Job1 Report - Hanzhen Qin

Implementation Overview

During the implementation, all grayscale images are first read from the specified folder and sorted in the order of the file name to ensure consistency in subsequent processing. For each image, the SIFT algorithm is used to extract key points and their 128-dimensional descriptors, and the number of key points detected is output for debugging and subsequent evaluation. Next, for each pair of images, the program uses a brute force matcher to perform k-NN matching, and uses the Lowe ratio test to filter out more reliable matching points. Using these matching points, the basic matrix is estimated by the RANSAC method to obtain an inlier mask to filter out matches that do not meet the geometric constraints, and the inlier matching results and corresponding epipolar lines are visualized. Subsequently, the program further estimates the homography matrix using the inliers of the basic matrix, and counts the proportion of inliers in the homography matrix to determine whether the two images can be accurately aligned; if the set threshold is reached, the images are considered to belong to the same scene and are stitched. The stitching process uses the cv2.warpPerspective function to map the image to a pre-calculated global canvas, and a simple weighted average method is used for regional fusion. For multi-image stitching, the program first constructs a connectivity graph between images, uses BFS to calculate the transformation matrix of each image relative to the selected anchor point, and determines the size of the final stitching canvas based on the corner coordinates of all images, and finally fuses all images to generate a complete panorama. The entire implementation process makes full use of the functions provided by OpenCV and the robustness of the RANSAC algorithm, which not only ensures the accuracy of matching and geometric correction, but also realizes the automated processing of multi-image stitching.

Analysis and Conclusion

The method presented in this project offers robust feature matching by employing SIFT for keypoint detection and RANSAC for both fundamental matrix and homography estimation. SIFT is well-known for its resilience to changes in scale, rotation, and moderate illumination variations, while RANSAC effectively discards outliers by identifying consistent geometric transformations. Through these combined techniques, the alignment accuracy is significantly improved, and the final mosaics can capture a wide field of view from multiple overlapping images. Additionally, by constructing a connectivity graph and selecting the largest connected component, the pipeline naturally extends beyond pairwise alignment to multi-image mosaics, allowing an entire set of images to be stitched together. Despite these strengths, the system also has certain limitations. When the overlap between images is minimal or the images themselves contain extensive low-texture regions, the number of reliable keypoints may be insufficient for stable alignment. Moreover, relying on a single homography restricts the pipeline to scenes that can be approximated by planar geometry; in scenarios involving significant parallax or pronounced three-dimensional structures, the mosaic may exhibit misalignment or distortion artifacts. Furthermore, the simple alpha blending strategy can lead to visible seams if the images have substantial differences in brightness or color balance. Several improvements could address these limitations. A more advanced blending technique, such as multi-band blending, would help conceal seams and provide smoother transitions between adjacent images. Incorporating additional feature detectors, like ORB or AKAZE,

might enhance robustness under diverse conditions or extreme scale changes. In truly complex or large-scale scenes, a global alignment strategy or bundle adjustment could be employed to more accurately model three-dimensional effects, thereby generating more seamless and geometrically consistent panoramas.

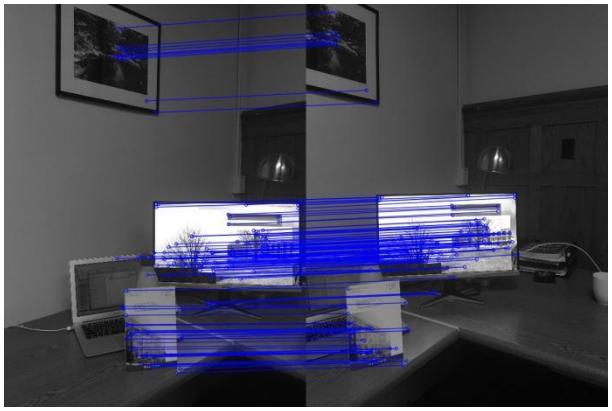
All in all, I can confidently say that I completed every step in the PDF. I generated all the images that needed to be generated, and all the images had the required image names. I also checked all the images, and each image looked very reasonable, and the final mosaic images also had very good stitching effects. Also I think I successfully implement the multi_image_mosaic function, which can merge more than two images. Therefore, I hope I can get extra credit for that.

Image Examples

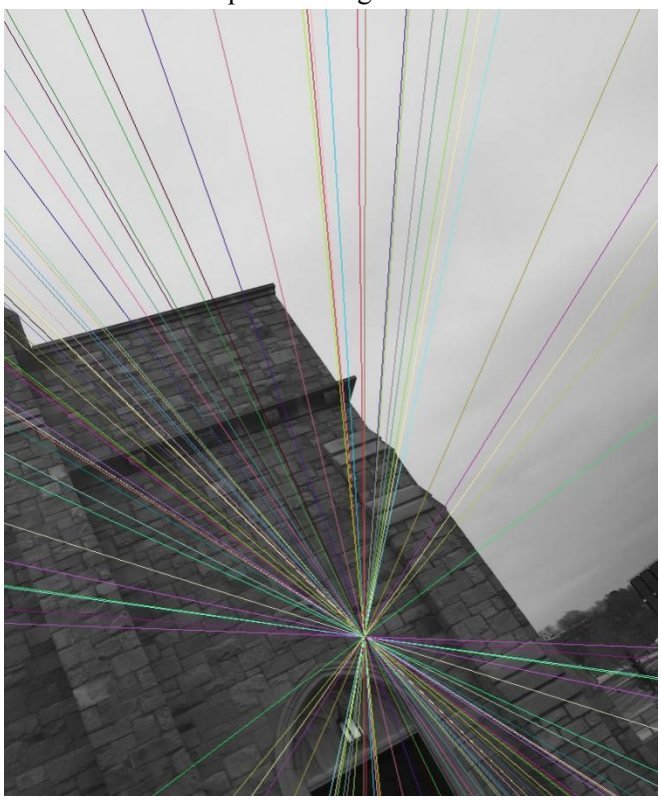
This is the multi-mosaic image for tree_mrc: Here I successfully merge four images to one tree by using the multi_image_mosaic function. I think it is pretty good for me.

This is one of the multi-mosaic images for frear-park

This is one of the inlier matches images for office



This is one of the epilines images for vcc-entrance



This is one of the multi-mosaic images for vcc-entrance

Result Statistics and Report

in_dir	drink-machine	frear-park	office	tree_mrc	vcc-entrance
out_dir	drink-machine-output	frear-park-output	office-output	tree_mrc-output	vcc-entrance-output
print info	drink-machine.txt	frear-park.txt	office.txt	tree_mrc.txt	vcc-entrance.txt

For drink-machine

image1: Detected 3607 keypoints

image2: Detected 4707 keypoints

image3: Detected 3683 keypoints

images	**matches**	**fraction1**	**fraction2**	**ratio F**	**ratio H**
image1 + image2	212	0.06	0.05	X	X
image1 + image3	59	0.02	0.02	X	X
image2 + image3	227	0.05	0.06	X	X

For frear-park

image1: Detected 668 keypoints

image2: Detected 863 keypoints

images	**matches**	**fraction1**	**fraction2**	**ratio F**	**ratio H**
image1 + image2	146	0.22	0.17	95.89%	92.86%

For office

image1: Detected 667 keypoints

image2: Detected 610 keypoints

image3: Detected 751 keypoints

images	**matches**	**fraction1**	**fraction2**	**ratio F**	**ratio H**
image1 + image2	229	0.34	0.38	82.97%	78.42%
image1 + image3	127	0.19	0.17	66.14%	54.76%
image2 + image3	95	0.16	0.13	58.95%	58.93%

For tree_mrc

image1: Detected 6860 keypoints

image2: Detected 7276 keypoints

image3: Detected 5997 keypoints

image4: Detected 5474 keypoints

images	**matches**	**fraction1**	**fraction2**	**ratio F**	**ratio H**
image1 + image2	1444	0.21	0.20	80.26%	88.87%
image1 + image3	392	0.06	0.07	X	X
image1 + image4	59	0.01	0.01	X	X
image2 + image3	1132	0.16	0.19	76.59%	88.35%
image2 + image4	186	0.03	0.03	X	X
image3 + image4	605	0.10	0.11	67.11%	87.68%

For vcc-entrance

image1: Detected 2337 keypoints

image2: Detected 4457 keypoints

image3: Detected 491 keypoints

images	**matches**	**fraction1**	**fraction2**	**ratio F**	**ratio H**
image1 + image2	897	0.38	0.20	78.82%	59.55%
image1 + image3	73	0.03	0.15	35.62%	65.38%
image2 + image3	143	0.03	0.29	44.06%	74.60%

Reference

OpenCV Documentation: https://docs.opencv.org