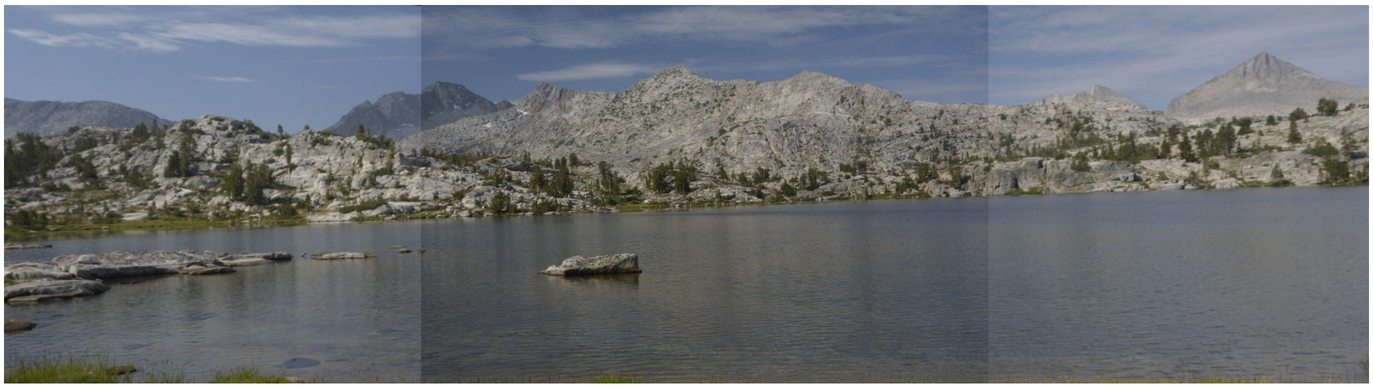
Project 2

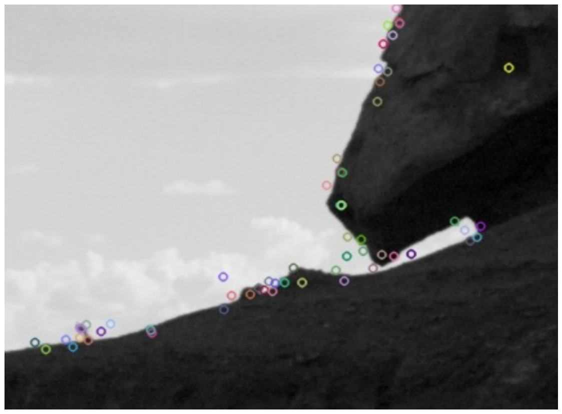
Ethan Armbrust
UBITname: ethanarm
Student Number: 37694242
Undergraduate Student

Image Stitching



The goal of this project was to be able to stitch 3 or less images into a single panorama. To achieve this, my program makes use of the SIFT and RANSAC algorithms to match images together.

My program starts by reading all image files in a given directory, and removing any images that are exact duplicates of other images. Then I use OpenCV to convert all of the images to a gray scale color space. I then begin finding the homography between two images. The program uses OpenCV's implementation of the SIFT algorithm to find key-points in both of the images. For each image, this returns a list of key-points and a list of feature vectors that correspond each key-point.



Key-points shown on the image

To find matching key-points between the two images, my program has a find_best_matches function. This is the most computationally expensive part of the program. This brute force matching function uses the following algorithm:

For each point and feature vector in image 1, calculate the sum of squared differences for every point and feature vector in image 2. The point in image 2 with the smallest SSD value is the best match for that point in image 1.

I then remove every match with an SSD value less than a specific threshold. I found 1150 to work the best. This gives a small list of matching points that have a high chance of being correct matches.

I then perform some checks to make sure that the images are in the correct order by checking that the x values for image 1's points are greater than image 2's x values.

With the small set of matching key-points, I can begin my implementation of the RANSAC algorithm. My RANSAC method accepts 2 key-point lists as input, as well as a threshold. I used the standard library function `itertools` to list all combinations of 4 points for the list of matching key-points. Using the combination of 4 matching key-points from both lists, I used the OpenCV function `getPerspectiveTransform` to calculate the transformation matrix. Then I applied the transformation matrix to all points from the first image to get the calculated points for the second image. I then counted how many of the calculated points were within the distance threshold for the measured points. The transformation matrix that resulted in the most inliers here will be used as the transformation matrix for the image.

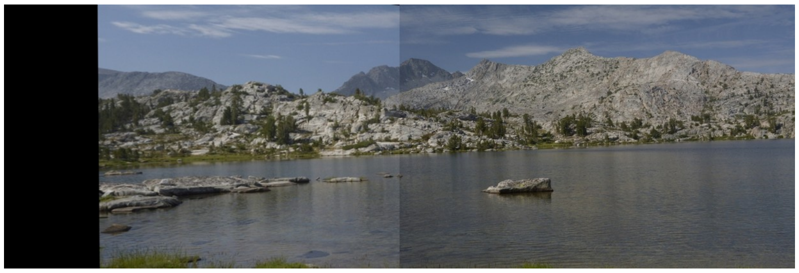
The transformation matrix is then applied to image 2 using the OpenCV function `warpPerspective`. The unaltered image 1 is then placed on top of image 2 to complete the panorama of the two images.

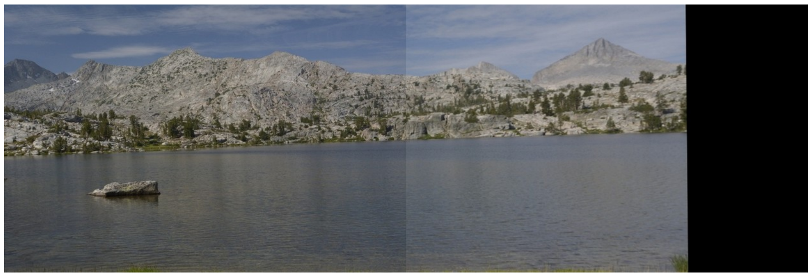
If the image being warped is on the left of the unaltered image, an additional transformation matrix must be applied to the points before the RANSAC matrix is applied.

```
[[1, 0, w]              w: width of the unaltered image.
[ 0, 1, 0]
[ 0, 0, 1]]
```

This offsets the image to the right by the image width to make sure the warped image will fit on the image space.

If only 2 unique images are provided as input, the program will warp the image on the right and finish. If there are three images, the homography must be calculated between all 3 images. The homography with the least amount of matches will be discarded. The center image is determined by the image that exists in both remaining homographies. The left and right images are determined by comparing the x values of the points found by the find_homography function. Then 2 small panoramas of 2 images each are created, with the center image being unaltered in both. Then the two smaller panoramas are placed on top of each other, overlapping with the entire center image.


Left side panorama


Right side panorama