Giorgio Mendoza
CS549-E24-E01
Dr. Alexandros Lioulemes

Week 4 Lab

Part 1:

In Task 1 of this, the objective was to develop a panoramic image by stitching together two separate images. This process involved a sequence of steps using standard CV techniques: Firstly, distinctive features in each image were detected using the SIFT algorithm particularly. Next, these features were matched across the two images employing either Brute-Force or FLANN (Fast Library for Approximate Nearest Neighbors) matching algorithms.

Then, a homography matrix, which is important for aligning images taken from different perspectives, was computed using RANSAC to ensure effectiveness against outliers. Finally, this homography was applied to warp the second image to the perspective of the first, by combining them into a single panoramic view.

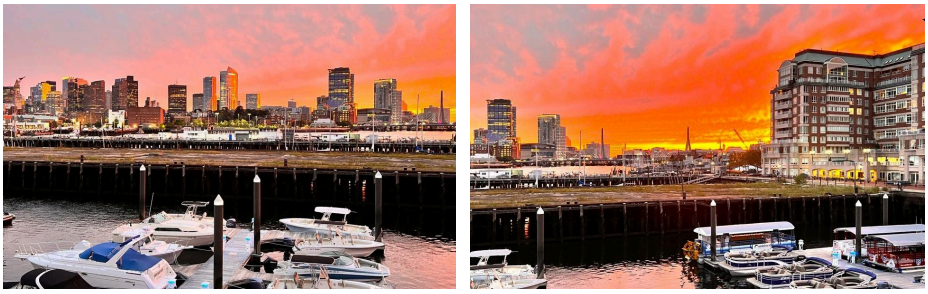

Figure 1. Input Images (Boston1.jpeg & Boston2.jpeg)

The images above were used as input images to generate the image below as the desired output.

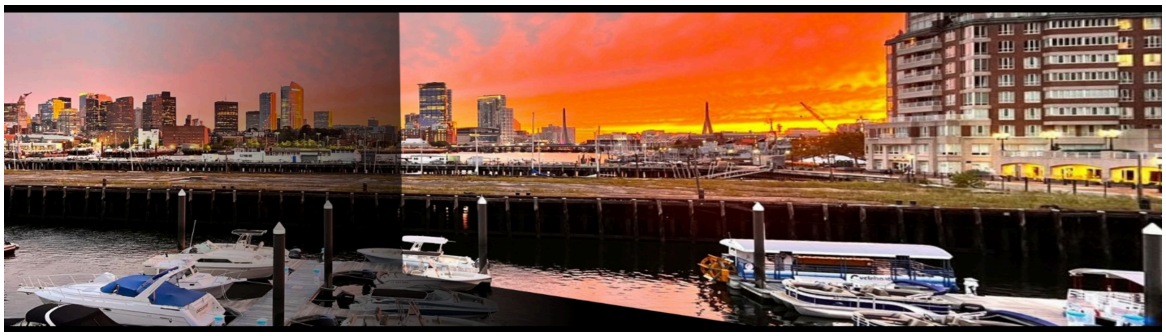

Figure 2. Panorama Output Image

Code Explanation:

First, we developed a method to create a panoramic image by stitching together two images using OpenCV. The process begins by loading the images and converting them to grayscale, which facilitates the feature detection. We utilized the Scale-Invariant Feature Transform to detect and extract keypoints and their descriptors from both images.

Using a FLANN-based matcher optimized by the Fast Library for Approximate Nearest Neighbors (FLANN) indexer, we matched the descriptors between the two images. Only the strongest matches, as determined by Lowe's ratio test, were stored to ensure the effectiveness of the match.

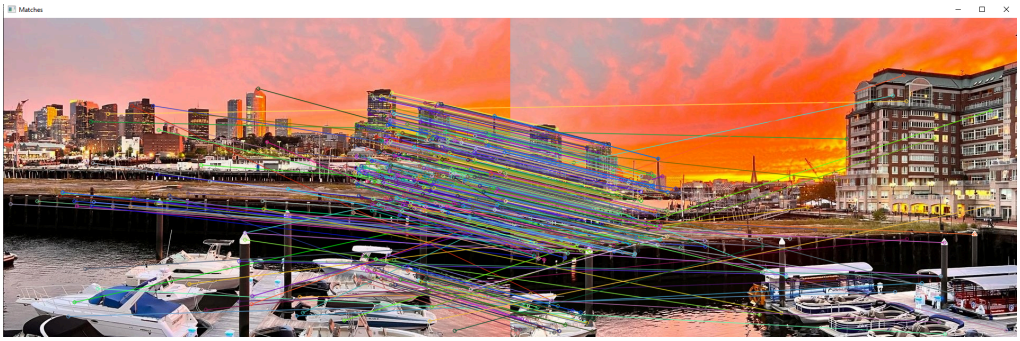

Figure 3. Potential Matches

For context, the image above was generated which shows the potential matches between the two input images.

From these matches, the source and destination points were extracted to compute a homography matrix with the RANSAC algorithm. This matrix defines the required transformations to align one image with another, addressing potential issues of rotation and scaling. The second image was then warped according to this homography, adjusting its perspective to match that of the first image.

The warped image was placed on a canvas sized to fit both original images. The first image was overlaid, using a mask to blend the images seamlessly wherever the warped image had content. If successful, the final panorama was displayed, integrating the two images effectively.

A key consideration was the size of the resulting panorama, which needed to be significantly larger than the individual images to accommodate the entire stitched scene without cropping.

Part 2:

Part 2 required us to improve the previous camera app to take at least three pictures and stitch them together using an optimized OpenCV stitching API.

Code Explanation:

This particular task was accomplished by integrating OpenCV's high-level stitching API. We developed a method that allows the user to capture several images in sequence by pressing a designated key, in this case ‘c’. These images are then automatically stitched into a panorama using the `cv2.Stitcher_create` function in panorama mode.

The process involves monitoring for the 'c' key press to capture images at defined intervals, storing them until a preset number of images are collected. Once the required number of images is reached, the app uses the Stitcher class to perform the stitching, handling alignment and blending automatically. The resulting panorama is displayed to the user, and the process can be repeated.

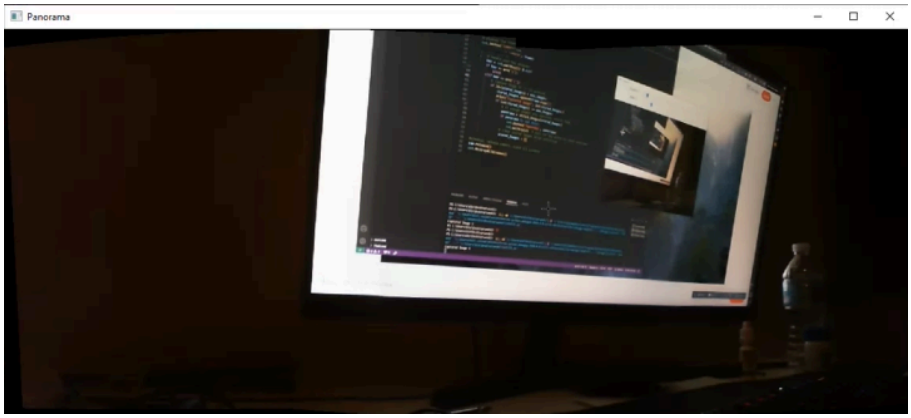

Figure 4. Potential Matches

As shown above, the generated panorama image has a much better quality that the panorama image from the previous part 1. Additionally, it performs well even with scenes with low luminosity as the one above.