CSCI 1430 Final Project Report: Panorama Image Stitching

Estelle Han, Casey Kim, Anya Li
Brown University
May 6, 2020

Abstract

Panorama image stitching is the process of combining multiple images with overlapping segments into a blended, wide-angle representation of the image. There have been many effective ways of image stitching that involve different methods of feature matching and blending. In this paper, we apply a combination of these methods as well as some of our own to produce a panorama from a dataset of images. Through our project, we have developed our understanding of feature detection and a real-world application of the RANSAC algorithm.

1. Introduction

Constructing panoramas has been of interest starting from a long time ago. From painting panoramas to stitching softwares, techniques to seamlessly blend together images have evolved greatly over time. Now, there are dedicated softwares specifically for stitching images together, minimizing parallax error. Some smartphone cameras can even conduct the stitching internally through the digital camera.

Panoramas are useful for capturing a wide-perspective shot of a space. They are often used to capture landscapes or areas that would be more efficiently describe by one long image rather than separate images. They are mostly used for recreational purposes by everyday consumers on smartphones but can also be used for more formal artwork or projects.

In our project, we have combined techniques to construct a panorama from multiple images. We also employ the OpenCV library to execute the matrix calculations and visual transformations necessary for our algorithm.

2. Related Work

Most methods use Harris corner detection and SIFT to detect keypoints and extract features like our program does. Some [2] make use of the OpenCV libraries for SIFT and feature detection, but we wrote these methods out. However, there are options to use DoG, difference of Gaussian, instead of Harris corner detection. Some implementations also use SURF (speed up robust features) instead of SIFT, as it may be a faster method. Since we had experience with Harris corner detection and SIFT in previous projects, we chose to use these two methods. This is also the method outlined in the beginning of Matthew Brown and David G. Lowe's paper [1]. Their method uses SIFT to extract features and the k-nearest-neighbors method to match features. They also use RANSAC to solve for the homography matrix, which we use in addition to adding RANSAC to the cv2 homography function. However, there blending utilizes bundle adjustment to account for rotational differences from image to image and renders the panorama using multi-band blending.

There are also several other ways to blend the two images together. One method utilizes layer masks [3] to get the intensity of a certain areas of the images. We implemented this by highlighting the left side of the first image and the right side of the second image. Thus, the middle overlapped segment was blurred and adding the two masked images together would produce a blended image. However, this method proved difficult in combination with our translational shifts so we decided to use Laplacian pyramid blending instead. The mask method usually applies to programs for putting certain parts of one image into the foreground of another image, whereas our goal was to merge two images together.

Another type of blending that can be used to construct panoramas is Poisson blending [5]. This is different from Laplacian pyramid blending because it solves the Poisson equation to generate blended images. On the other hand, Laplacian generates a pyramid for each image and then combines them into one.

3. Methods

3.1. Keypoint Detection, Feature Extraction, and Feature Matching

A simplified version of SIFT for the local feature matching was implemented in three steps: detection, description, and matching.

To detect the key interest points of each image, we used the Harris corner detection method. This included first computing the image derivatives with Gaussian filters. Then we implemented a dynamic cornerness threshold and a minimum distance of feature points of 9.

Feature extraction was done using SIFT, taking in the detected feature points. Our implementation uses 16x16 patches in the algorithm so that we loop through 16x16 subsection 4x4 squares at a time where each cell of the 4x4 grid of cells contains a histogram of the local distribution of gradients in 8 orientations by adding the corresponding magnitude to the appropriate orientation. Thus, a 1616 patch around the feature point is converted into a 1281 descriptor of the gradient magnitudes and orientations.

Finally, to match the features, we used the Nearest Neighbor Distance Ratio Test to assign matches between features in the current two images. We compute the ratio of the distance of the closest feature vector neighbor (NN1) to the distance of the second-closest feature vector neighbor(NN2); this ratio can then be used to compute order of confidence for the NN2 matches. In addition, we do not match if this ratio exceeds a certain set threshold as this is then indicative that the two neighbors are both similarly close. This method helps lower the number of false positives and is able to efficiently trade-off between the number of false positives and true positives. Using the ratio of first best match and the second best match helps find reliable matches and addresses issues regarding repeated features.

3.2. Application of RANSAC

Our application of RANSAC is where we deviated from traditional methods in that we used it to calculate not only the inliers of each image but also the average distance between the inliers. This information we use later on in the translational phase where we shift each image accordingly to account for the overlapping segments in the images. By knowing the distance between matching features, we were able to calculate how much each respective image needed to shift towards each other in order to match up the combined image features. We initially tested this shift measurement by using the minimum distance between inliers so as not to overlap the images too much, but then found that this resulted in not enough overlap so we decided to take the mean of the inlier distances instead. Ultimately, we return the mean of the differences between the x-coordinates and y-coordinates of the inliers corresponding to the best fundamental matrix.

The inliers returned from our RANSAC function are used as parameters in the cv2.findHomography function call as the source and destination points. However, we discovered that without using cv2.RANSAC as a parameter as well (in addition to using RANSAC to find the inliers), our warped image that resulted from applying this calculated homography matrix was inconsistent with repeated iterations. In other words, some outputted images came out more warped than others. But finding the best homography matrix with RANSAC provided a more consistent and stable warped image output so we decided to keep this parameter. We think the initial inconsistency may be due to natural variations in the matches and inliers from the previous steps between iteration to iteration, thus resulting in varied homographies and warped images.

3.3. Warping and Shifting the Image

Before warping the image, we multiplied a translational matrix to the original homography matrix.

```
transl = np.array([
        [1,0,-H[0,2]+100],
        [0,1,-H[1,2]],
        [0,0,1]
    ])
final_transf = np.matmul(H, transl)
```

This is done because the original homography matrix could map the pixels to negative point locations and we'd see part of the warped image outside of the visible border. This additional matrix makes sure the whole warped image is mapped to positive pixel coordinates within the borders.

We then warp the image with this updated homography matrix using cv2.warpPerspective. Our method warps the second image to the perspective of the first so that each additional image is changing to match the view of the former. In other words, we transform the coordinates of the second image by applying the homography matrix so that the coordinates are now in their corresponding location in the first image.

After the warping the image, we take the two images to be stitched and make some adjustments so that they will be ready to blended together. In addition to the calculated shift from the RANSAC distance calculation, we also calculate how the warped image is transformed and manually calculate the translational shift by computing where the coordinate (0, 0) will map to after application of the homography matrix to the image. We then use both of these information and pad both images so that image A is effectively on the left half of the newly padded image A with the overlap not included on the left half of this padded image. Similarly, image B is effectively on the right half of the newly padded image so that it is aligned with image A with the shifts accounted for.

3.4. Blending and Stitching

In order to stitch the two images together, we opted to use Laplacian pyramid blending [4]. While initially, we simply copy and pasted the warped image B next to image A, this resulted in noticeable disparities between the

two different images. Thus, we attempted to both get rid of such a distinct seam while simultaneously maintaining a high quality in the area where the two images joined by getting blending the two images with minimal perceptual damage.

To blend, we found the Gaussian and Laplacian pyramids of each image with the number of levels equal to 4. We did this using cv2.pyrDown and cv2.pyrUp, respectively. Then, we join the left and right halves of the images, which align with one another as we padded and accounted for the shift between the two as mentioned previously, at each level of the Laplacian pyramids. Finally, the stitched image output is reconstructed using cv2.pyrUp again by collapsing the combined Laplacian pyramid to get the final blended image.

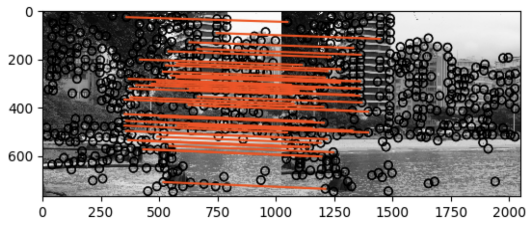
Afterwards, we crop the stitched images to get rid of the black bordering/padding we previously added for the final output.

3.5. Application of Algorithm to Set of Images

Once we determined how to stitch together two images, we placed this algorithm in a for loop to stitch together the remaining images. The stitched image becomes the first image and the next image in the data set becomes the second image to be warped to the perspective of the first.

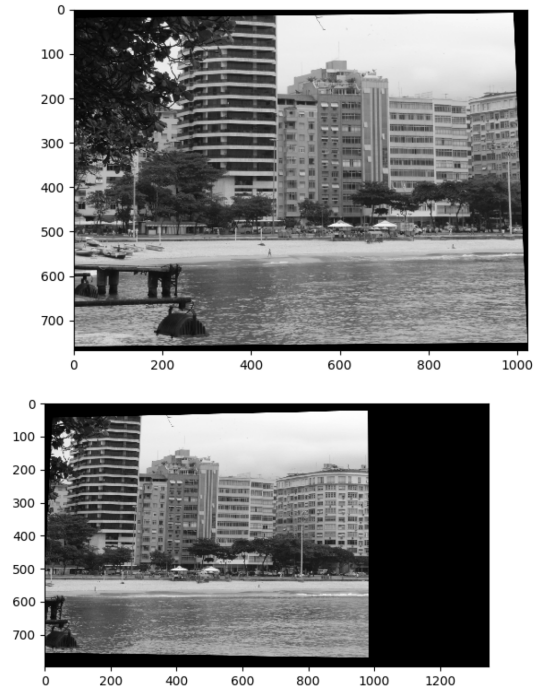
4. Results

4.1. Feature Detection and Matching



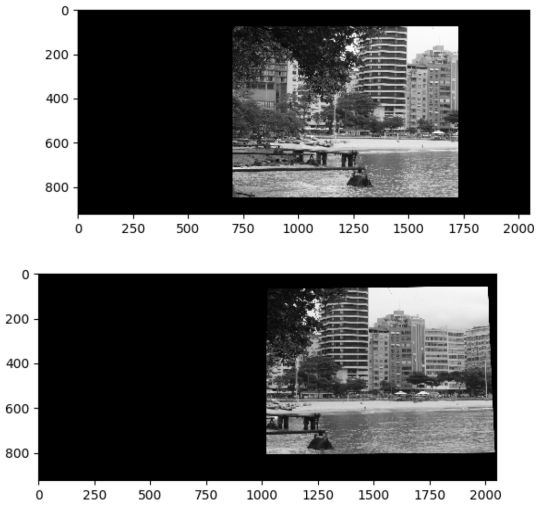
The detected features are the circles and the feature matches are indicated by the lines crossing from one image to the other.

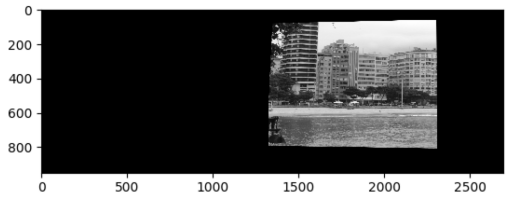
4.2. Warped Image



These images are warped to have the same perspective as the previous image. The top image is the second image added to the panorama image and the bottom image is the third image added to the panorama.

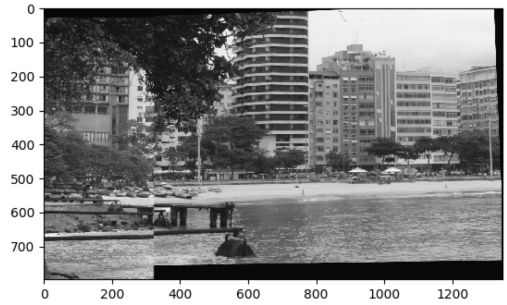
4.3. Translational Shift Image





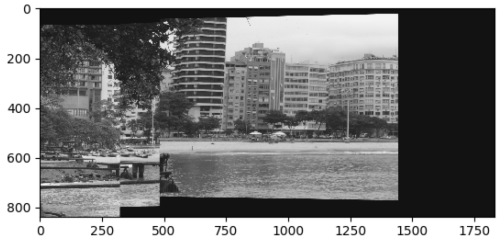
The images are positioned based on the translational matrix to accurately stitch the images on the matched feature points. The top image is the first picture and the bottom image is the second picture in the final panorama image output.

4.4. Stitched 2-Image Output



This image is the result of stitching and blending the first image and the warped second image together. This is also the output of both images being shifted by the translational matrix.

4.5. Entire Stitched Image Output



This image is the final panorama image that we generated. It combines the stitched 2-image output (4.4) and a third image together. The third image was warped to match the perspective of the 2-image output.

5. Discussion

The final panorama picture is three images stitched together. After warping, shifting, and stitching the images together on their matched feature points, the final panorama image appears as if it was taken from one perspective. The results from feature detection and matching is as expected and depicts the specific features that were matched between the two images, indicating which points to conjoin the images. The results from the warped image are also as expected as they successfully skew the perspective of the inputted picture to match the perspective of the previous image. The translational shift images are significant as they correctly position the images to be stitched on the right feature points. The stitched 2-image output is somewhat surprising in that the top half appears almost seamlessly stitched together while the bottom half reveals some rigid edges. Similarly, in the entire stitched image output, the top appears smooth while the bottom of the image, again, exposes some unaligned edges. These discrepancies may be due to the warping of the images.

The final image output reveals some areas of potential future work; we could reevaluate our techniques for stitching and warping, account for the differences in contrast/saturation/brightness and move towards stitching unordered images together. By reevaluating the techniques we used in this project, we could try other techniques such as Bundle Adjustments and Multi-Band Blending as mentioned in Brown and Lowe's paper [1]. Additionally, we could experiment with the results of color images, which would require also adjusting parts of the overall images such as contrast and brightness to create a seamless finish.

6. Conclusion

In this paper, we described the techniques and algorithms used to produce our final panorama picture. We found that matching features, warping one of the images to the perspective of the previous image, shifting the images together on the matched points, and repeating this process was the best approach and produced the best results for us. This algorithm allows us to stitch together images of the same scene but of different perspectives and angles to create one large image. This project aimed to produce images similar to those produced by phones and softwares made for the purpose of stitching images together to create a panorama picture. While there are areas for potential future work, our final image output demonstrates the successful effort to stitch, shift, and blend images together to produce one final panorama image of the scene as if all the pictures were taken from the same perspective.

References

[1] Matthew Brown and David G Lowe. *Automatic Panoramic Image Stitching Using Invariant Features*. 2007. URL: matthewalunbrown.com/papers/ijcv2007.pdf.

[2] Python Lessons and Analytics Vidhya. *Image stitching with OpenCV and Python*. 2019. URL: https://medium.com/analytics-vidhya/image-stitching-with-opencv-and-python-1ebd9e0a6d78.

[3] nkmk. *Alpha blending and masking of images with Python, OpenCV, NumPy*. 2019. URL: https://note.nkmk.me/en/python-opencv-numpy-alpha-blend-mask/.

[4] Kari Pulli. *Stitching and Blending*. 2014. URL: https://web.stanford.edu/class/cs231m/lectures/lecture-5-stitching-blending.pdf.

[5] Kai Wang and Pengbo Li. *Panoramic Image Stitching*. 2013. URL: http://www.cim.mcgill.ca/~siddiqi/COMP-558-2013/LiWang.pdf.

Appendix
Team contributions

Estelle Han

Estelle helped integrate the various methods and algorithms so that the information from each step would be coherent. She adjusted the code for RANSAC to calculate the translational shifts between the two images. She also implemented Laplacian blending and wrote the code for stitching the two images with adjustments including the calculation of the shift after the image is warped, padding of the images, and cropping of the padding after blending. She was also involved in applying the algorithm to a set of images and generating images for the Results and helping write the Methods section of the report.

Casey Kim

Casey implemented the RANSAC function and helped manipulate the code to have the functions run together to produce a final image. She also worked on using our panorama image results to create a tiny planet, which we did not include in the final implementation/report. She was also involved with writing the Results, Discussion, and Conclusions sections of the final report.

Anya Li

Anya implemented the feature detection and feature matching code and helped modify it to make it easier to use for panorama-stitching purposes. She also added to the homography matrix section to make sure the entire warped image would show up instead of some points mapping out of bounds. She also helped code the application of the algorithm to a set of images and was involved in writing the Introduction, Related Work, Methods, and References section of the report.