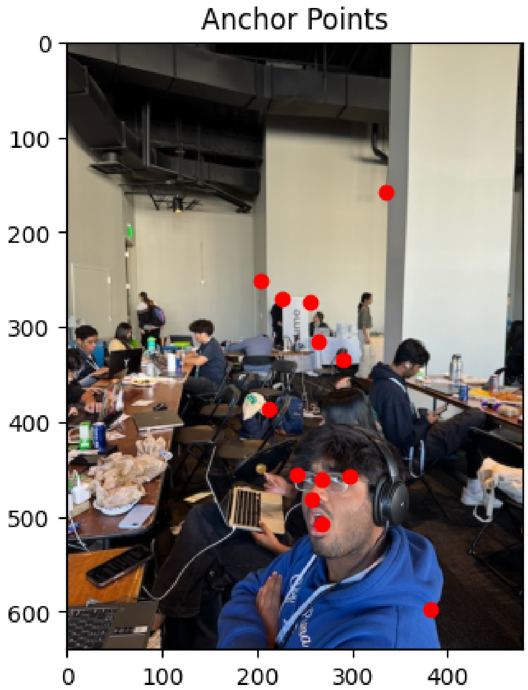
SF Skyline Right

Recovering Correspondences

I used a GUI to manually select correspondences between the two images. This was done by clicking on a point in one image and then clicking on the corresponding point in the other image.

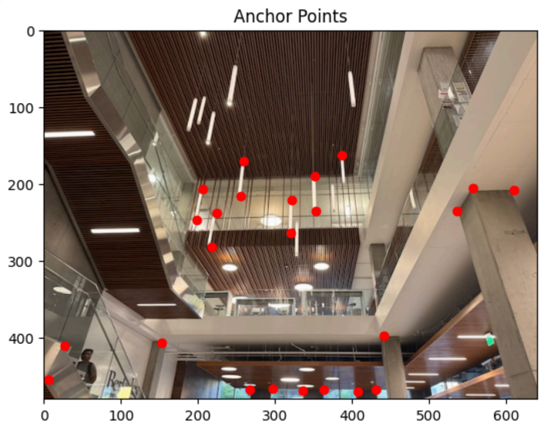
This is an example of one of the images with the correspondences marked.

Table of Contents

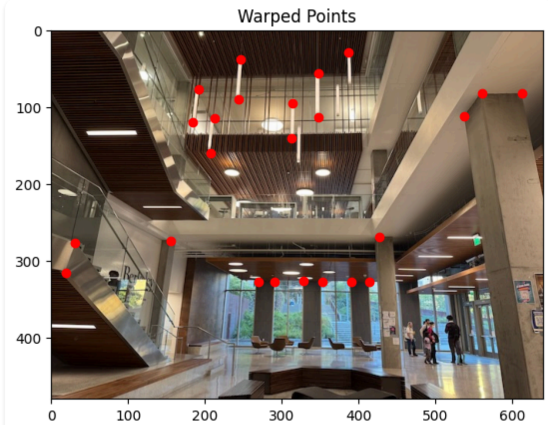


Pranav Left Points

Pranav Right Points



Berkeley Way West Bottom Points



Berkeley Way West Top Points

Computing the homographic transformation

The homographic transformation is computed using the following equation:

$$Hp = p'$$

Where H is the homography matrix, p is a point in the source image, and p' is the corresponding point in the destination image.

Expanding this out, we get the following equation:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = w \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Expanding this out further, we get the following equations:

$$wx' = ax + by + c$$

$$wy' = dx + ey + f$$

$$w = gx + hy + 1$$

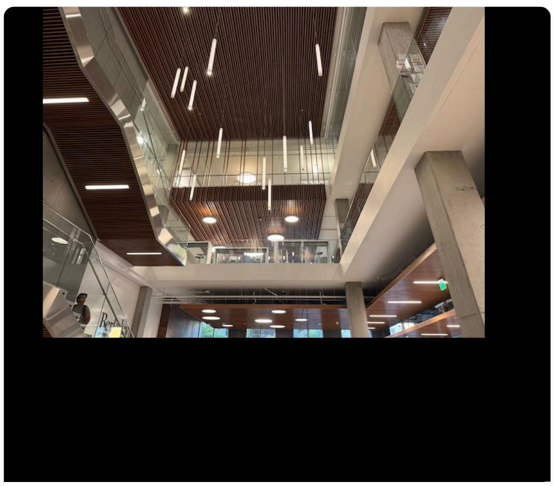
Reframing this so that we can use the new points and old points to solve for the homography matrix, we get the following equations:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -yx' \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

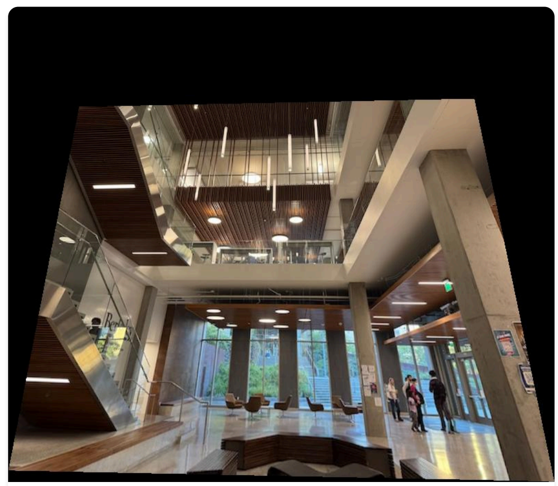
To reduce the chance of overfitting, we can use more than 4 points and then solve this system of equations using least squares to get the homography matrix.

Warped Images

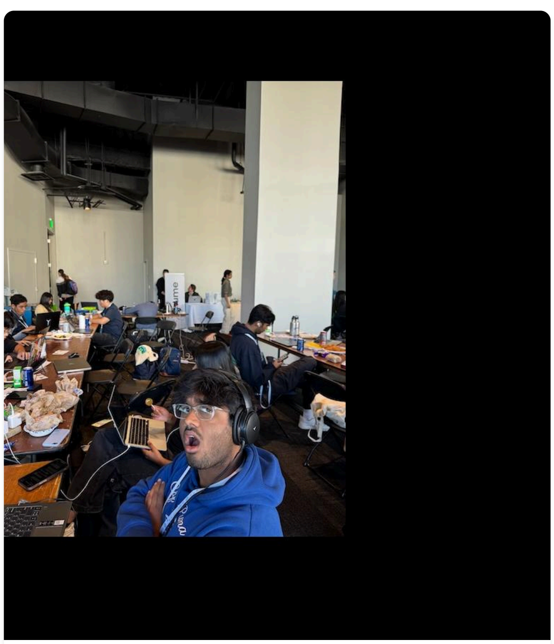
Here are some of the warped images I got from this technique.



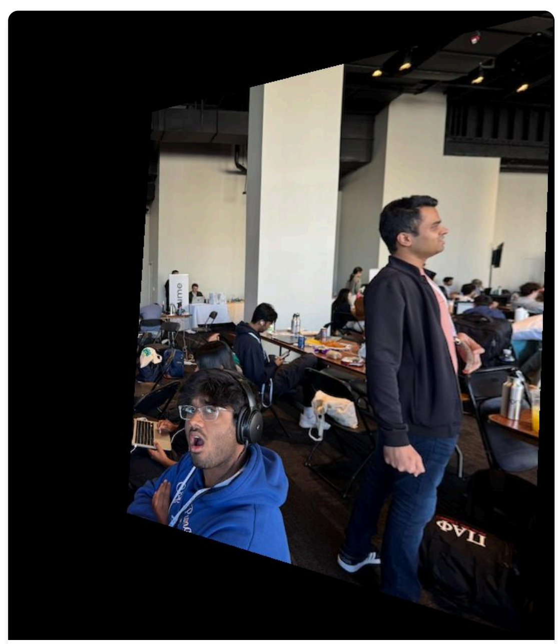
Berkeley Way West Top



Berkeley Way West Bottom Warped into Top



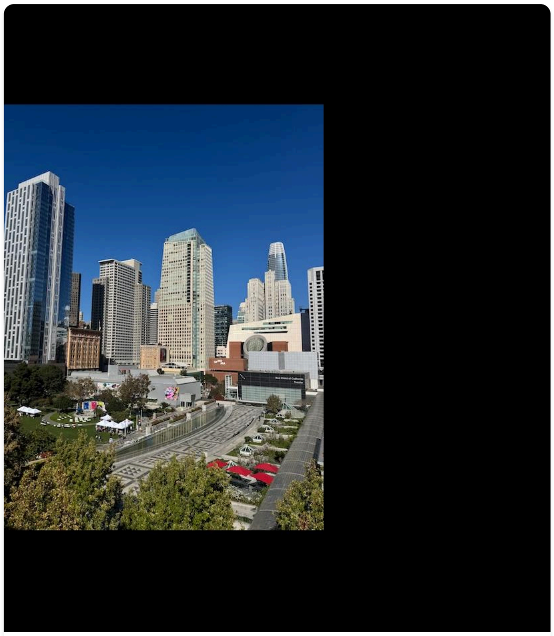
Pranav Left



Pranav Right Warped into Left

Table of Contents

Table of Contents



SF Skyline Left

SF Skyline Right Warped into Left

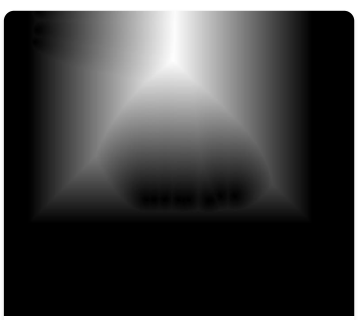
Blending Images

To blend the images, we create a mask that is the same size and shape as the warped images. Then, to ensure a smooth transition, we use a dissolve factor that changes linearly with the distance from an images edge from 0 to 1. We then use this mask to blend the pixels of the two warped images together, normalizing the pixels by the mask so that the brightnesses are correct.

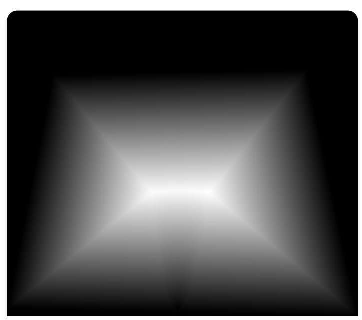
Note that the masks look a bit weird because some pixels in the images are also black which is interpreted as part of the background when calculating the alpha mask, but as long as we normalize by the mask after applying it this doesn't affect the result.

Here are some of the blended images and the corresponding masks I got from this technique.

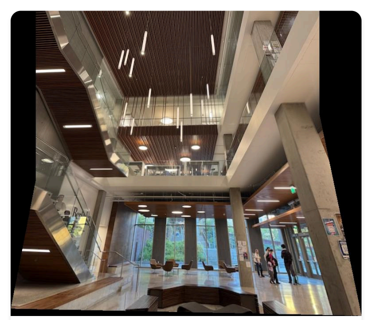
Berkeley Way West



Anchor Alpha Mask



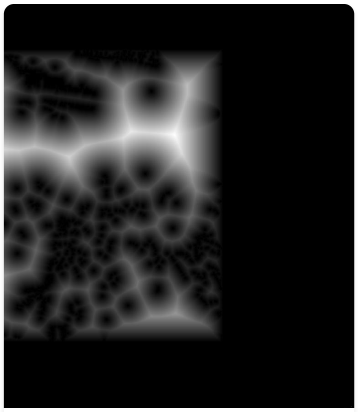
Warped Alpha Mask



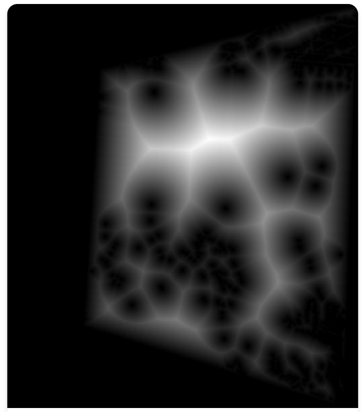
Blended Images

Pranav

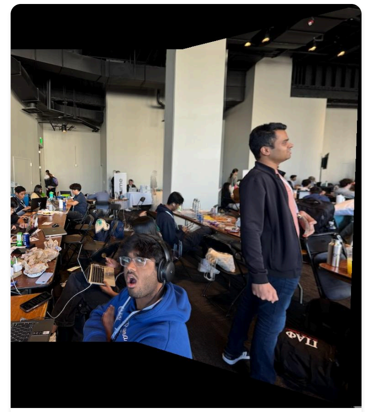
Table of Contents



Anchor Alpha Mask

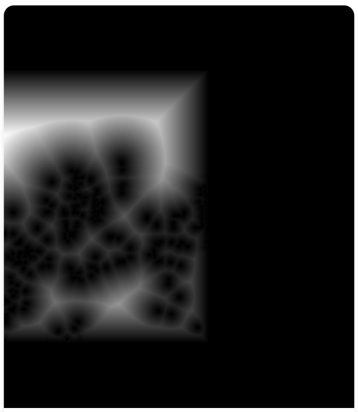


Warped Alpha Mask

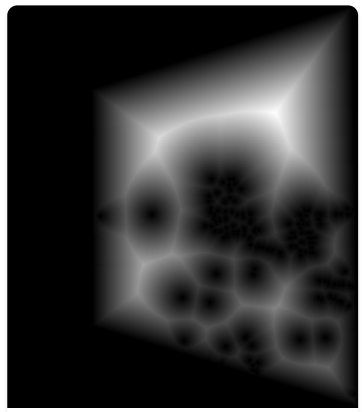


Blended Images

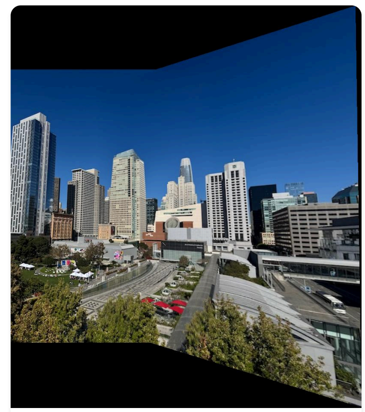
SF Skyline



Anchor Alpha Mask



Warped Alpha Mask

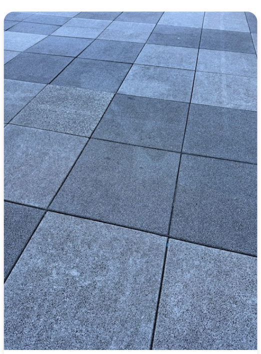


Blended Images

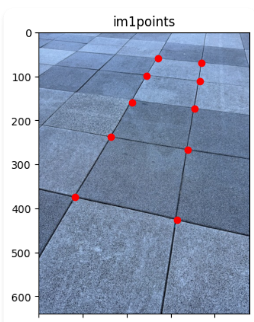
Rectifying Images

To rectify images, we can use the homography matrix to warp the image to a new point of view. This is useful for things like correcting perspective shift in photos.

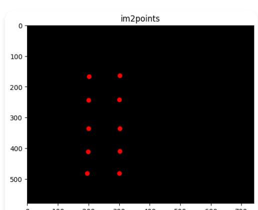
Here are some of the rectified images I got from this technique.



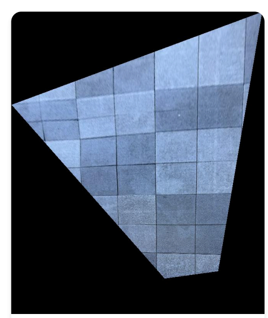
Original Tiles



Original Tile Points

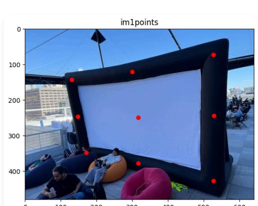


Target Tile Points

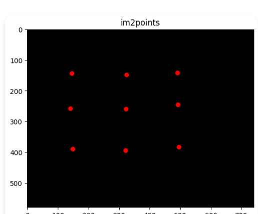


Rectified Tiles

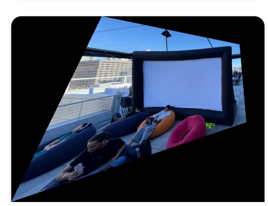
Original Presentation Screen



Original Presentation Screen Points



Target Presentation Screen Points



Rectified Presentation Screen

Part B: Feature Mapping and Autostitching

This part involves automatic feature detection and matching to stitch images together.

Harris Corner Detection

Harris Corner Detection is a technique to identify interest points (corners) in an image, which are areas with significant changes in intensity in multiple directions. The algorithm is based on the auto-correlation (second moment) matrix, which measures the local changes of the image intensity.

The main idea is to analyze how the sum of squared differences (SSD) changes when shifting a windowed patch by a small amount (u, v):

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Here, $I(x,y)$ is the image intensity at point (x,y), and $w(x,y)$ is a weighting function (e.g., a Gaussian window) that gives more importance to pixels near the center of the window.

Using a first-order Taylor expansion of $I(x+u, y+v)$, we approximate:

$$I(x+u, y+v) \approx I(x,y) + I_x(x,y)u + I_y(x,y)v$$

where I_x and I_y are the image gradients in the x and y directions, respectively.

Substituting back into $E(u,v)$, we get:

$$E(u,v) \approx \sum_{x,y} w(x,y)[I_x(x,y)u + I_y(x,y)v]^2 = [u \quad v]\mathbf{M}\begin{bmatrix} u \\ v \end{bmatrix}$$

where $\mathbf{M}$ is the second moment (structure) matrix:

$$\mathbf{M} = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

The eigenvalues λ_1 and λ_2 of $\mathbf{M}$ characterize the local intensity variation:

- If both λ_1 and λ_2 are large, the region is a corner.
- If one eigenvalue is large and the other is small, the region is an edge.
- If both eigenvalues are small, the region is flat.
  The Harris corner response function is defined as:

$$R = \det(\mathbf{M}) - k \cdot (\text{trace}(\mathbf{M}))^2$$

where $\det(\mathbf{M}) = \lambda_1\lambda_2$, $\text{trace}(\mathbf{M}) = \lambda_1 + \lambda_2$, and k is an empirical constant typically set between 0.04 and 0.06.

Corners are detected by finding local maxima of R where the response is above a certain threshold.

Here are some of the corners detected using the Harris Corner Detection algorithm:

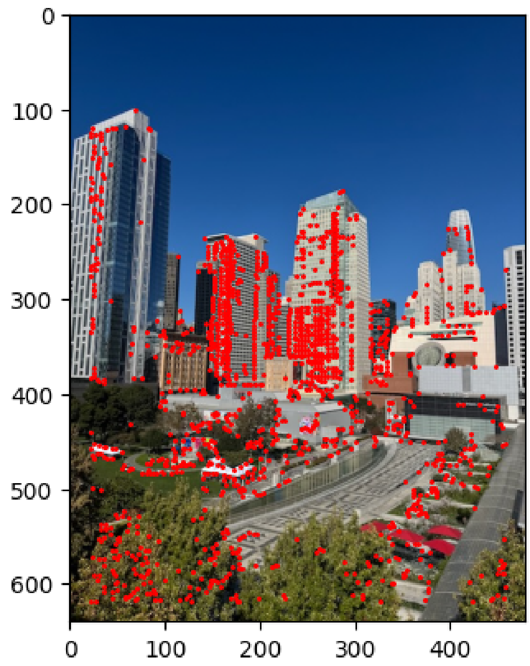


Image with Top 500 Harris Corners

Harris Response Matrix

Adaptive Non-Maximal Suppression (ANMS)

While the Harris Corner Detector can find many interest points, they are often clustered in certain areas, leading to an uneven spatial distribution. Adaptive Non-Maximal Suppression (ANMS) is used to select a subset of corners that are both strong and well-distributed across the image.

The idea behind ANMS is to suppress corners that are not significantly stronger than their neighbors within a certain radius. For each corner point x_i with corner strength f_i, we compute the suppression radius r_i defined as:

$$r_i = \min_{x_j} \{ \|x_i - x_j\| \mid f_j > c \cdot f_i \}$$

where:

- x_j are other corner points.
- f_j is the corner strength of x_j.
- c is a constant slightly less than 1 (e.g., 0.9), ensuring that only significantly stronger corners suppress a point.

After computing r_i for all corners, we sort them in descending order of r_i and select the top N corners with the largest suppression radii. This results in a set of corners that are both strong and evenly distributed across the image.

Here are some of the corners selected after applying ANMS:

Table of Contents

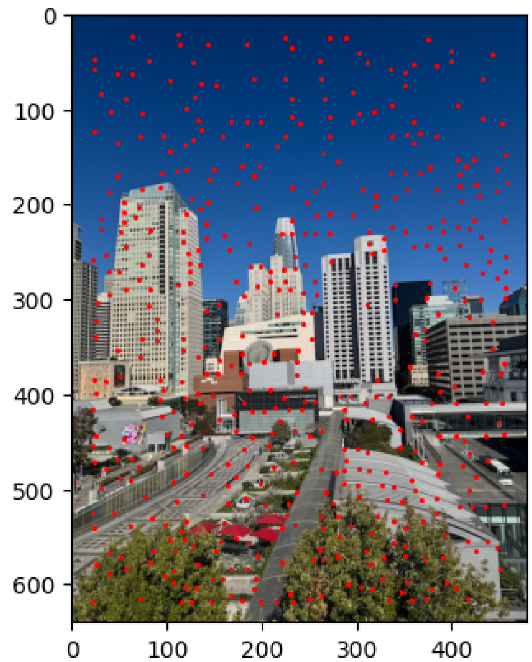


Image with Top 500 ANMS Corners

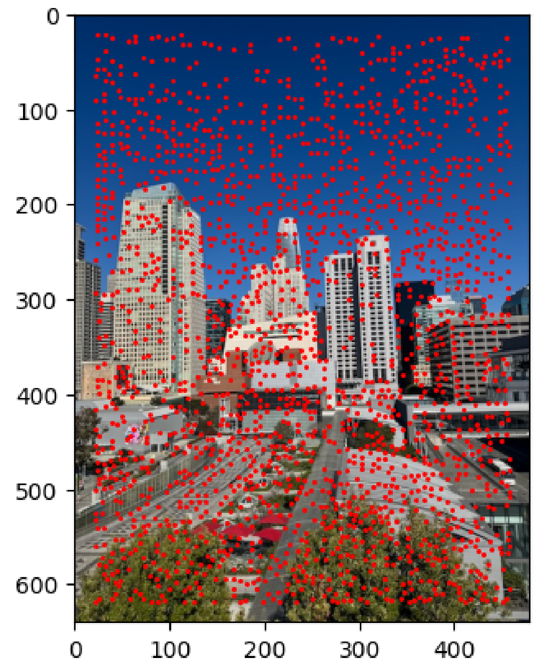
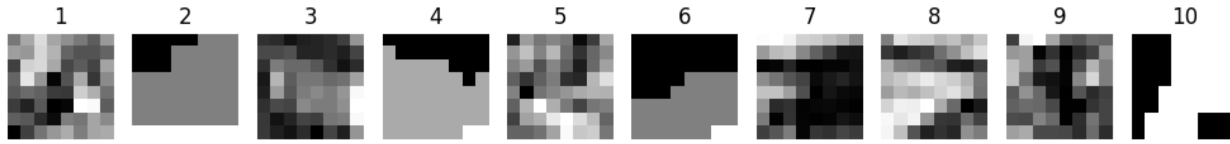


Image with Top 2000 ANMS Corners

Feature Descriptor Extraction

Feature descriptor extraction is a technique to extract features from an image. By extracting feature descriptors, we can capture the local appearance around each keypoint in a way that is robust to changes in illumination and minor geometric distortions. In this project, we:

- Extracted a **40x40 pixel patch** around each keypoint.
- Applied a **Gaussian blur** to reduce high-frequency noise and aliasing.
- **Downsampled** the patch to an **8x8 descriptor** to reduce dimensionality.
- **Normalized** each descriptor to have zero mean and unit variance, ensuring that descriptors are comparable across different images.
These steps produce compact and discriminative feature descriptors that can be reliably matched between images. Here are some of the features I got from this technique (not normalized for display).



Feature Descriptors (10 of them)

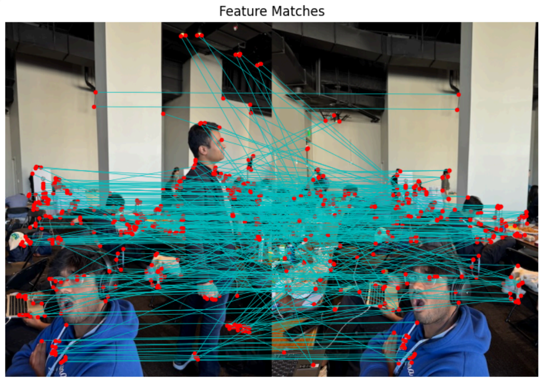
Feature Matching

In order to match features between multiple images, we use the previously calculated descriptors. By comparing the feature descriptors from one image to those from another, we can identify corresponding keypoints. We compute the **Euclidean distance** between descriptors—the smaller the distance, the more similar the descriptors are.

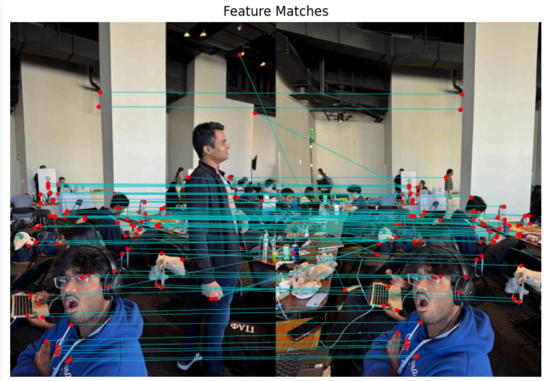
We can also limit the number of false matches by using Lowe's trick. This is known as **Lowe's ratio test**, which involves comparing the distance of the closest neighbor to that of the second-closest neighbor for each descriptor. If the ratio of these distances is below a certain threshold (e.g., 0.7), we consider the match to be reliable. This helps eliminate ambiguous matches where a descriptor is similarly close to multiple descriptors.

By applying Lowe's ratio test, we reduce the number of false positives and retain only the most distinctive matches. This enhances the robustness of our feature matching process. Here are some of the matches I got from this technique.

Table of Contents



Matches with no threshold

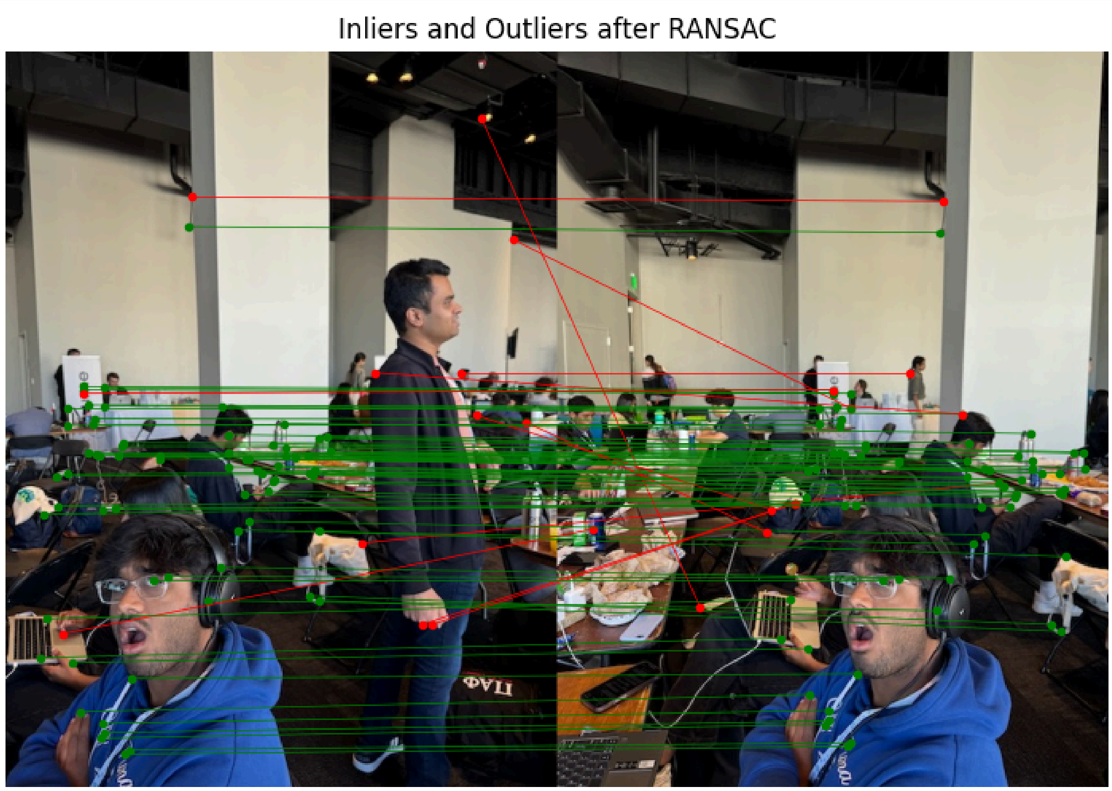


Matches with Lowe's threshold t = 0.7

RANSAC

RANSAC is a technique to remove outliers from a set of matches. RANSAC stands for **Random Sample Consensus**. It's an iterative method used to estimate a mathematical model (in this case, the homography) from data that contains outliers. The algorithm works by:

- Randomly selecting a minimal subset of correspondences (e.g., 4 pairs) to estimate a candidate homography.
- Applying this homography to all correspondences to determine how many are inliers (matches that fit the model within a certain error threshold).
- Repeating the process for a number of iterations and selecting the homography with the highest number of inliers.
This approach allows us to robustly estimate the homography by considering only the inliers and discarding outliers caused by incorrect matches. Here is an example of the matches I got from this technique.



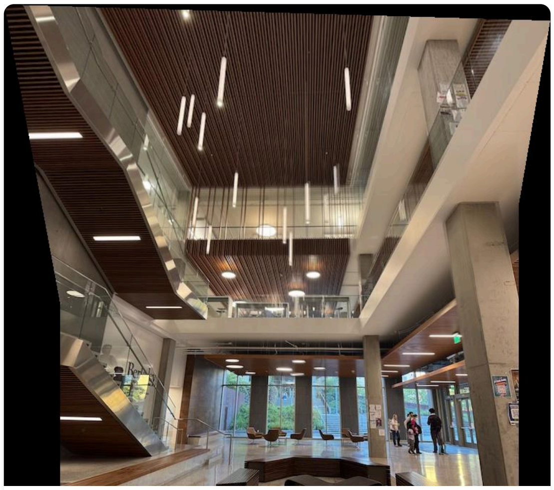
RANSAC

Final Results

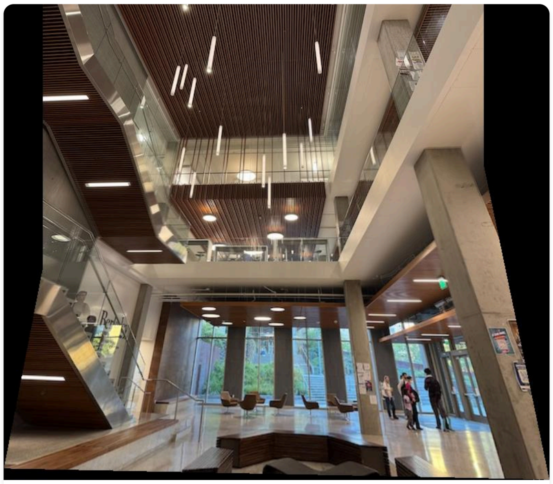
Here are some of the final results I got from this technique.

Differences are slight, but visible if you look closely.

Table of Contents



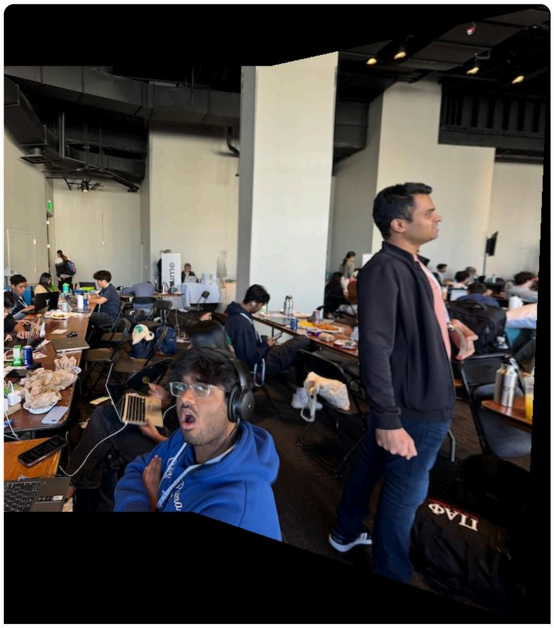
Automatic Correspondences



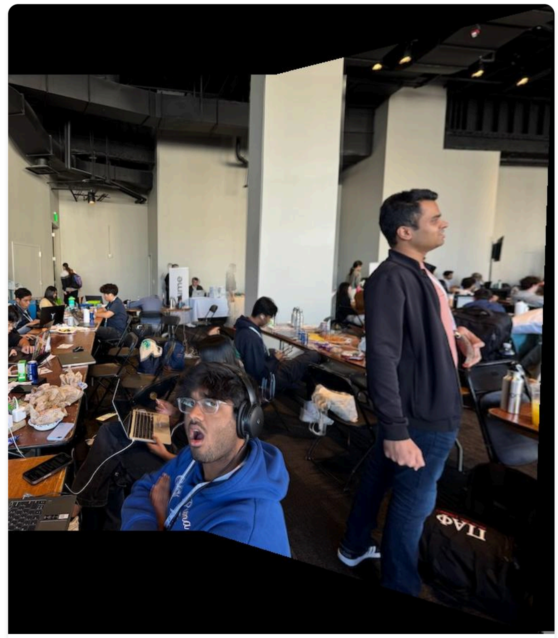
Manual Correspondences

Automatic Correspondences

Manual Correspondences



Automatic Correspondences



Manual Correspondences