Image Mosaics and Panorama Stitching

github.com/aditimundra05/cs180/4/index.html

Part A.1: Shoot the Pictures

In order to create panoramic mosaics, I took pictures of a building interior and landscape exterior. I fixed the center of projection and rotated my camera while capturing the photos, kept 40% of overlap between the images, and shot in a short time span.

Image Set 1: Outdoor Landscape Panorama

Landscape 1

Landscape 2

Landscape 3

Landscape 4

Image Set 2: Indoor Interior Panorama

Inside 1

Inside 2

Inside 3 *Inside 4*

Part A.2: Recover Homographies

A homography is a projective transformation that maps points from one image plane to another. Mathematically, it relates corresponding points $\mathbf{p}$ and $\mathbf{p}'$ between two images as:

$$\mathbf{p}' = H\mathbf{p}$$

where H is a 3×3 matrix with 8 degrees of freedom (the lower-right element is set to 1 for normalization).

Procedure

First, I used the tool given to find point correspondences between the two images. Given corresponding points in homogeneous coordinates:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

and the homography matrix:

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix}$$

The transformation yields:

$$H\mathbf{p} = \begin{bmatrix} xh_1 + yh_2 + h_3 \\ xh_4 + yh_5 + h_6 \\ xh_7 + yh_8 + 1 \end{bmatrix}$$

Then, I normalized using the z component to come back to the original (x, y) plane, and obtained:

$$\begin{cases} \frac{xh_1+yh_2+h_3}{xh_7+yh_8+1} = x' \\ \\ \frac{xh_4+yh_5+h_6}{xh_7+yh_8+1} = y' \end{cases}$$

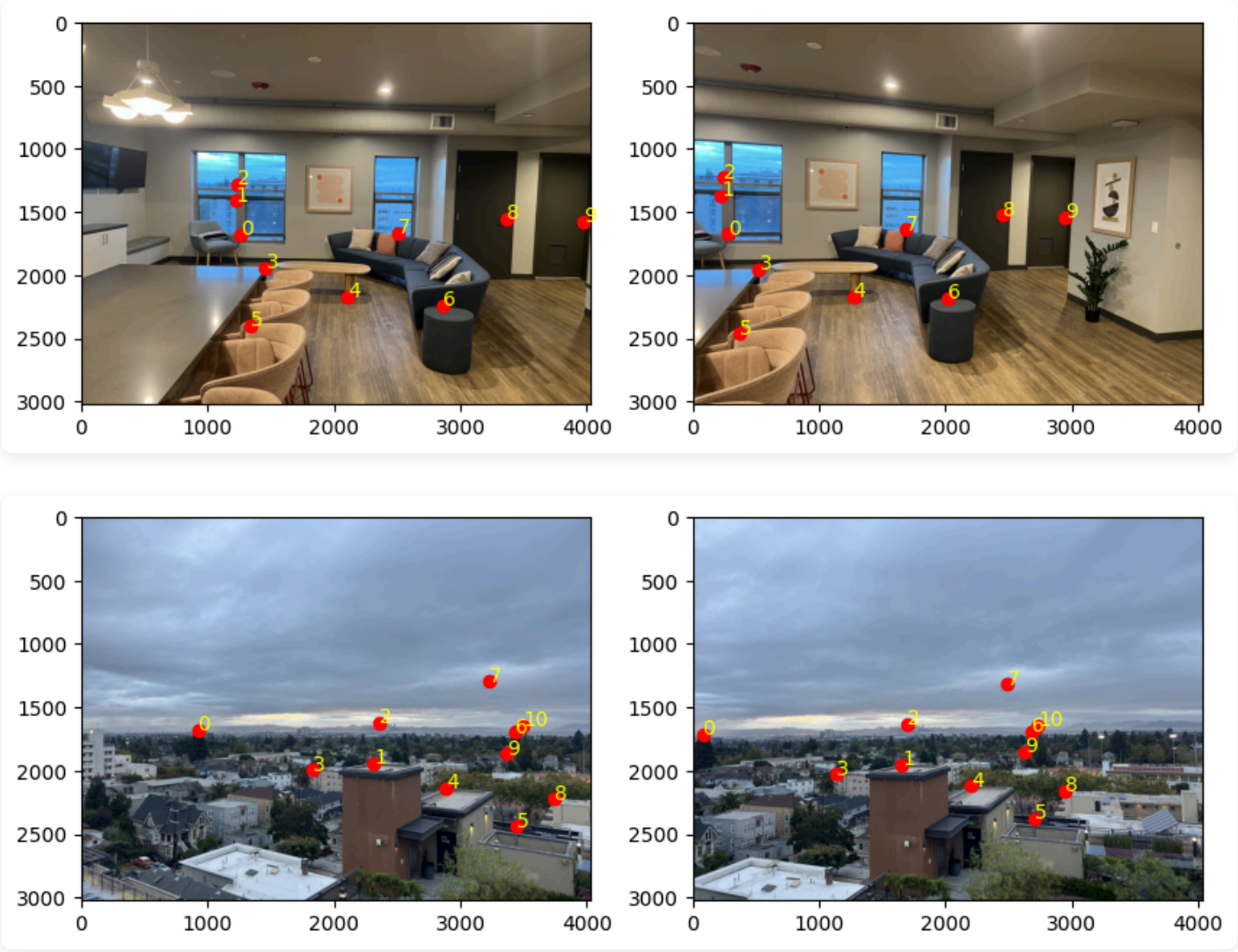
As per discussion, I learned that we can rearrange these equations to get two linear constraints per point correspondence:

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

For n point correspondences, we construct an overdetermined system $A\mathbf{h} = \mathbf{b}$ where A is a $2n \times 8$ matrix, and solve using least squares: $\mathbf{h} = (A^T A)^{-1} A^T \mathbf{b}$.

Point Correspondences

Here are my point correspondences for the first pair of interior images and exterior images.



Visualized point correspondences between two images

Recovered Homography Matrix

Using the point correspondences shown above, the computed homography matrix is:

```
H_interior = [[ 1.51771976e+00 -1.13667471e-04 -1.59786792e+03]
 [ 1.89873898e-01  1.28438160e+00 -4.61985366e+02]
 [ 1.27445928e-04 -2.76085941e-06  1.00000000e+00]]
```

```
H_landscape = [[ 1.33437925e+00 -4.47218776e-02 -1.07262312e+03]
 [ 1.40441962e-01  1.13252177e+00 -2.39534564e+02]
 [ 9.44219893e-05 -2.51788604e-05  1.00000000e+00]]
```

Part A.3: Warp the Images

Once the homography is recovered, we can warp images to align them. I implemented two interpolation methods using inverse warping to avoid holes in the output:

Inverse Warping

Rather than mapping source pixels to destination (forward warping), which can leave holes since pixels can hit some of the same points during their transformation while leaving others untouched (essentially no pixels map to that place), I used inverse warping. For each pixel in the output image, I computed its corresponding location in the source image using H^{-1} and interpolated the color value. First, I found the location of the destination image by applying H to the corners of my source image. From this bounding box, I went coordinate by coordinate and applied H_inverse to find the source location. Here, I applied the respective interpolation methods detailed below.

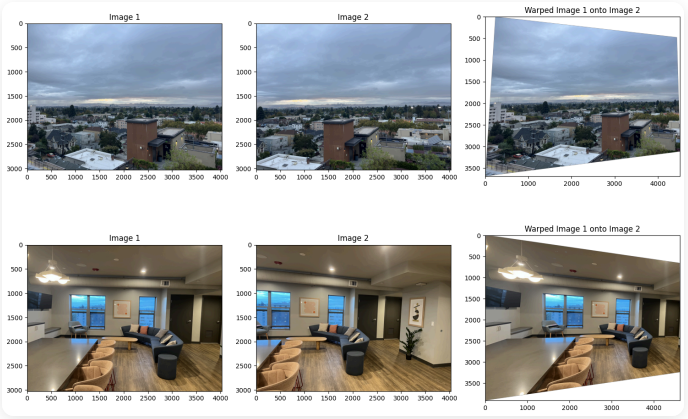
Interpolation Methods

> **1. Nearest Neighbor Interpolation**
>
> In this method, I rounded the computed coordinates to the nearest integer pixel location and used that pixel's value directly.
>
> > **Comparison:** This method was faster (less time to compute) since very little algebra was involved and there was no blurring in my results. However, though it's not visible in my images, it can produce block-like results since there's no gradual transition or incorporation of other pixel values.

> **2. Bilinear Interpolation**
>
> This method computes a weighted average of the four nearest pixels based on their distance from the query point.
>
> > **Comparison:** This method had smoother results since other pixel values were incorporated which resulted in better visual quality of my images. However, it took

much longer since there are more computations involved and there was slight blurring.

Quality Comparison



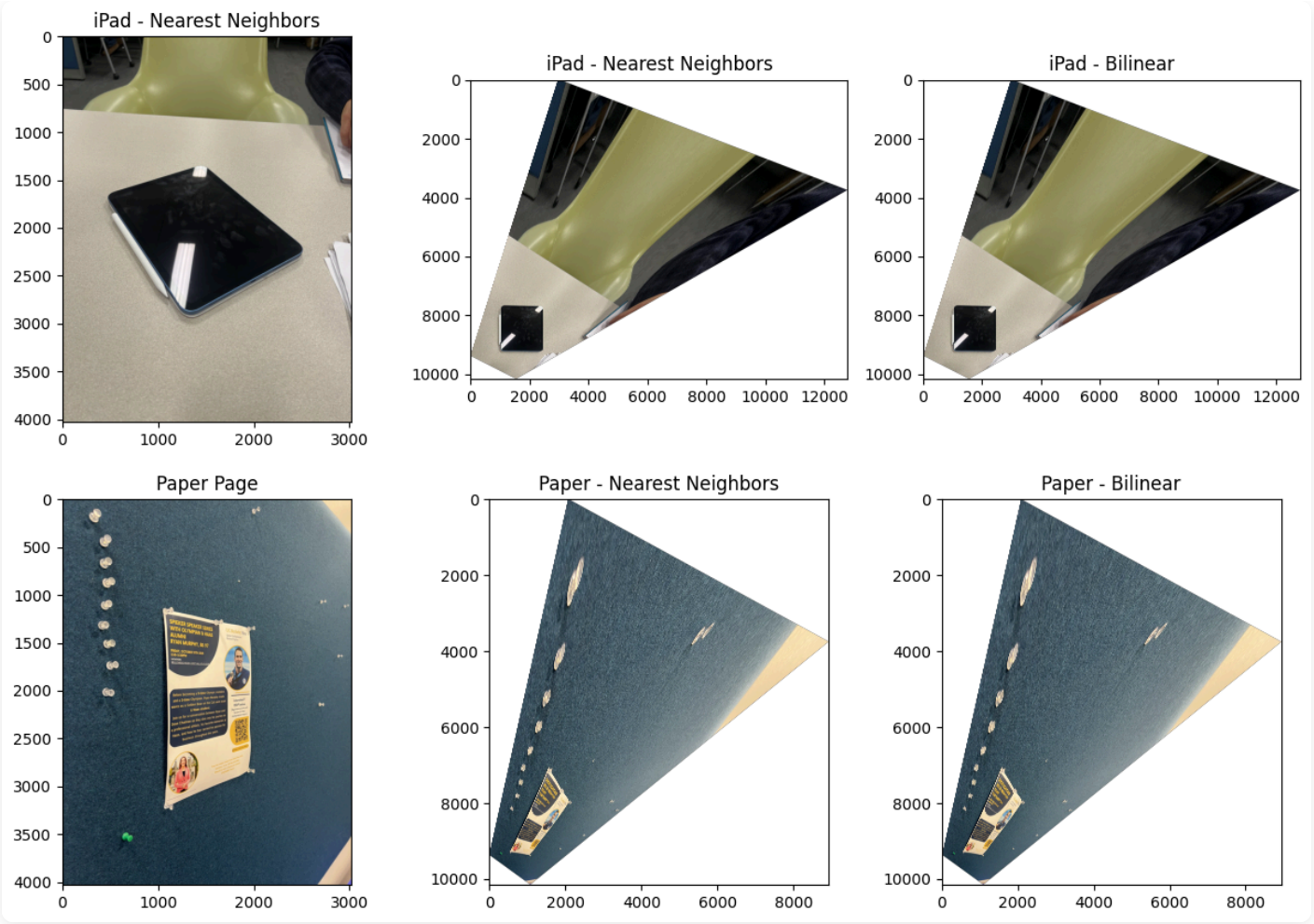
Nearest Neighbor Interpolation

Bilinear Interpolation

As shown in the comparison above, bilinear interpolation produces smoother results, particularly in areas with fine details and edges. The nearest neighbor method preserves sharp edges better.

Rectification

To verify my homography and warping functions were working correctly, I performed rectification on images with rectangular objects (an iPad and a piece of paper). I manually defined correspondences between the rectangles and a standard square, so the homography transformed the distorted objects back to their rectangular shapes.



Part A.4: Blend the Images into a Mosaic

Here, I blended the images into a mosaic by warping the images one on top of the other and extending the dimensions accordingly. Also, I added simple feathering at every channel using the alpha channel by setting it to 1 at the center of each image and making it fall off until 0 at the edges.

Blending Procedure

1. Compute the homography from image 2 to image 1 (src -> dest).
2. Warp image 2 (and beyond) into image 1 using warpImageBilinear() and include the alpha channel as detailed above (light feathering).
3. Adjust the bounding box / dimensions / offsets to account for the new image size (find the new min/max x/y values, adjust the offset, expand the height and width of the image).

4. Blend the images together using the alpha channel values (at each pixel, compute the final color as a weighted average of all contributing images using their alpha masks as weights).

Mosaic Results

> **Mosaic 1:** Lecture Hall Interior

Lecture Hall Image 1

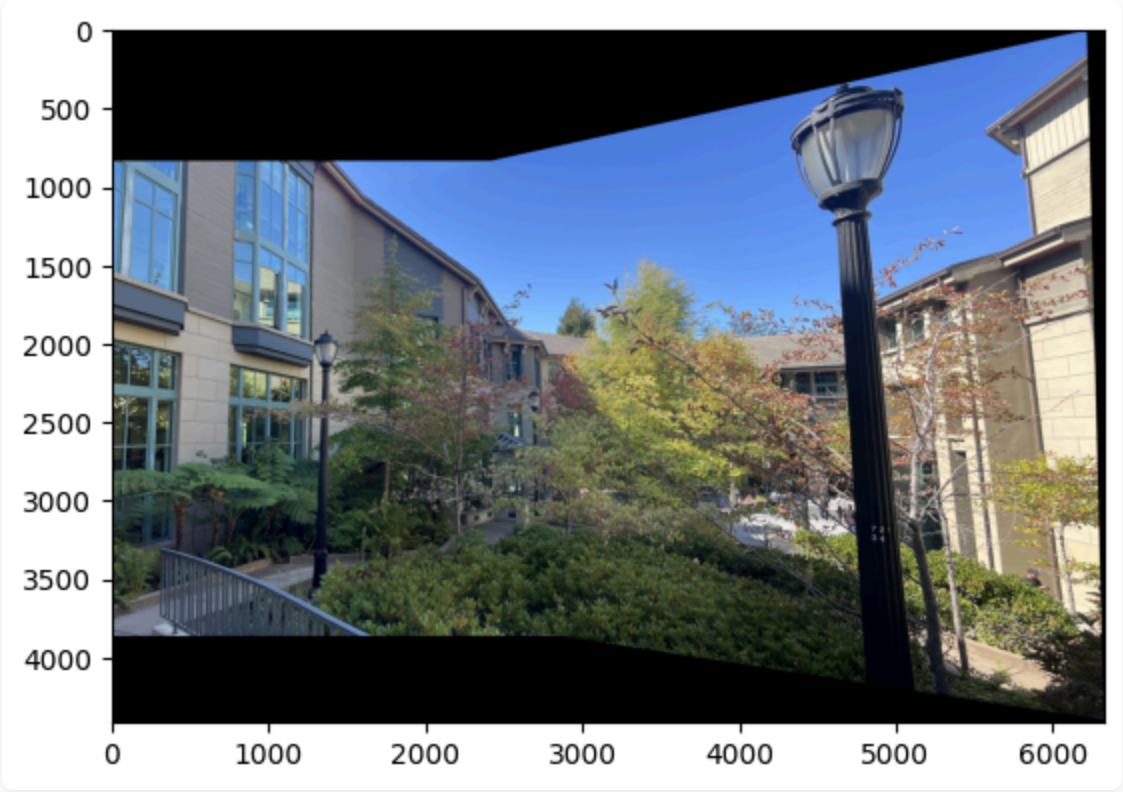
Lecture Hall Image 2

Panoramo of Lecture Hall (Note: There is some distortion of the men since they were walking during the picture. However, from the text shown on the screen, the warping was accurately done. The men moving adds a time dimension to my image.)

Mosaic 2: Landscape Panorama

Library Image 1

Library Image 2

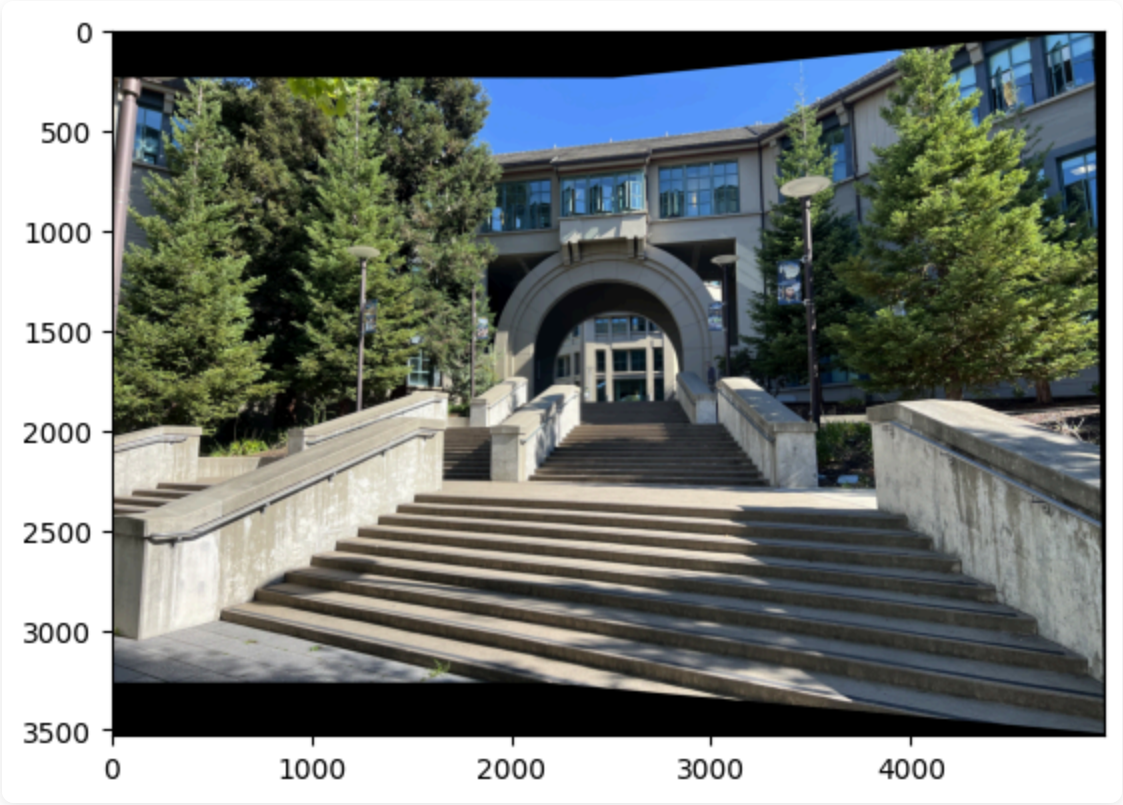


Library Panorama

Mosaic 3: Haas Arch Panorama

Haas Image 1

Haas Image 2



Haas Arch Panoramo