Project Report

CS512- Computer Vision
FALL-2018

Topic: Panoramic Image Stitching

Parthkumar Patel – A20416508
Arpit Hasmukhbhai Patel – A20424085

❖ Problem Statement:

Image stitching is considered as a very active research area in image processing and computer vision. It is fairly well researched problem which has been solved under certain constraints in last several years. Image stitching involves two or more images of the same scene which needs to be stitched together into one high resolution image. Image stitching techniques can be classified into two categories: 1. Direct techniques and 2. Feature based techniques. Direct techniques compare all the pixel intensities of the input images with each other, whereas feature-based techniques try to find association between the images through distinct features extracted from the input images. Feature based techniques has the advantage of being more robust against scene movement, faster, and has the ability to automatically discover the overlapping relationships among an unordered set of images. Our project will focus on implementing an accurate image stitching algorithm using feature-based techniques.

❖ Proposed Solution:

We take multiple image from a sourced directory and pass it to the program. These images will be processed and feature point are detected from images and stitched it together one by one. We will be using RANSAC algorithm and Harris corner detection to get a Panoramic image.

Here is how image we will processed to get a panoramic image from multiple individual images:

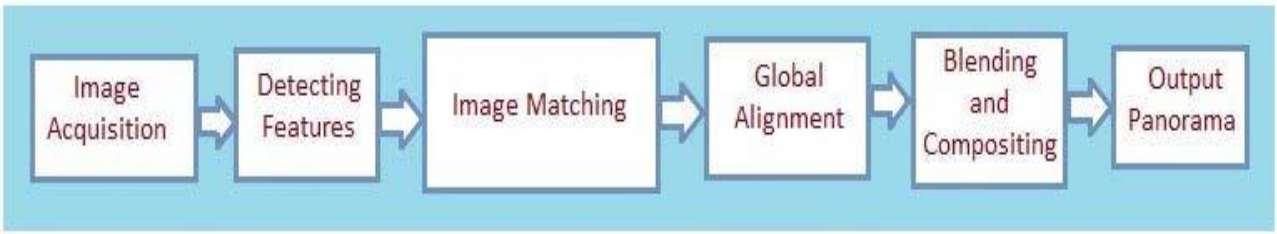

Figure 1: Process flow of panoramic image stitching [1]

1. **Image Acquisition:**
   Image acquisition means selecting files from computers from secondary sources.

2. **Feature detection and image matching:**
   In feature detection and matching we need some information like registration of image, stabilization of video and reconstruction of 3D. In this corner are matched to give quantitate measurements. It provides a rational matching for the image pairs based on rotation, translation and scaling.

3. **Alignment:**
   The most suitable method is photogrammetric method to merge many images of the identical scene in a perfect reconstruction of 3D scene and it is known as bundle adjustment. The purpose of the alignment is to locate a reliable alignment parameter set which can decrease the miss-registration between the pair of the images. It is very useful to widen the pair wise image matching.

4. **Blending and composition:**
   When few images are stitched together, one image is selected as the reference and then twisted all images in the system of that location. The result is often panorama. Feathering is used to blurring the edges of the features. From the two overlapping images, the average pixel value is evaluated for blended region.

5. **Output:**
   The result of the above steps is often known as flat output panorama.

❖ Implementation Details:

Libraries required to run the program:

- OpenCV Version: 3.4
- Python Version: 3.6.6
- numpy
- matplotlib
- multiprocessing
- os
- sys
- math

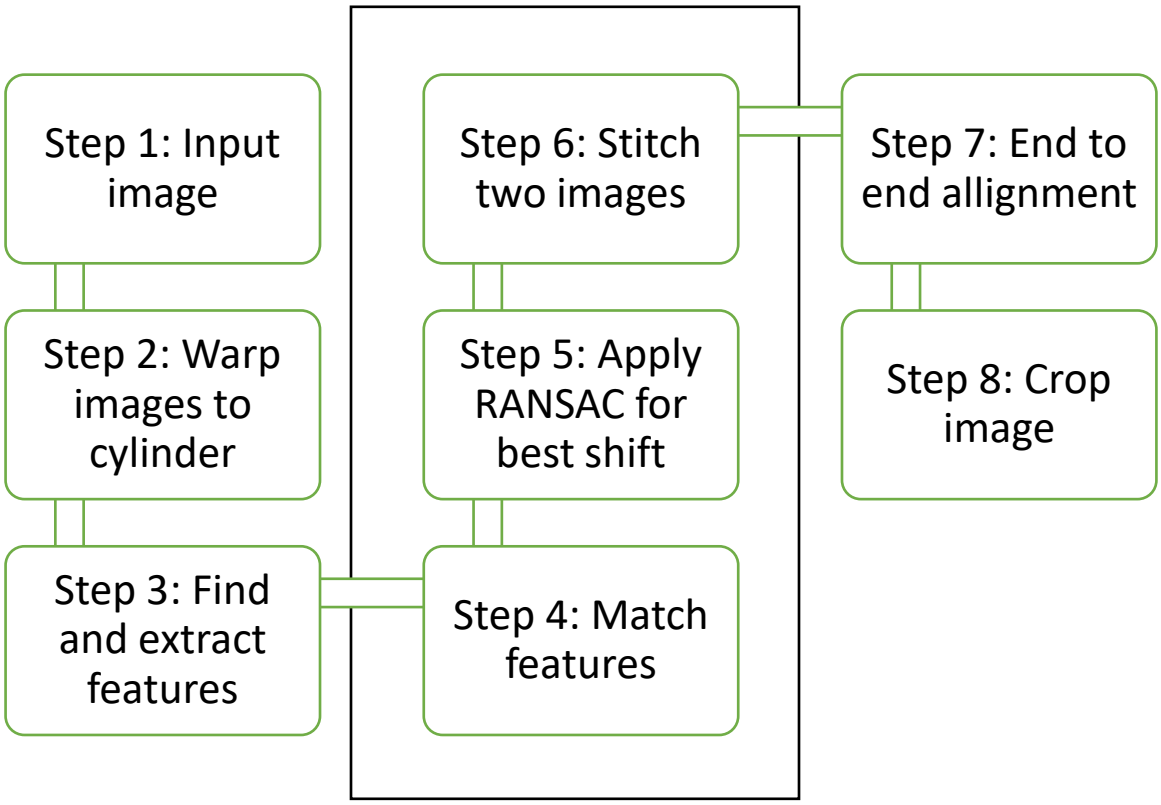

Figure 2: Implemented Process flow of our panoramic image stitching

Step 1: Input Image

First, we read individual images from the given path by user and take text file that contains the focal length of images. If the user does not specify the path of images correctly then it shows the error and print the help message which shows how to specify the path of input images. For that, we have built load_images_and_focul_length () function in main.py file which takes any source directory as an input and returns an array of images name and its focal length.

Step 2: Warp images to cylinder

When we take one image as an input, we know that how second image will look from first image's perspective. So, we need to transfer into new space. This process is called warping. We are using cylindrical warping. In cylindrical warping every image is represented as if the coordinate system was cylindrical and the image was plotted on the curved surface of the cylinder. For that, we have built

image_cylindrical_projection () function in main.py file which takes an image and its focal length as an input and returns cylindrical projection of an input image.

Step 3: Find and extract features:

First, we apply Harris corner detection algorithm to an input image and compute corner response matrix with the same dimension as an input image and then we extract the descriptor from corner response image which will give us the feature detected from an input image. For that, we have built harrisCornerDetection () function in featureextractionandmatching.py file which takes as an input an image, k - harris corner constant value and block size - harris corner windows size. It will return a corner response matrix. Now, corner response matrix will pass onto the extractDescofCornerResponse () function with threshold and kernel value as an input and it will give us the extracted feature from an input image.

Step 4: Match features

We have extracted feature from two images by using the step 3 functionalities. Now, we will match the pairs of images to match the feature between two images. For that, we have built matching Images () function which will take features extracted from both the images as an input with the range. Now it will take these inputs and it will compute the matching features between two images using computeMatchingOfTwoImages () function and it will return the refined matched pairs of images and will print how many features are matched.

Step 5: Apply RANSAC for best shift

Now, after matching the features we will apply RANSAC algorithm and calculate shifts. For that, we have built algoRANSAC () function which will take as an input value of matched pairs and previous shift for checking shift direction and it will return the best shift possible for an image. Also, it will raise the error if shift direction is not same as previous shift.

Step 6: Stitch two images

After performing an RANSAC on images now we will stitch those two images together using the best shift value. We will cut the unnecessary regions and will add padding to both the images wherever required and blend those images together. For that, we have built stitchingImages () function in the imagestitching.py file which will take both the images, shift value, and blending as true. It will return the shifted image of both the images stitched together.

Step 4,5 and 6 are repeated for every pair of images and it will continuously stitch the images together with the previous stitched image.

Step 7: End to end alignment

In end to end alignment we align the panoramic image and remove the y shift. So, we get the better panoramic image. For that, we have built endtoEndAlign () function which will take the last stitched image and total of all the shift as an input and it will return the aligned image as an output.

Step 8: Crop image

In crop image we detect the black pixel from upper and lower part of the panoramic image. After detecting the black pixels, we remove the black part of the image so we get real like panoramic image. For that, we have built a cropImage() function which will take as an aligned image as an input and it will return cropped panoramic image as an output.

❖ Results:

Example 1: Grail

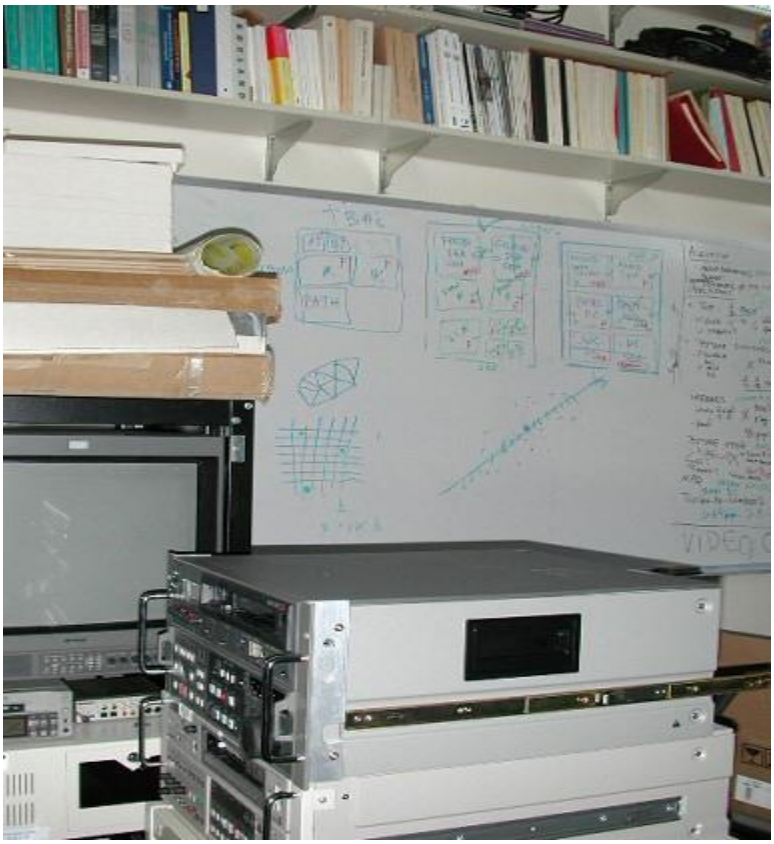

Figure 3: Input Image 1

Figure 4: Input Image 2

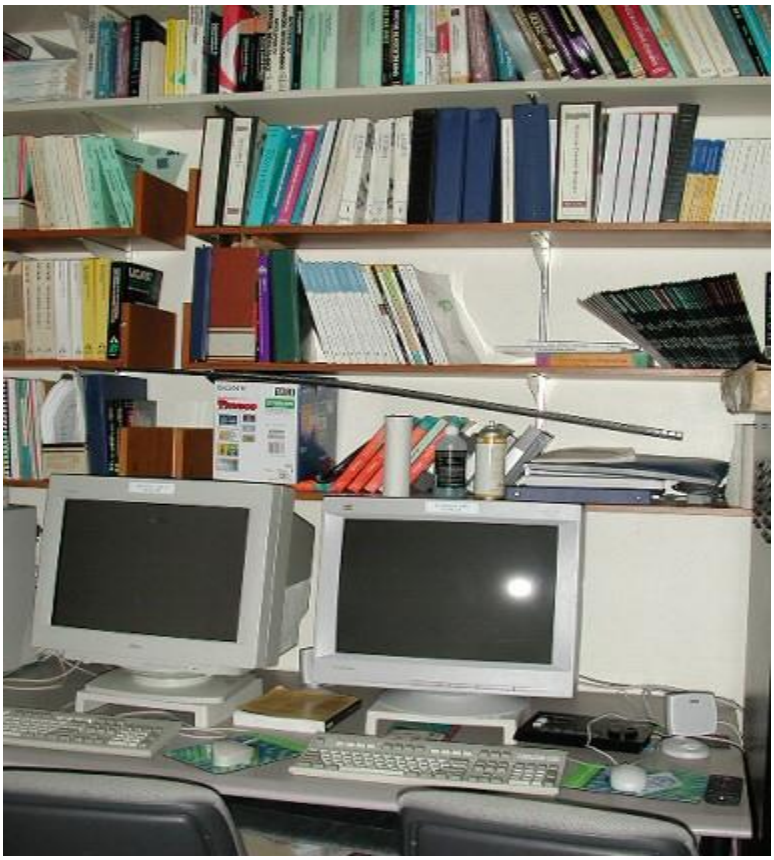

Figure 5: Input Image 3

Figure 6: Input Image 4

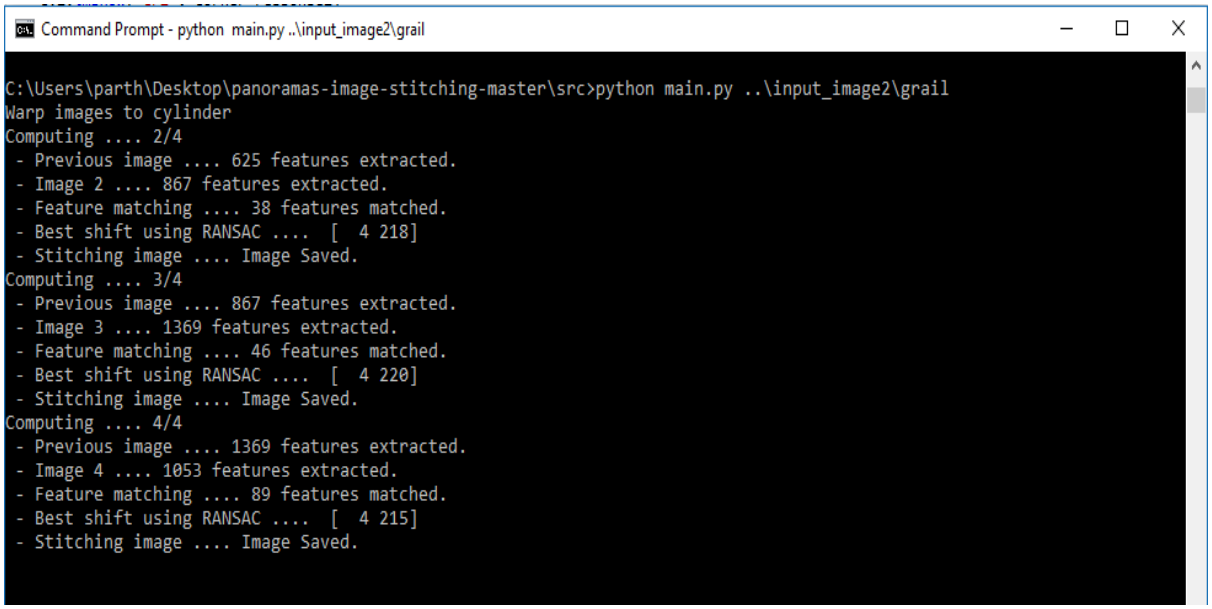

Figure 7: Snapshot of Code running

- Stitched Image 1:

Figure 8: Stitched Image 1

In image 1, 625 features are detected. In Image 2, 867 features are detected. Total, 38 features are matched between two images.

- Stitched Image 2:

Figure 9: Stitched Image 2

In image 2, 867 features are detected. In Image 3, 1369 features are detected. Total, 46 features are matched between two images.

- Stitched Image 3:

Figure 10: Stitched Image 3

In image 3, 1369 features are detected. In Image 4, 1053 features are detected. Total, 89 features are matched between two images.

- Aligned Image

Figure 11: Aligned Image of Grail

- Cropped Image

Figure 12: Final Cropped Panoramic Image of Grail

Example 2: Book

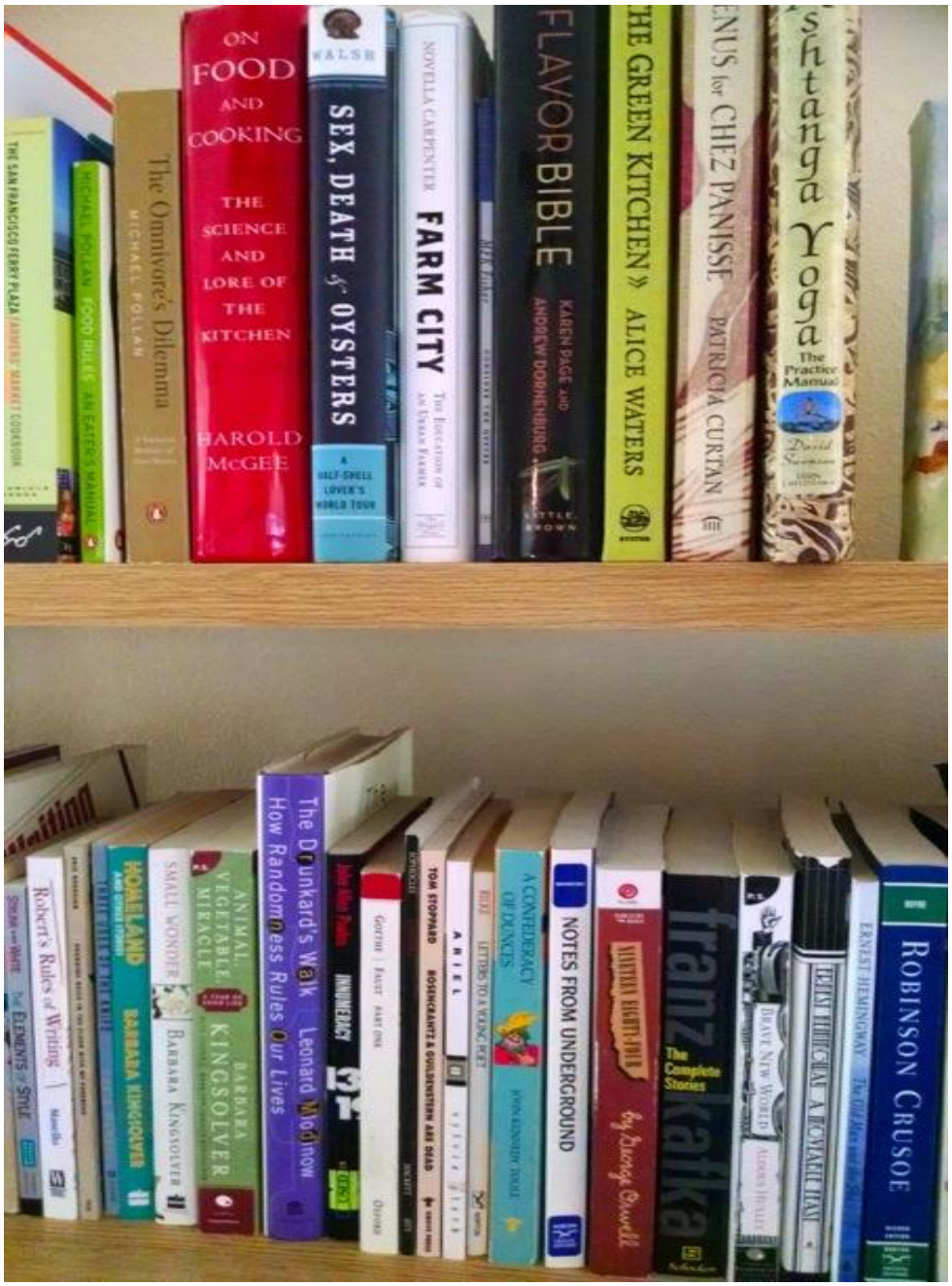

Figure 13: Input Image 1

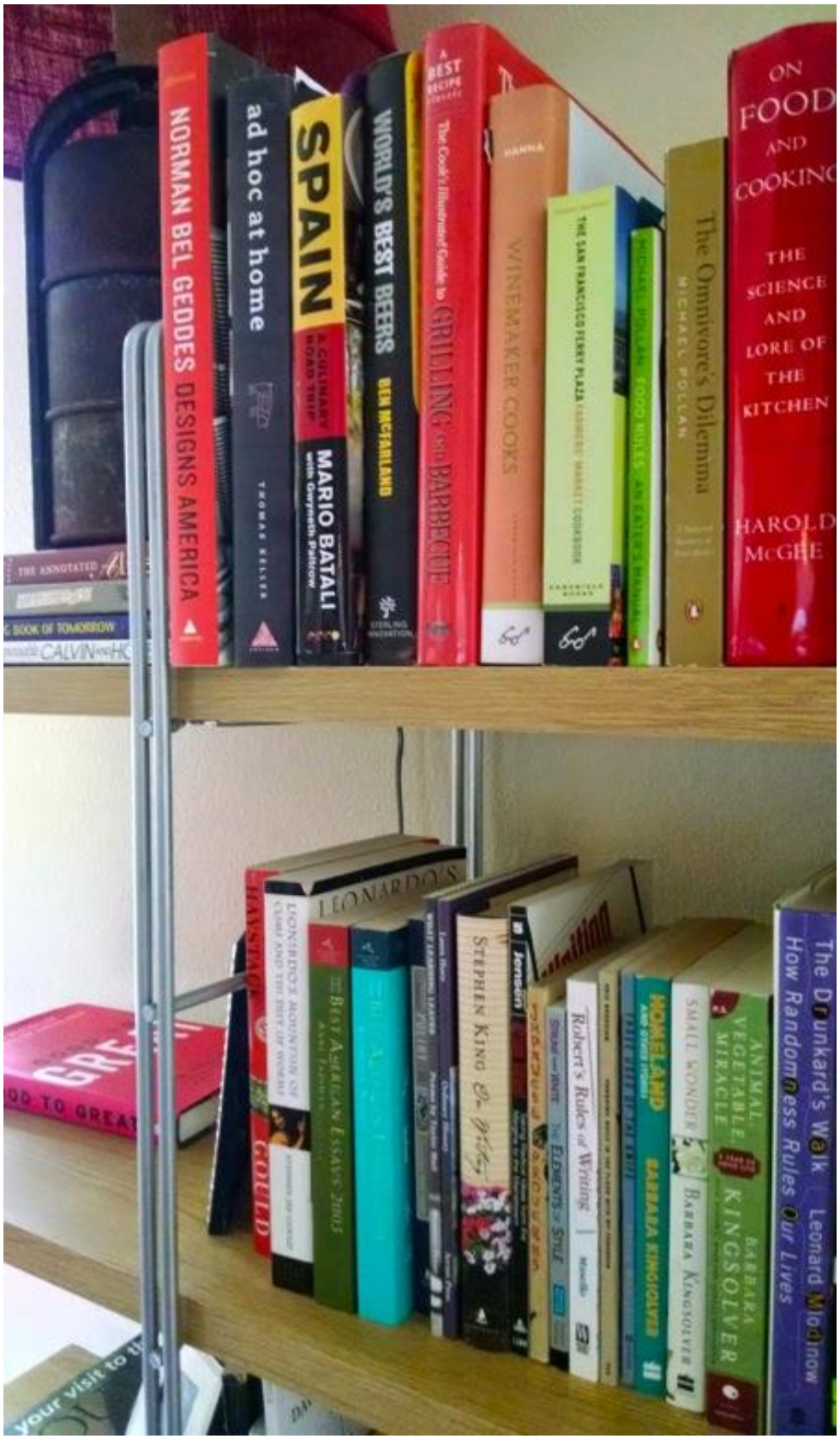

Figure 14: Input Image 2

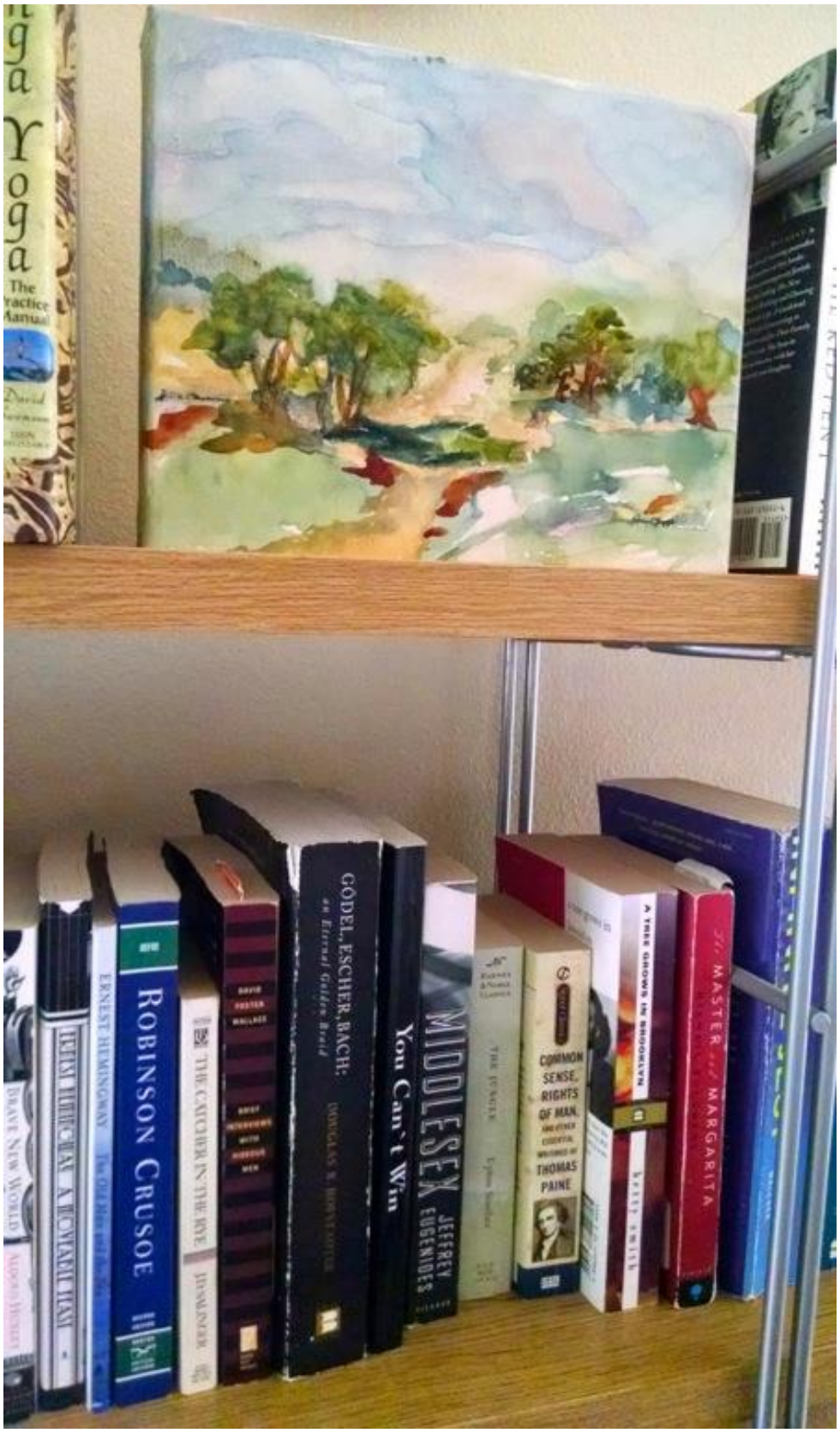

Figure 15: Input Image 3

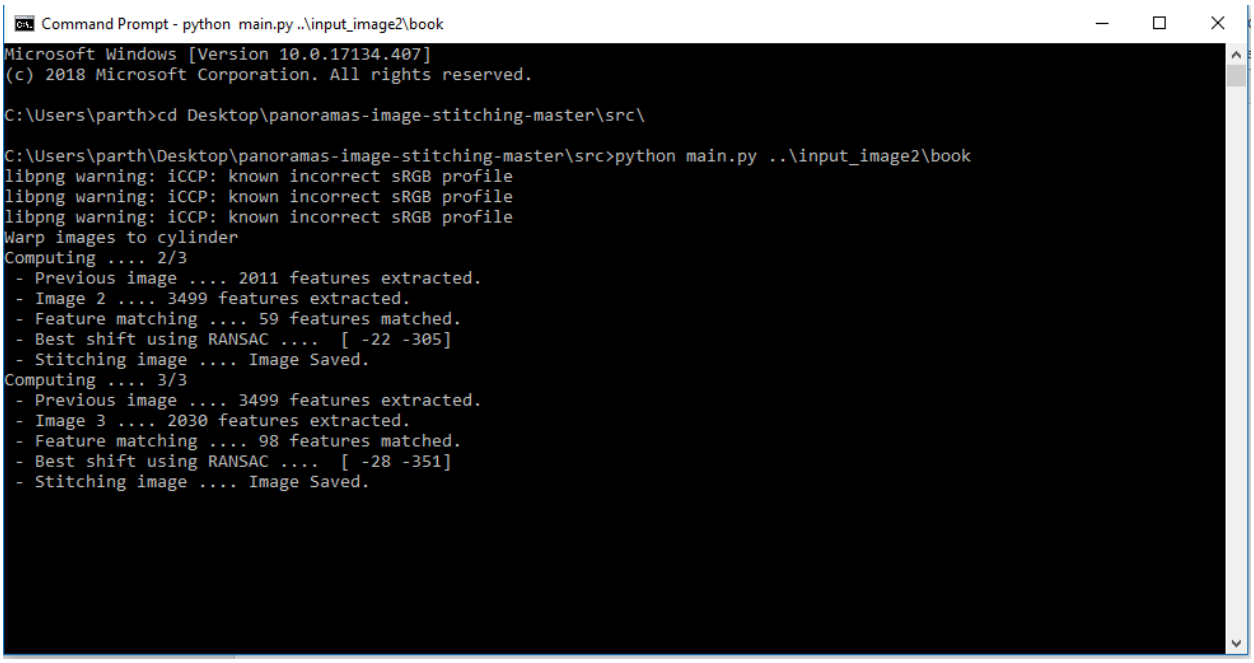

Figure 16: Snapshot of Code running

- Stitched Image 1:

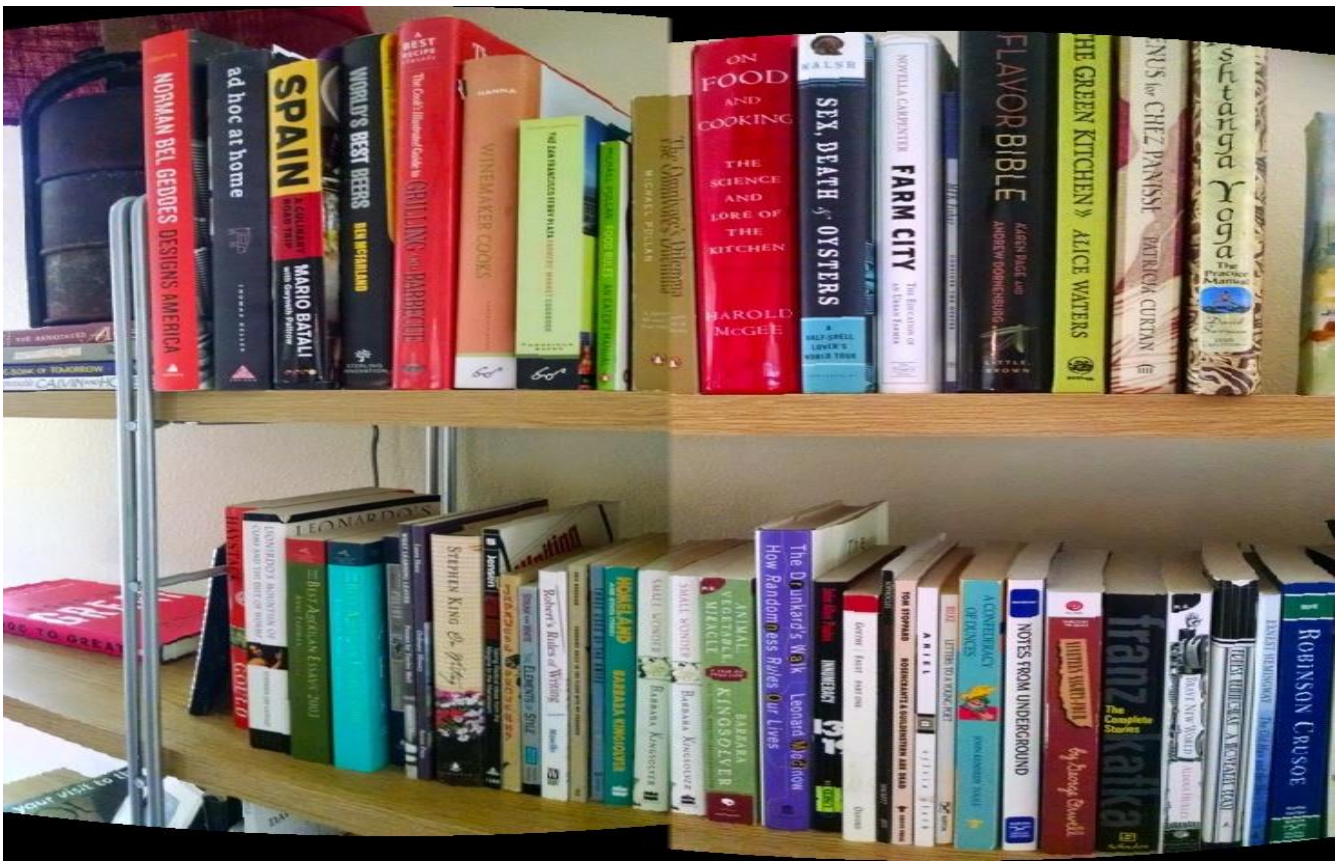

Figure 17: Stitched Image 1

In image 1, 2011 features are detected. In Image 2, 3499 features are detected. Total, 59 features are matched between two images.

- Stitched Image 2:

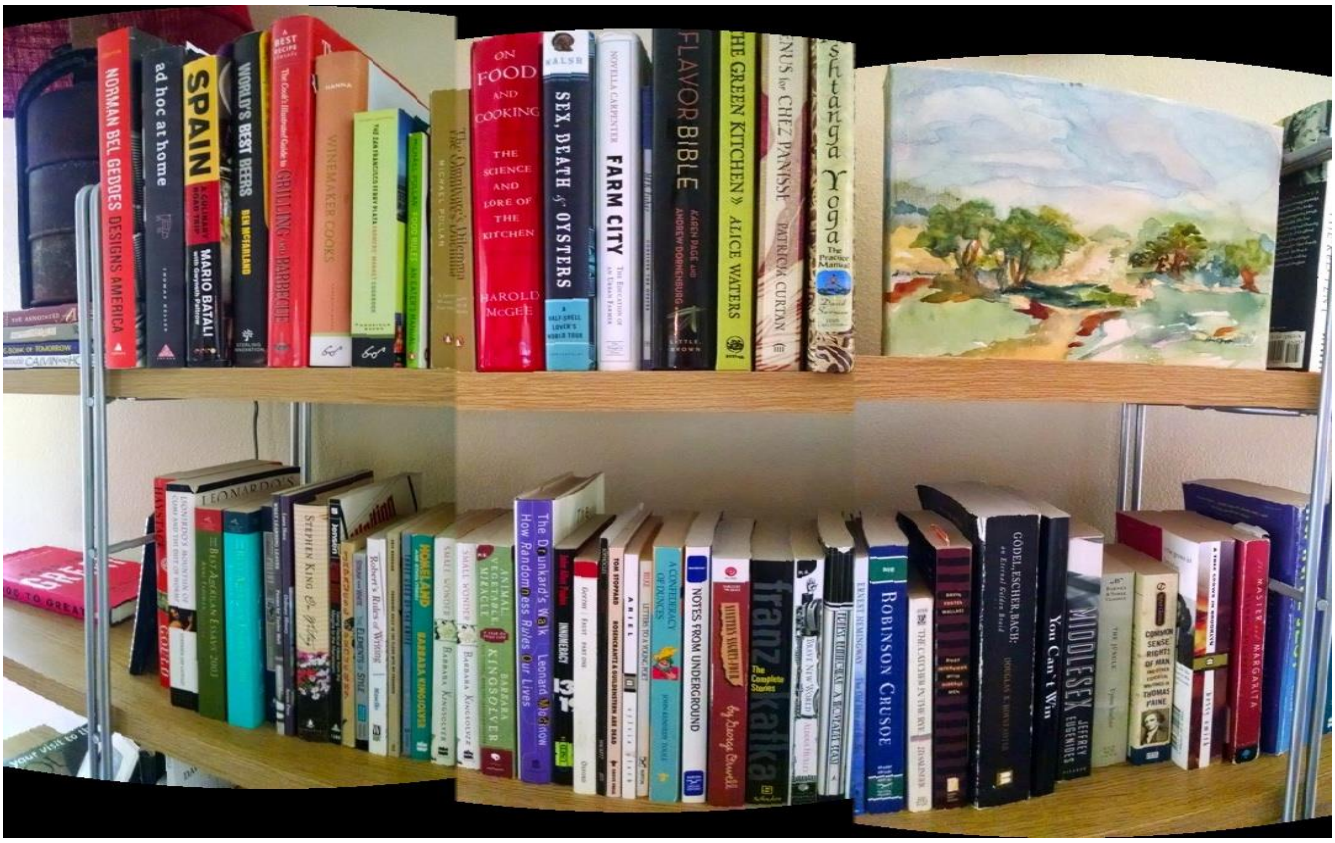

Figure 18: Stitched Image 2

In image 2, 3499 features are detected. In Image 2,030 features are detected. Total, 98 features are matched between two images.

- Aligned Image

Figure 19: Aligned Image of books

- Cropped Image

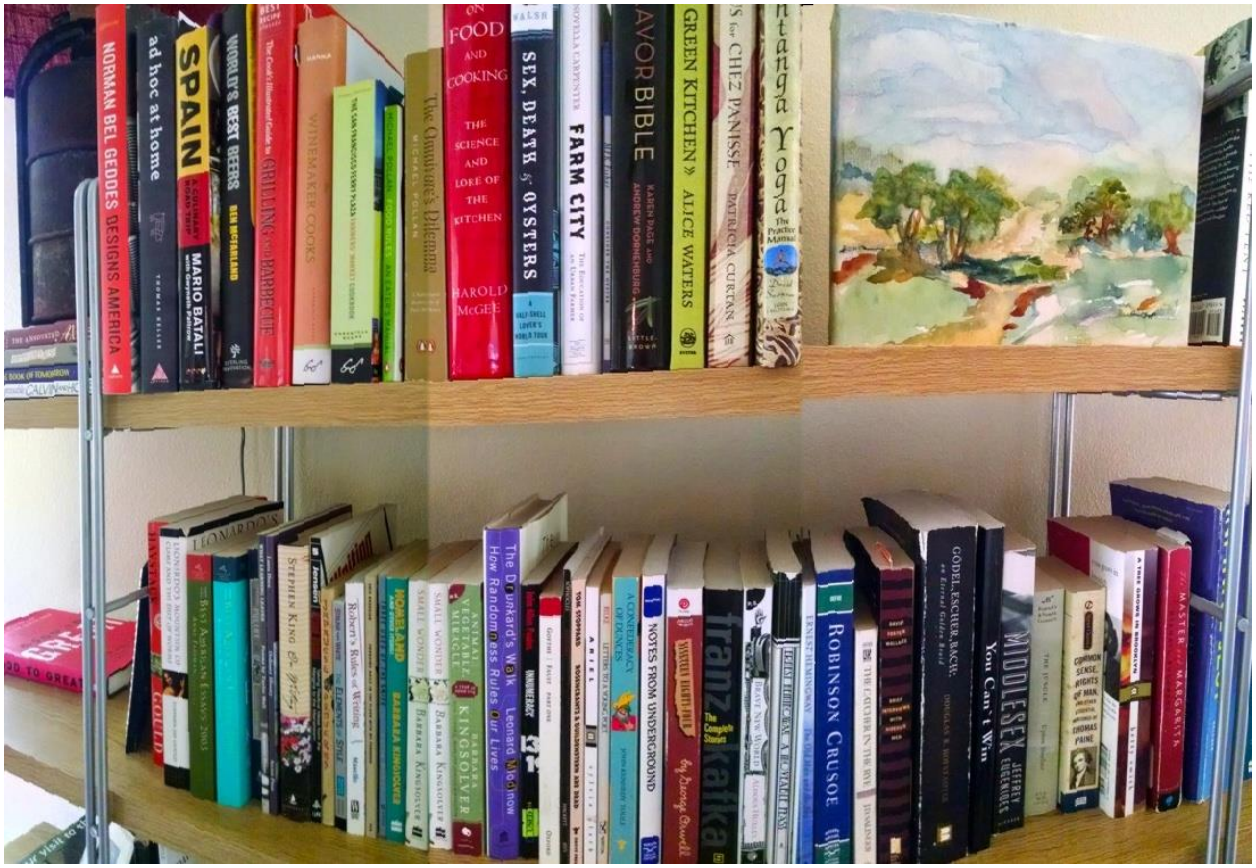

Figure 20: Final cropped panoramic Image of books

❖ References

1. Bonny, Moushumi Zaman, and Mohammad Shorif Uddin. "Feature-based image stitching algorithms." Computational Intelligence (IWCI), International Workshop on. IEEE,2016. [5]