Project 3: Panorama Stitching

In this project, I explore homographies, warping, and blending to stitch together images in a panorama.

Part 1: Manual Stitching

A.1: Collecting Images

I shoot two or more photographs that differ from each other via a projective transformation. I fixed the Center of Projection (COP) and rotated my camera while shooting.

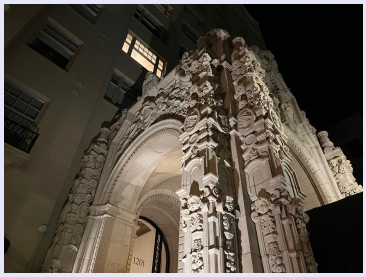
Hearst Mining Memorial Building, Floor 3

Left

Right

A San Francisco building, near Grace Cathedral

Straight forward



Tilted up

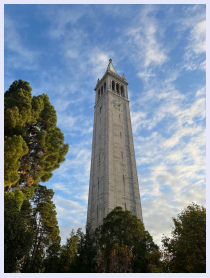
The T. Rex in the Valley Life Sciences Building

Straight forward

Tilted up

The Campanile

Straight forward



Tilted up

A.2: Recovering Homographies

I manually select correspondences on an image set, and recover the homography matrix h.
2d homgraphies follow:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

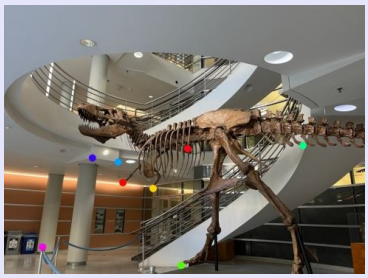
where lambda is a scalar and H is a 3x3 matrix:

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

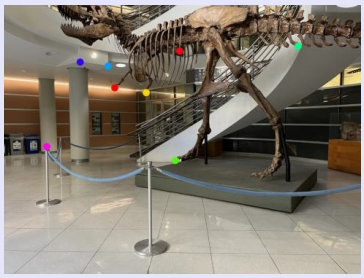
We can recover the matrix H by solving an OLS problem of the form Ah = b, where h is the flattened H matrix. Each point yields two equations, of the form:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -v_1y_1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \dots \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \dots \end{bmatrix}$$

The colored circles are the point correspondences for the T.Rex image set. The system is overdefined: 4 correspondences are required to get a solution, but I provide 8 to reduce noise and improve stability.



Correspondences on T.Rex shot straight forward



Correspondences on T.Rex shot tilted up

The system of equations is as follows. If Ah = b,

$$\begin{bmatrix} 403.578 & 292.331 & 1.0 & 0.0 & 0.0 & 0.0 & -170861.985 & -123763.763 \\ 0.0 & 0.0 & 0.0 & 403.578099 & 292.331522 & 1.0 & -256034.470 & -185458.395 \\ 513.835189 & 310.707704 & 1.0 & 0.0 & 0.0 & 0.0 & -273468.930 & -165362.173 \\ 0.0 & 0.0 & 0.0 & 513.835189 & 310.707704 & 1.0 & -334698.800 & -202386.870 \\ 618.438069 & 551.011618 & 1.0 & 0.0 & 0.0 & 0.0 & -389459.202 & -346997.631 \\ 0.0 & 0.0 & 0.0 & 618.438069 & 551.011618 & 1.0 & -570679.750 & -508460.246 \\ 1055.22577 & 143.908517 & 1.0 & 0.0 & 0.0 & 0.0 & -1113501.43 & -151855.975 \\ 0.0 & 0.0 & 0.0 & 1055.22577 & 143.908517 & 1.0 & -527743.449 & -71972.0641 \\ 358.344421 & 124.118783 & 1.0 & 0.0 & 0.0 & 0.0 & -142087.267 & -49214.3803 \\ 0.0 & 0.0 & 0.0 & 358.344421 & 124.118783 & 1.0 & -171111.930 & -59267.5740 \\ 689.115690 & 204.691271 & 1.0 & 0.0 & 0.0 & 0.0 & -48656.9649 & -144528.069 \\ 0.0 & 0.0 & 0.0 & 689.115690 & 2.0.4691271 & 1.0 & -379710.751 & -112787.269 \\ 378.134155 & 214.586138 & 1.0 & 0.0 & 0.0 & 0.0 & -153675.688 & -87208.9233 \\ 0.0 & 0.0 & 0.0 & 378.134155 & 214.586138 & 1.0 & -211563.387 & -120059.428 \\ 376.720603 & 169.352460 & 1.0 & 0.0 & 0.0 & 0.0 & -154166.242 & -69304.4983 \\ 0.0 & 0.0 & 0.0 & 376.720603 & 169.352460 & 1.0 & -194797.085 & -87569.8472 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} = \begin{bmatrix} 423.36783285 \\ 634.41121124 \\ 532.21137026 \\ 651.37384045 \\ 629.74648821 \\ 922.77590777 \\ 1055.2257708 \\ 500.12373001 \\ 396.51033661 \\ 477.50689107 \\ 706.07831964 \\ 551.01161763 \\ 406.40520364 \\ 559.49293224 \\ 409.23230851 \\ 517.08635922 \end{bmatrix}$$

The recovered H matrix is:

$$H = \begin{bmatrix} 0.871993633 & -0.241168013 & 94.5905696 \\ -0.00314957574 & 0.656993197 & 372.706782 \\ -0.0000195512678 & -0.000351052095 & 1 \end{bmatrix}.$$

A.3: Warping

I manually warp the image using both nearest-neighbor and bilinear interpolation.

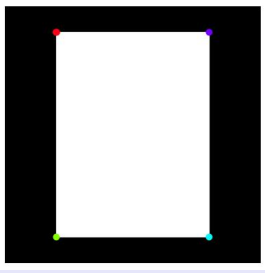
To warp the image, I use the following procedure:

1. Define the corners of the unwarped image as (0,0,1), (width-1, 0,1), (0, height-1, 1), (width-1, height-1, 1)
2. Transform each corner according to H: H * c. These corners define the bounding box for the final image, so I then find the size of the rectangular box that contains these coordinates. This rectangular box represents the destination image.
3. Iterate over every coordinate pair (i,j) in the destination image. For each coordinate,
    1. Translate the coordinate by the minimum x and y values (found when calculating the bounding box) to ensure it is within the correct coordinate frame
    2. Inverse-map the coordinate to the source image by calculating H_inverse * (i,j,1)
    3. Interpolate the value and assign it to the destination image.

For nearest_neighbor interpolation, I round the inverse mapped coordinate to the nearest integer point and use that value in the destination image. For bilinear interpolation, I find the nearest four coordinates (if the coordinate is on the edge, not all four neighbors will be valid coordinates: I remove those and re-adjust the weighting accordingly). The four nearest neighbors are combined in a weighted average.

Here are the results of rectifying three images.

Original guitar image, with correspondences plotted on top.

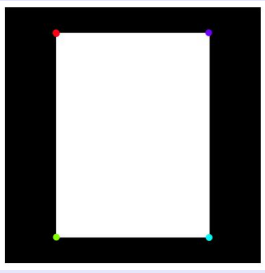


Binary rectangle image, with correspondences plotted on top.

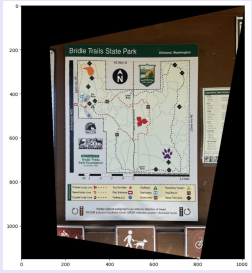
Rectified guitar. Nearest-neighbor interpolation.

Rectified guitar. Bilinear interpolation.

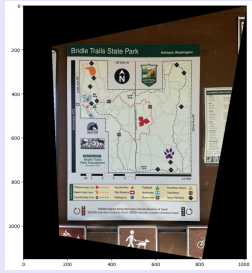
Original image (Bridal Trails trail map), with correspondences plotted on top.



Binary Bridal Trails map image, with correspondences plotted on top.

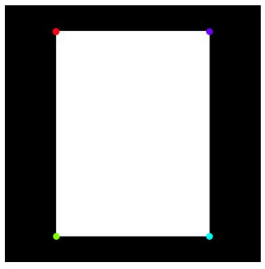


Rectified Bridal Trails map. Nearest-neighbor interpolation.

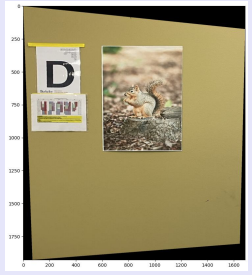


Rectified Bridal Trails map. Bilinear interpolation.

Original image (squirrel poster in Earl Warren Hall), with correspondences plotted on top.



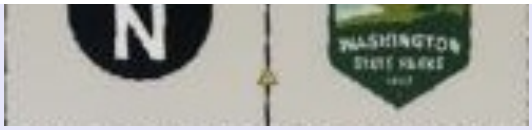
Binary rectangle image, with correspondences plotted on top.



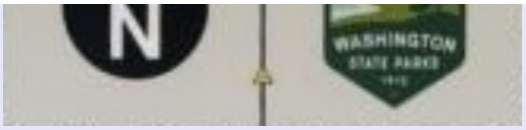
Rectified squirrel. Nearest-neighbor interpolation.

Rectified squirrel. Bilinear interpolation.

Nearest-neighbor interpolation ran quicker than biliear interpolation. However, bilinear interpolation generated higher-quality images. This is most obvious on the Bridal Trails image, where the text is much cleaner. The nearest-neighbor interpolation for the map resulted in jagged edges, which for small words make the text very difficult to read.



The nearest neighbor interpolation, which results in low-quality text and edges.

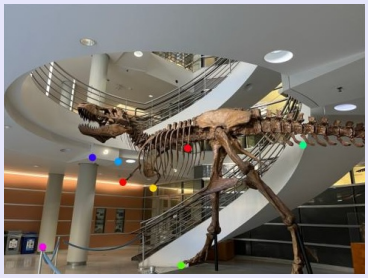


The bilinear interpolation. Although these sections are extremely zoomed in, the text and edges are cleaner.

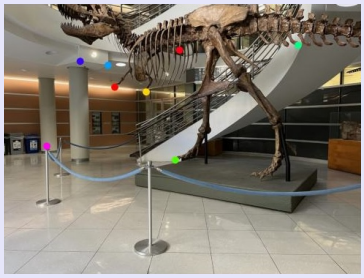
A.4: Blending

After warping the image, I align and blend the images into a mosaic. I use the following procedure:

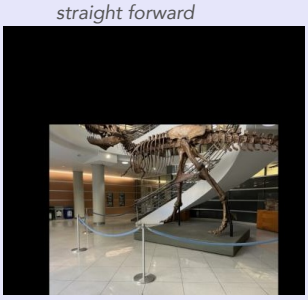
1. Determine the size of the final mosaic. This is calculated from the extents of the fixed and warped image.
2. Place the fixed and warped image onto an all-zero image of the same size as the final mosaic. At this point, I have both the fixed and warped image aligned on their own individual mosaic-sized backdrops.
3. I blend the warped and fixed images via a two-layer Laplacian pyramid. I use a kernel size on the images of 10, and blur the mask with a kernel size of 30. The standard deviations are the kernel size divided by 6.



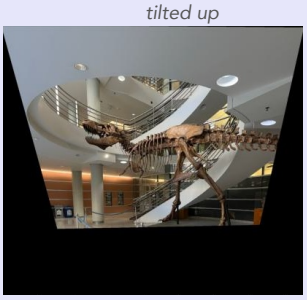
Correspondences for T. Rex straight forward



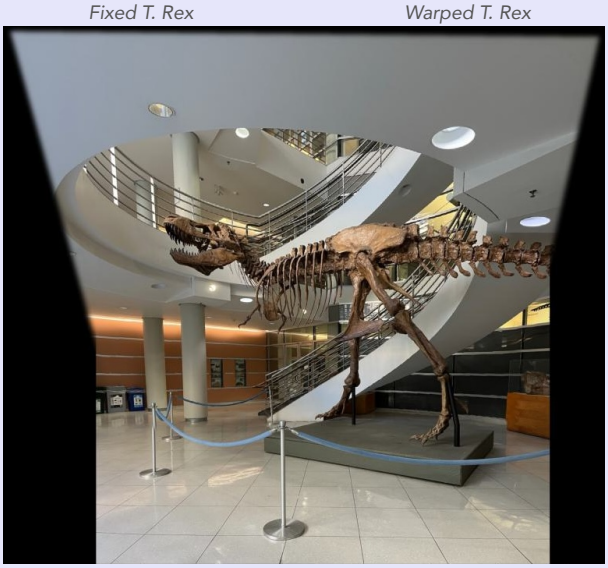
Correspondences for T. Rex tilted up



Fixed T. Rex



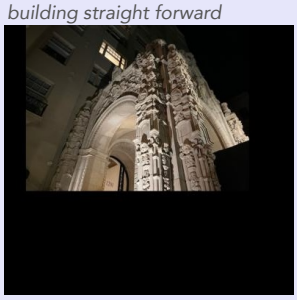
Warped T. Rex



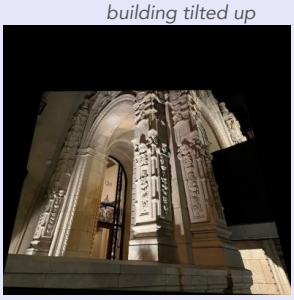
Blended T.Rex

Correspondences for SF building straight forward

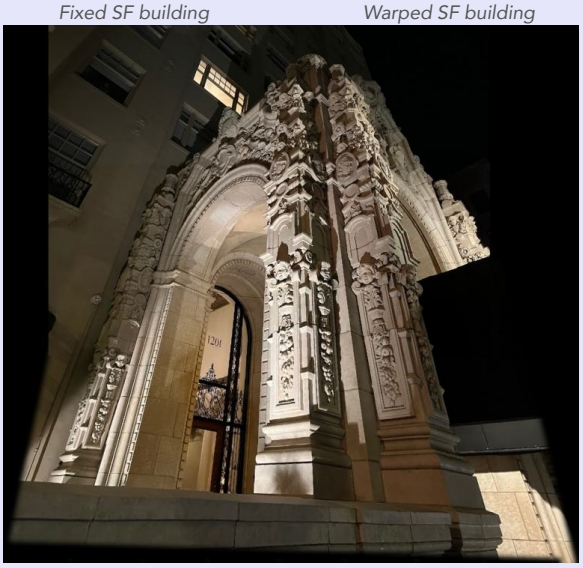
Correspondences for SF building tilted up



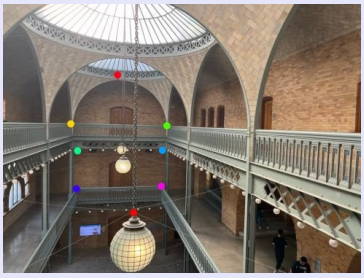
Fixed SF building



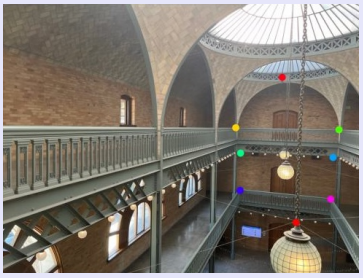
Warped SF building



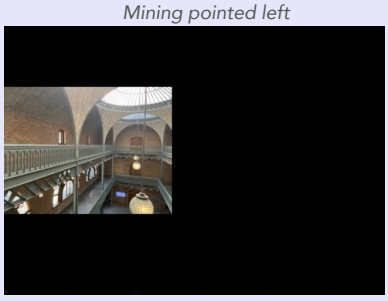
Blended SF building



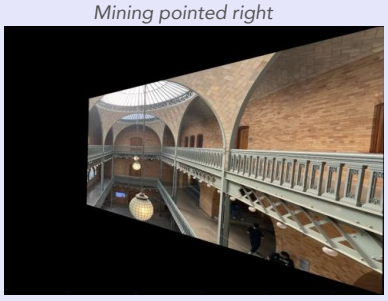
Correspondences for Hearst Mining pointed left



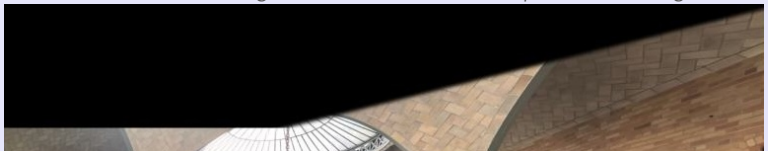
Correspondences for Hearst Mining pointed right

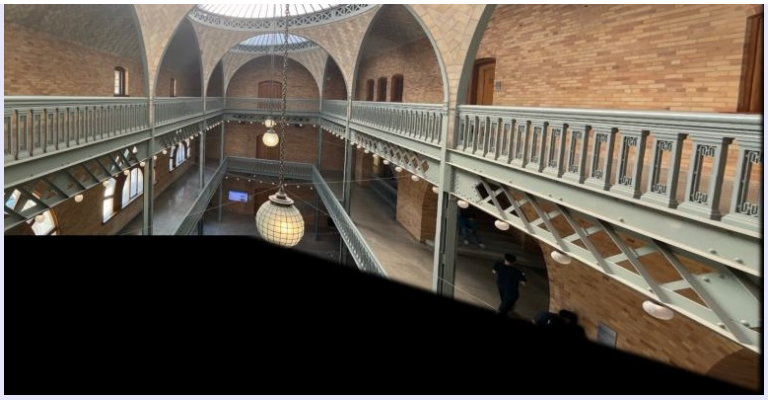


Fixed Hearst Mining

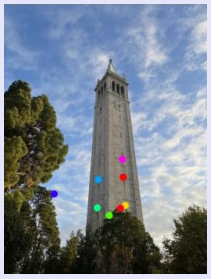


Warped Hearst Mining



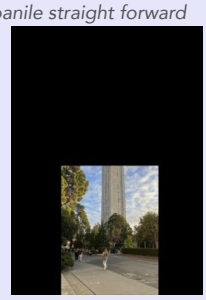


Blended Hearst Mining



Correspondences for the Campanile straight forward

Correspondences for the Campanile tilted up



Fixed Campanile

Warped Campanile

Blended Campanile

A.5: Bells and Whistles: Cylindrical Projection

I project the mosaic onto a cylinder rather than a plane. Although the main procedure for warping is the same, there are a few changes.

1. The inverse-map is now two stages. Previously, I only needed to inverse map the destination image coordinate to the source image via H_inverse. Now, the inverse map is 1) destination coordinate through an inverse cylindrical projection, 2) the inverse

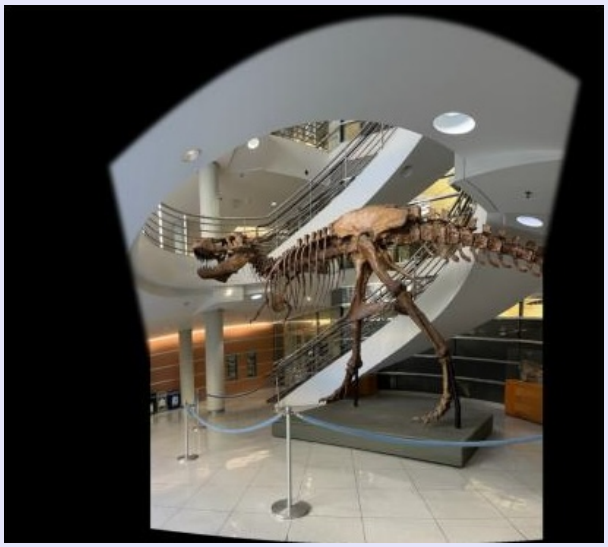
cylindrical projection to the source image via H_inverse.
2. I need to transform the fixed image as well, or the images will not align. I inverse-map the original image using the same inverse cylindrical projection.
3. When performing the inverse map for both images, I shift each coordinate about the center of the final mosaic. This ensures that both images are transformed about the same cylinder.

I use nearest-neighbor interpolation for the speed benefits. I blend the images using the same two-layer Laplacian pyramid as previous.

The inverse cylindrical projection is as follows:

$$\begin{aligned}
\text{center}_y,\ \text{center}_x &= \tfrac{1}{2}i_{\text{full_mosaic_size}},\ \tfrac{1}{2}j_{\text{full_mosaic_size}} \\
x' &= \text{dest_coord}_x - \text{center}_x \\
y' &= \text{dest_coord}_y - \text{center}_y \\
x &= f\tan\left(\frac{x'}{s}\right) \\
y &= \frac{y'}{s}\sqrt{x^2+f^2} \\
\text{inverse_coord} &= [\,x+\text{center}_x,\ y+\text{center}_y,\ 1\,]
\end{aligned}$$

I chose to set s = f to minimize distortion. I also chose an f value of 1000 based on aesthetics.



T.Rex cylinder projection

Hey, thanks for reaching the bottom :)