**CSE 691: Image and Video Processing**

**Spring 2020 Final Project**

**Panorama Stitching**

***Lichen Liang***

**Abstract**

This paper talks about panorama image stitching. Image stitching has been a widely studied topic in the past two decades. Its goal is to combine multiple similar images of a scene or object, usually photographs, into a single image that has a panorama view. Image stitching is now very commonly used in digital cameras, mobile phones, and surveillance systems. This paper implements the algorithm introduced by Brown and Lowe [1]. On top of that, the stitched result is improved by pixel estimating to fill the black regions where there is no information on that pixel in the input images.

1. **Introduction**

Panorama image stitching traditionally uses direct method which includes using corners, edges, shapes as features [1]. This method is also described as region related image registration [4]. These feature are variant to scaling and rotation. It is also semi-automatic in the algorithm such that human defined parameters are heavily affecting the result. Later, people developed a fully automatic algorithm that is feature based. It uses SIFT (Scale Invariant Feature Transform) features with RANSAC (Random Sample Consensus)[1, 2, 4, 6]. Many papers uses the feature method as a base and elaborate on that. Xiong and Pulli [3] also introduced the optimal seam method for color correction and transition. Both feature and seam methods are implemented in [4] with a minor change. The method used in this paper also used image warping. The warping part can be improved by [5] where they introduced an optimized warping method that can be applied to the code. This paper improves the output of the stitched image. However, it has a lot of limitations and improvements that can be further done.

Section 2 includes the analysis on each method. Section 3 describes the method used, the improvement, and the problems along. Section 4 imposes a series of future improvements based on analysis of output and the problems suggested in section 3. Section 5 concludes the paper and section 6 includes the references.

1. **Method**

**SIFT Features**

SIFT features are the key points of scale space. These points are scale, rotational, and intensity invariant which is critical to image matching. First, the image is convolved with multiple Gaussian filters and taken the difference of gaussian (DOG). Then find the minima and maxima of the DOG. These are the key points. Filter out the unstable and low contrast points to make sure the matching is stable. From the key points, we can compute the descriptor vector. First, rotate the axis to match the key point’s orientation. Each key point has a region. Divide the region into sub-regions with each region having a histogram of gradients. Finally, sum these gradients and magnitudes to form a descriptor vector and normalize. Each feature has its own descriptor vector and images with same features will be matched.

The matching is done by using a k-d tree to calculate the minimum Euclidean distance. Matched images are formed into pairs to be further matched and later verified using RANSAC.

**RANSAC**

Random Sample Consensus uses random sampling in each iteration to finally create a mathematical model. We can use this model to retrieve the inliers and outliers information. Using this information, we can use the matched number of inliers to verify if two images actually match. This is essentially building a verification model. With this approach, we calculate the probabilities of inliers in an image match to the inliers of another image to verify if two images are the best match. These probabilities are used in the bundle adjustment.

**Bundle Adjustment**

Bundle adjustment uses Levenberg-Marquardt (L-M) algorithm. This algorithm solves the non-linear least squares problem. Project each feature of an image to all the images with the matching feature and calculate the sum of squared error. Images are matched one by one to another image with most number of matches. The newly added image is set to the same parameters as the best match image in terms of focal length and orientation. The output of this part will be images all matched with its best match.

**Color Blending**

There are two types of blending techniques: alpha blending which uses smooth transition approach but creates ghosting by moving objects and alignment errors, and optimal seam finding [3]. The latter method finds overlapping areas with minimal difference between different images. It works by correcting colors in all images and smoothing colors in overlapping images. First, calculate the color correction for all images and use this to get a global correction coefficient. Obtain an image with best colors, and use it to correct the first image in the sequence. Get the next image in line, perform color correction to it and get the overlap between the current one and the previous one. With the overlap, calculate the error surface and this is used as optimal seam. Finally, cut the overlapped area along the seam and merge them. Repeat this method for all remaining images left in the sequence. Once it is done, what is left is the panoramic image. Note that this uses images in a left to right order.

**Other Methods**

In paper [4], they combined algorithms used by [1] and [3]. At the same time, they also pointed out some problems using these two methods. For example, the left image (merged images) in the optimal seam algorithm continues to grow in size and complexity as the algorithm runs. This uses a lot of system resources. Also, the sequence in optimal seam is fixed, such that the next image may not be the best fit. In other words, it ignores the area of the overlapped portion. This means that if the an image is not the best fit to another, it produces an error that will be carried on.

Instead, they imposed an algorithm with minor changes. With a sequence of images and all the features stored in a feature list, calculate the affine matrix that can transform one image’s coordinate space to another image’s. Store this affine matrix in the array *Hlist* and repeat for all images in the sequence. Start from the middle image in the sequence, find another image with the most matched features, and calculate the affine matrix between the two. Optimized this matrix using Levenberg-Marquardt algorithm. Finally, find the optimal seam and combine. The advantage of this algorithm is that it reduces the number of SIFT features needed. It also start from the middle and work in two directions, which prevent the potential error being carried in the merging process.

1. **Improvement**

The output of the algorithm in [1] results in black pixels around the stitched image due to unknow data pixels. For example, if an camera is taking photos by turning in a horizontal level, the stitched image would be a wide-angled with smooth borders. However, if the camera itself has turned a little bit in the clock or anticlockwise direction, it would result the corners of the image being extra. When the images are stitched together it would create a black region. If we crop the image to remove the black region, then some of the details in the stitched image would also be lost. If all images are taken in different angles, then cropping is not a valid solution.



Figure 1. Stitched image before fix

Figure 1 shows a stitched image before the improvement. The black regions are there due to missing data. The imposed solution is to estimate pixel values from the valid pixels and replace the black region. First, find the top, bottom, left, and right boundary where there is data. Crop the image into the size of the boundaries. Count the number of black pixels and get the ratio of black pixels to the whole image. While the ratio is more than a threshold, fill in the black pixels by using a 3x3 mask over that pixel and calculate average of the 8-neighborhood pixel values which exclude 0s. Repeat the process until the ratio drops below certain threshold. Finally, the output should be a filled image with one pixel black border, due to the 3x3 mask. Remove this simply by cropping it out.

**Problems and Limitations**

1. The algorithm highly depends on the input images. Images with less details such as a sky, ocean, or being same color at the boundary can work well. However, images with large number of details such as Figure 1, where there is a lot of edges, corners and color changes will work poorly.
2. The use of black pixel ratio requires a threshold. Each image set requires a different threshold. This makes the algorithm unable to perform image stitching on multiple sets of data. Threshold will need to be changed for every different image set, which needs human testing multiple times. Instead, the number of iterations are set to 40. With a small number of iterations, all the black regions might not be all filled, and with a large number of iterations, it will increase the execution time.
3. The distribution of black pixels is also a factor. If the majority at the top, it would be acceptable because only one row of pixels would be added in each iteration. However, the bottom pixels would be added in the first iteration and makes the image look very poorly filled.

In the next section, where each image set is analyzed separately, there will be cases with the above problems pointed out, along with some extra limitations and the proposed fix to that.

1. **Result Analysis and Future Improvement**

To test the code, first go to /lib/vlfeat-0.9.20/toolbox/vl\_setup.m and run it in MATLAB. Then run main.m

There are 11 test sets, with some of them producing an acceptable result, and the others are poorly processed due to both stitching problem or limitation in the improvement. There will also be a proposed future improvement to current results



Figure 2. Stitched image before the improvement.



Figure 3. Output Image after the improvement

Comparing figures 2 and 3, we can see that the black regions are cropped and filled with neighboring pixels. The bordering values do not have too much change in the grey scale value, therefore it is hard to tell which pixels are filled in. Also, the output shape of the stitched image is very friendly to the pixel filling because it is in a regular rectangle shape.



Figure 4. Office Before



Figure 5. Office After

Comparing figures 4 and 5, we can see the top of the image is filled with white that has been successful. However, the very bottom border is blurred (3rd problem in the previous section) which looks very unnatural. A proposed solution will be pattern detection over a large area and fill in a region of pixels instead of one pixel.



Figure 6. Yosmite Before



Figure 7. Yosmite After

From figure 6 and 7, the most obvious limitation is that this currently only works with grey images or it will turn colored image into grey. An improvement would be processing images directly regardless of their color. There have been attempts but it did not successfully work.



Figure 8. Rio Before



Figure 9. Rio After

Looking at the top-right filling of figure 9, it is noticeable that there are some white fillings instead of grey. It may look not too obvious due to the fact that the images stitched so far are in a fairly regular shape. More of this problem will be discussed later.



Figure 10. Yard Before



Figure 11. Yard After

From figures 10 and 11, a portion of the top right corner is not filled. There can be two reasons for this: number of iterations or those pixels are not purely black. The first can be fixed by increasing the number of iterations, and the latter set a range of acceptable grey scale levels rather than fixed 255 (i.e all values less than 5). However, the solution to the latter imposes a problem where if there is a pure black pixel in the middle of the image, it will be replaced by a grey value instead.



Figure 12. Diamond Head Before



Figure 13. Diamond Head After

Same problems from figure 13 happened here. There is black unfilled on top left, wrongly filled on top right, and blur in the bottom.



Figure 14. Fishbowl before



Figure 15. Fishbowl after

The larger the bottom empty region, the more unnatural the filling at the bottom.

So far, mostly all image sets produced a fairly regular shaped stitching. Even though there are some unnaturalness, it may not seem too obvious. The following images below will point out the exact problems.

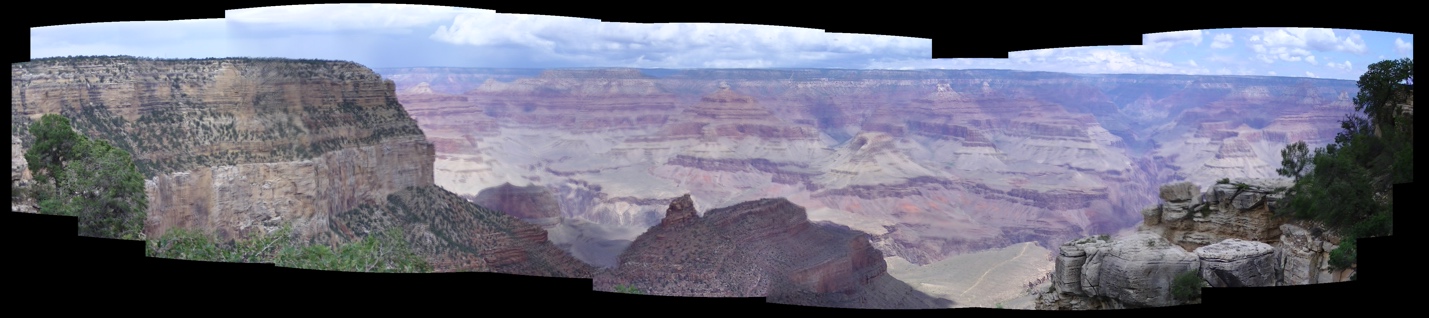


Figure 16. Grand Canyon Before



Figure 17. Grand Canyon After

The stitching of Grand Canyon image set resulted in a very irregular shaped output. The top right still has unfilled parts which can be resolved with more iterations. The whole top border is filled with white instead of grey which the reason is unknown. Most probably because of the original colored layers of the image. This made the image to be very unnatural. Moreover, the bottom left and right had a huge space of empty, which is filled with blur. A proposed solution here is to use edge and corner detection to correctly fill in the edges along with the correct pixel values. These values does not have to be average values but rather exact neighboring pixel values. Also, copying a region of pixels is also a possible solution by directly filling the holes.



Figure 18. Hotel Before



Figure 19. Hotel After

The most unnatural part of figure 18 and 19 is the top left corner. It is not filled and has an edge. However, in figure 18 at the same location, the colors were not completely blended. This problem is caused in the stitching phase.

The bottom of the image is also blurred. An edge detection and area relocation of pixels would work very well in this image, where the edge of the road, and area relocate the rocks.



Figure 20. Carmel Before



Figure 21. Carmel After

Again, the shadows of the tree is distorted. Computing the edge of the shadow an fill in the correct pixel values would solve the problem.

1. **Conclusion**

In this paper, we have implemented the panorama stitching by Brown and Lowe [1]. The output image has black regions due to missing pixel data. In addition to the algorithm, we implemented a fix for that issue. That is filling out the holes using nearby pixel values. At the same time, we discovered a lot of limitations and problems with this improvement. There are also a lot of improvements we can make in the stitching process. The problem and fix for that is listed in table 1 below.

Table 1. Issue and proposed fix(unordered from time of appearance in this paper)

|  |  |
| --- | --- |
| Problem | Proposed fix |
| Colored image must turn to grey to fix border | Develop the code to let it work with colored images |
| Blur in the original stitching | Use multiband blending or optimal seam method(might also be caused by moving objects) |
| Black pixels unfilled | Increase number of iterations |
| Bottom regions are blurred | Develop edge, corner, pattern detection and fill in pixels that match to that specific area. Also use a region of pixels rather to fill another region rather than a single pixel. |

1. References