Automatic Panoramic Image Stitching

Introduction:

For this project, I decided to implement the algorithm described in Brown and Lowe's paper, "Automatic Panoramic Image Stitching using Invariant Features". The algorithm takes an unordered set of n images and produces a panoramic output by aligning and blending the images following a mutli-step process: extracting SIFT features, finding nearest-neighbor matches using a k-d tree, selecting candidate image pairs, computing homographies with RANSAC, verifying matches, and identifying connected image components. To begin implementing the panoramic stitching pipeline, I started with a pair of images, and then scaled up to more images. I ensured everything worked correctly for just two images before adapting the pipeline to handle multiple unordered images.

Part 1 Approach (No Blending):

The first step was to extract local invariant features from each image using cv2.SIFT_create(). SIFT, or Scale-Invariant Feature Transform, is essential in panoramic stitching because it helps reliably match features between overlapping images and it produces stable and distinctive features that are robust to changes in scale, rotation, and lighting.

Next, I used the FLANN-based matcher to find the closest feature matches between image pairs. I applied Lowe's ratio test to filter out incorrect or ambiguous matches, and only kept the ones with high correspondence. The output is a dictionary storing the good matches for each image pair.

With the matched features, I computed the homography matrix between each image pair using cv2.findHomography() and RANSAC. This helped detect outliers and allowed me to find the geometric transformation needed to warp one image onto another. I stored each computed homography matrix in a dictionary.

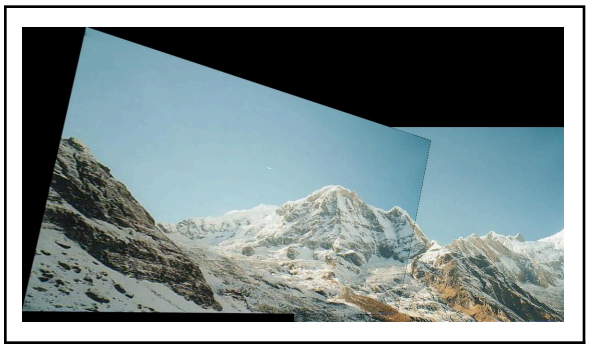
To align all images into a single frame, I built a graph where nodes represent images and edges represent the matching image pairs where there are good matches between the image pair. I constructed a minimum spanning tree from this graph and selected the image with the highest

degree as the reference image. Then, using the shortest paths from the reference to every other image, I computed cumulative homography by multiplying the homographies of these paths, inverting homographies as necessary. This allowed all images to be consistently wrapped relative to the reference image.

After computing each image's cumulative transformation, I calculated the output canvas bounds by projecting image corners using cv2.perspectiveTransform(). I adjusted each transformation with a translation matrix so all warped images would fall within a positive coordinate space. I then applied cv2.warpPerspective() to warp and position each image accordingly.

Now, the images are geometrically aligned and placed onto a panorama canvas. However, clear seams and overlapping edges are visible due to exposure differences and misalignments, especially in overlapping regions. These are what the current results look like.

Part 1 Results:



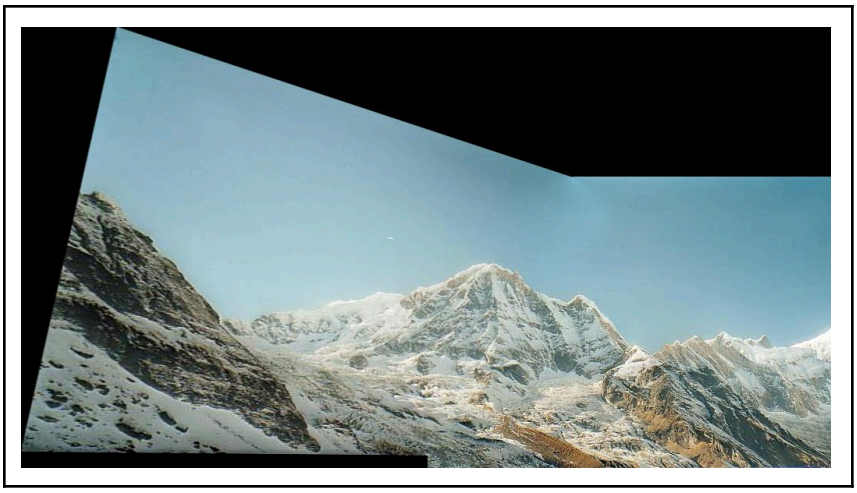
Part 2 Approach (With Blending) :

After warping these images into a shared coordinate frame, although the images were aligned, there were noticeable seams and exposure differences. To fix this, I initially implemented standard pyramid blending using Gaussian and Laplacian pyramids. However, even with pyramid blending there were still some issues with the overlapping regions. This issue occurred because regular pyramid blending did not account for how much each image should contribute to a given region, particularly the overlapping areas.

To improve the result, I tried weight pyramid blending, which improves blending in overlapping regions by explicitly controlling how much each image contributes to each pixel, using spatially varying weight maps.

First, I generated a weight map for each image based on the distance from valid, non-zero pixels, giving higher weights to image centers and tapering off near the edges. I then constructed Gaussian pyramids of the blurred weight maps and Laplacian pyramids for each input image. I also used validity masks to avoid blending imperfections from the empty regions. At each level of the pyramid, I combined the Laplacian pyramids using the corresponding weight maps and normalized the result. The final blended image was reconstructed by collapsing the blended pyramid back to the full resolution. A global mask was applied at the end to ensure black regions were not included in the output. This method produced smoother panoramas with minimal visible seams.

Part 2 Results:



Future Work:

There are several improvements that could be made to my implementation including automatic panorama straightening which would correct curved or tilted outputs. Also, gain compensation would help normalize brightness differences between images, and reduce visible transitions in overlapping areas. Additionally, to correct the empty, black regions in the final panorama, future work could include automatic cropping and image inpainting to fill in missing areas and produce a cleaner result.

Conclusion

In this project, I successfully implemented most of the panoramic stitching pipeline described in Brown and Lowe's paper. After aligning images using SIFT features, homographies, and a minimum spanning tree, I used weight pyramid blending to reduce seams and blend overlapping regions smoothly. The results show that this method produces clean and coherent

panoramas. Future improvements could include automatic straightening, gain compensation, and filling empty regions.