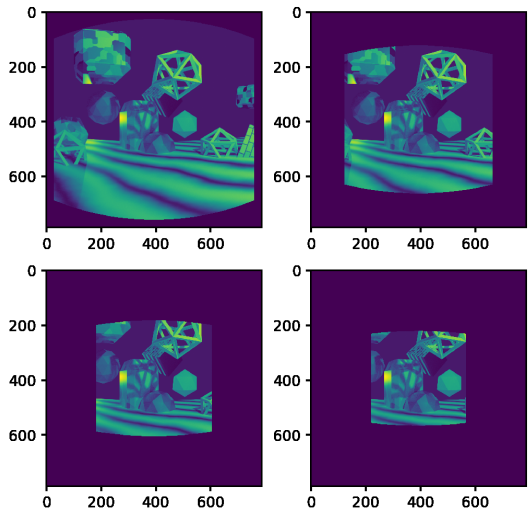
P4 Panorama Stereo

Benjamin Geyer
G00997436

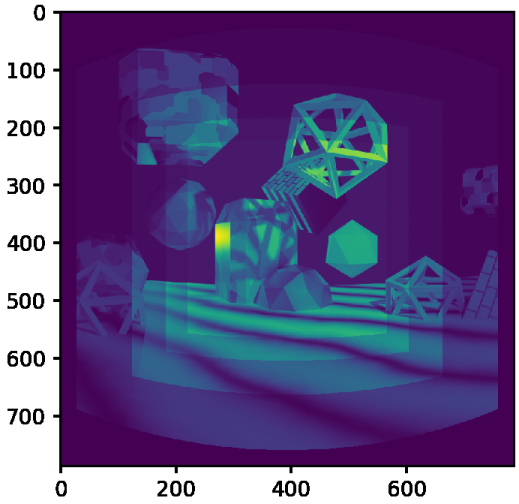
P4.1 Panorama Spherical Reprojection

TASK Generate 4 images by changing the Focal Length of the camera in Blender and name them as follows:
Plots Run the Evaluation and Plotting code I have included below. This will generate three figures (all of which you should include in your writeup). (1) shows the four images after the spherical reprojection. (2) shows the images added together, showing that in the center where all images have visibility of the scene, the images properly overlap. (3) The "differences" between consecutive Focal Lengths; if your code is implemented well, the center region (where the two overlap) should be nearly zero ("white" in the color scheme) and large outside of that image (where they do not overlap).

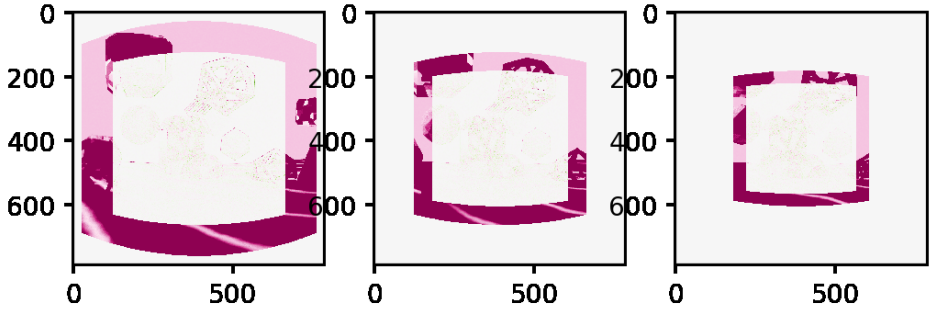
1.



2.



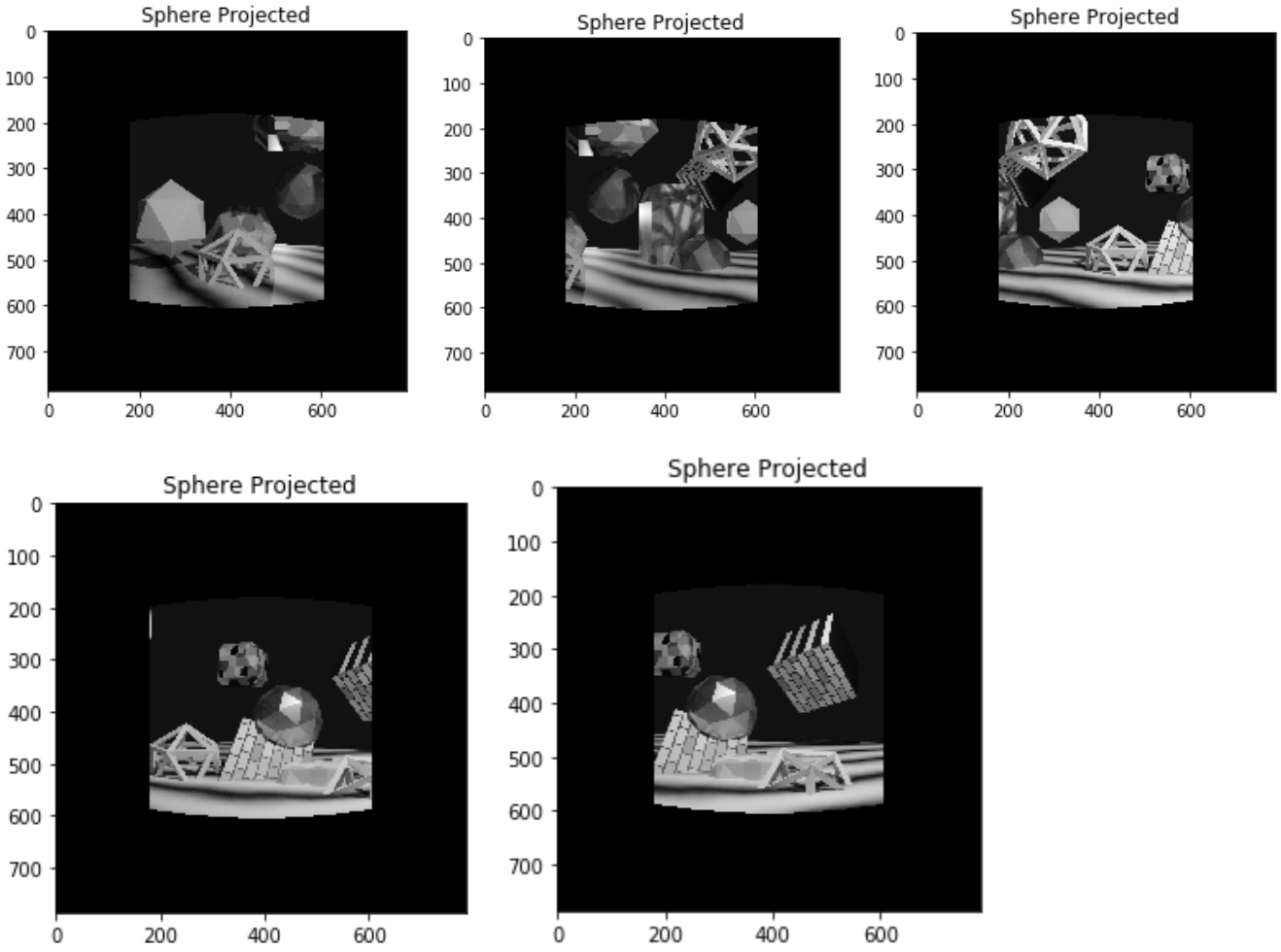
3.



P4.2 Panorama Stitching

TASK Generate images from Blender. To do this, you may use the simple_pano_env.blend environment that I have provided you with. By rotating the camera (done by modifying the rotation about its Z-axis). You should set the Focal length of the camera to 40 mm and sweep the rotation from +40 degrees to -60 degrees; you should rotate the camera in increments such that consecutive images have an overlap of roughly 1/3. You will likely need to generate roughly 5 or 6 images in this range.

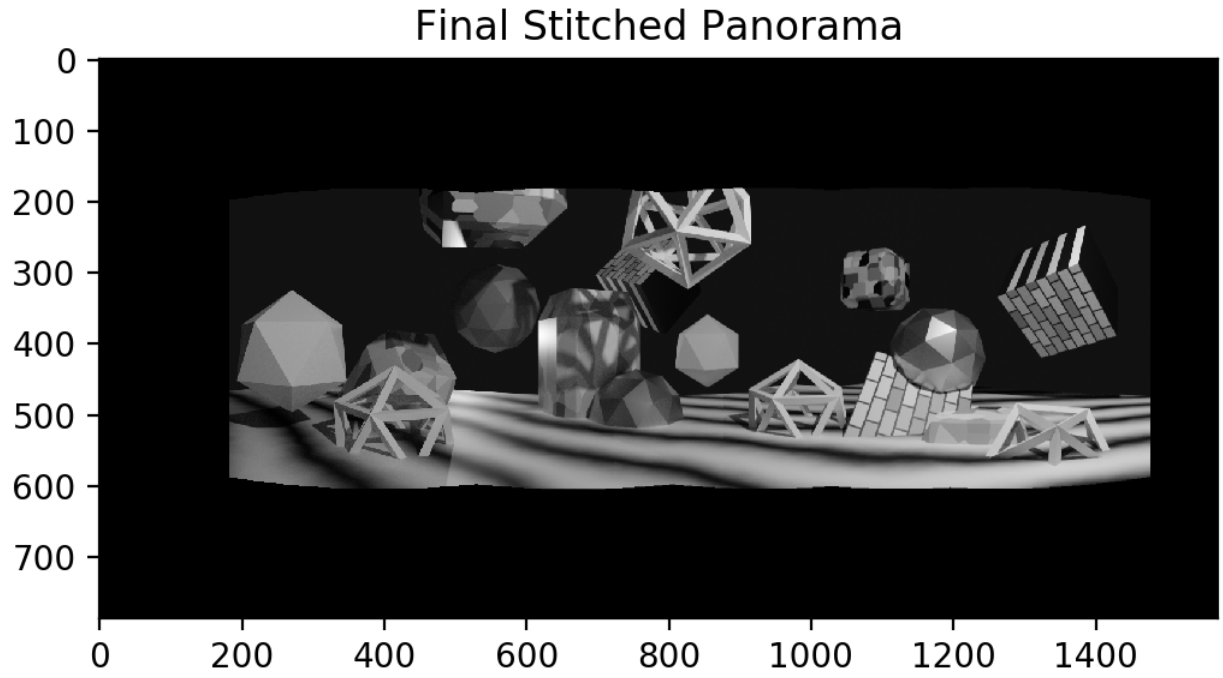
PLOTS Reproject the images using the reproject_image_to_sphere function from the previous question and compute the translation transform between each pair of "consecutive images" (images next to one another in angle space) using OpenCV.



To compute the transformation, you may use the same OpenCV Homography tutorial from the last assignment. However, we know that the transformation is a translation, and so we do not want to allow the system to generate a general homography matrix, which is what results with cv.findHomography. Instead, you should use affine_mat = cv.estimateAffinePartial2D(src_pts, dst_pts)[0], which returns a 2x3 matrix (you will need to convert this to a 3x3 homography by adding a row of [0, 0, 1]) that only

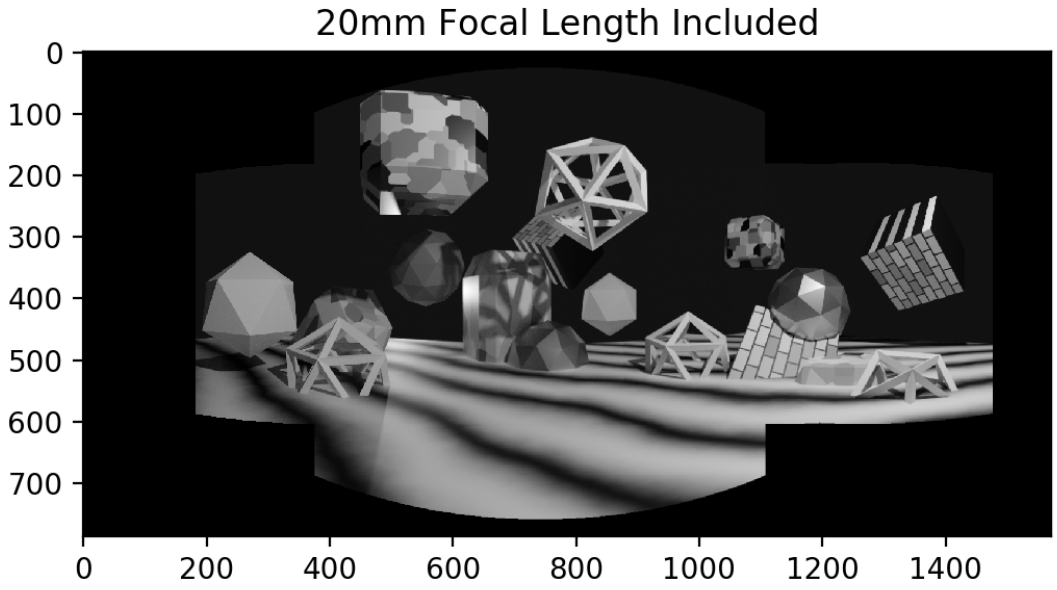
allows for scale, rotation, and translation. Create a new transformation matrix that includes only the estimated translation parameters. Using this procedure should be more numerically stable.

PLOT Create the panorama and include it in a plot!



Note: I am actually quite pleasantly surprised by how that turned out! I can even see the wavy lines in the floor are preserved along with the straight lines of the various objects.

PLOT Finally, add the 20 mm focal length image you generated as part of the previous question to your panorama. It might be interesting to see how the significant change in field of view reveals more of the panorama at once and more of the space above and below the horizon.



This is a very interesting image because the 20mm shot did not match my other shots very well but it surprisingly fit in quite well and gave more context above and below the panorama.

P4.3 Triangulation

P4.3.1 Projecting Into Image Space

Below, I have provided you with two images taken by two cameras a and b. In this question, we will go over some camera basics, namely how to compute the image-space point from a 3D point in the scene and the known camera matrices.

Some information about the two camera matrices:

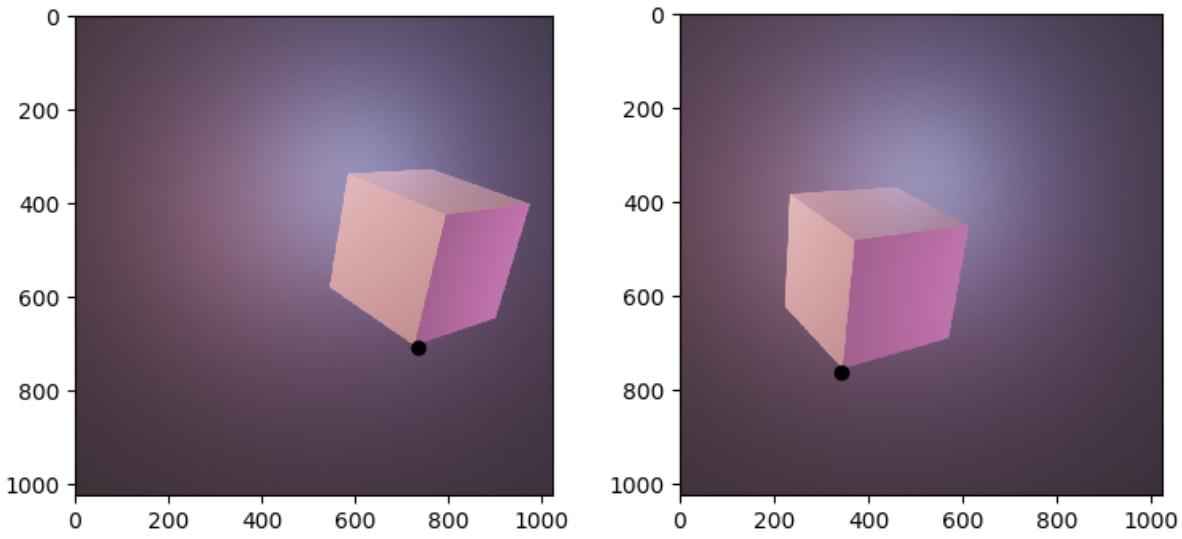
- The first camera is translated such that t_a = [0, -0.2, 5] and t_b = [-1.5, 0, 5]
- No rotation is applied to either camera (so the rotation matrix is the identity matrix)
- The focal length of the camera (for these 1024 px) images is f = 1170.3 (in units of pixels).
- The camera center is located at the center of the image.

QUESTION What are the camera matrices Pa and Pb? I will accept either the final matrix, or the matrix written in terms of its component matrices (the intrinsic and extrinsic matrices), as long as these are defined.

P_a = [[1170.3, 0, 512, 256],
[0, 1170.3, 512, 2327],
[0, 0, 1, 5]]

P_b = [[1170.3, 0, 512, 805],
[0, 1170.3, 512, 256],
[0, 0, 1, 5]]

TASK + PLOTS Implement the function get_projected_point(P, X) which takes in a camera matrix P and a 3D scene point X. If your matrices are implemented correctly, you should see that the projected 3D point overlaps with one of the corners of the cube in image space. Include the two images with the point X0 projected onto the two images.



P4.3.2 Determining the Size of the Cube

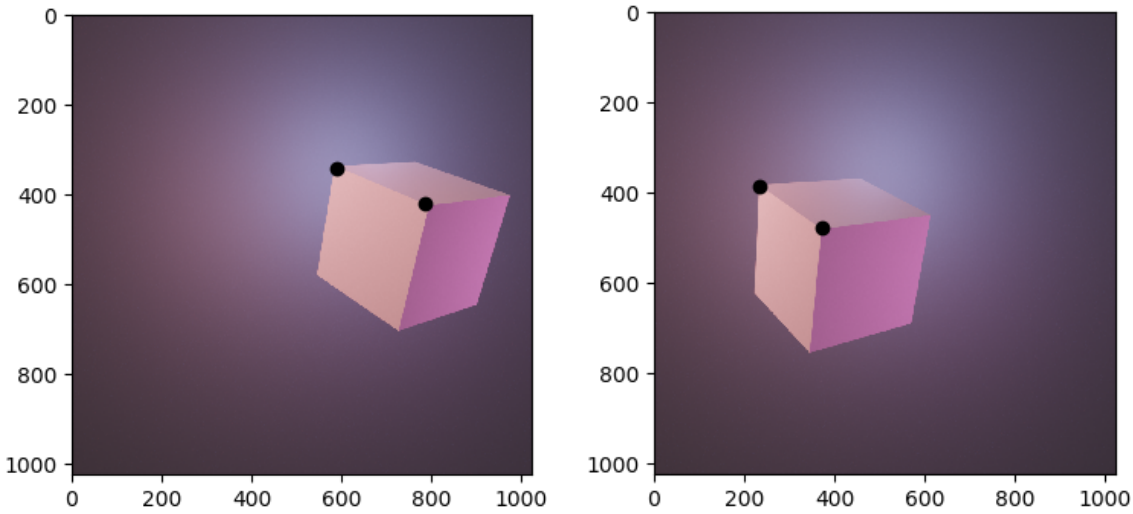
Now you will invert this operation. In class, we discussed how to triangulate a point from two correspondences.

TASK Pick two corners of the cube and include the (x,y) image coordinates for both image_a and image_b and the 3D world coordinate (X,Y,Z) in your writeup.

QUESTION What is the side length of the cube shown in the two images above? (The answer might be somewhat sensitive to the coordinates you measure in image space, though we are only looking for a "close enough" number within maybe 10% of the "correct" answer. You should feel free to use more than two points and average the results to get a more accurate result.)

You can confirm that your estimated 3D coordinates are correct by reprojecting them back into image space using your solution from the previous question to check for accuracy.

The average cube side length between the two images is 0.8395200381207685 (0.9352188842130114, 0.7438211920285256).



I compared the top right corners to the bottom right corner you provided and the distances were very close:
dist_vertical_a: 1.4056291564037884
dist_vertical_b: 1.4866715323221515
dist_vertical_average: 1.44615034436297

P4.4 Stereo Patch Matching

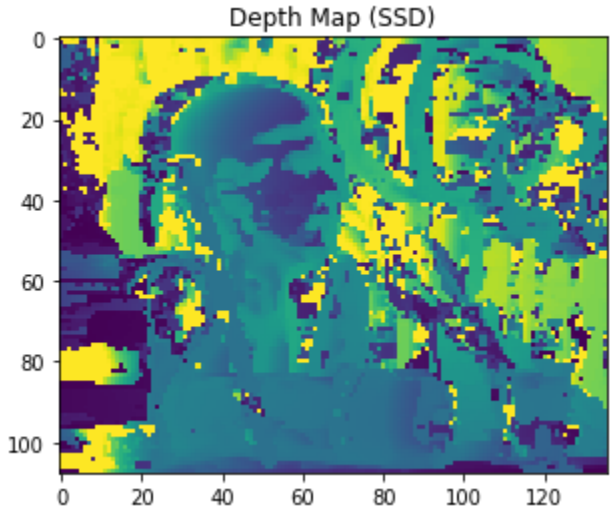
Now I have provided you with a stereo pair of images (already rectified) and a handful of features in one of the images. Your job is to locate the locations of the corresponding features in the other image using patch match stereo as we discussed in class. I have provided you with some starter code in the function patch_match_stereo below, which iterates through the possible locations

QUESTION The possible feature matches in the second image are along the epipolar line. Since the images are properly rectified, what is the epipolar line in the second image corresponding to coordinate (x_a, y_a) in the first image?

The epipolar line is just a horizontal line at height y_a in the second image starting at the half patch width and ending just before half patch width before the right side of the image.

TASK Define the possible_coordinates vector in the patch_match_stereo function using your answer. Once that is defined, the patch_match_stereo function will loop through all possible feature coordinates in the second image and return the coordinate with the best match_score.

TASK Implement the function compute_match_score_ssd (Sum of Squared Differences) using the formula we discussed in class where g is the patch from image_a and f is the patch from image_b. If this function is correctly implemented, you should see some of the features are aligned between the two images.



The NCC Image is not included here but it will generate if you run the notebook. Here is a smaller version applied on just the bottom right corner of the art image: