Panorama Stitching

Fernando Gómez <xgomez02@stud.fit.vutbr.cz>
Jordi Guimerà <xguime00@stud.fit.vutbr.cz>

27. December 2020

1 Introduction

In this project, we develop our own program which enables us to merge multiple photos taken from a single viewpoint into a panorama image.

Some better methods already exist nowadays, however, it serves as an exercise to further our understanding of feature extraction, feature matching and image manipulation, using python and OpenCV.

2 Theory

Our program shares its main outline with the existing panorama creators that exist nowadays, as shown in Figure 1.

The images to merge are taken as the input. We assume the pictures to be ordered from left to right and to present an overlap of at least 30% with the adjacent ones.

After reading the images, features are extracted from each one of them. In our case, we consider two different feature extracting algorithms implemented in OpenCV to briefly compare the results obtained using them afterwards. These two algorithms are SIFT (Scale Invariant Feature Transform) and ORB (Oriented FAST and rotated BRIEF), further explained in [L99] and [RRKB11], respectively.

Once the key points and their descriptors are obtained, correspondences between each pair of images are computed using an OpenCV implementation

of the Brute Force matching algorithm, which consists of a one by one comparison of the descriptors of each key point between each pair of images to find the best match for each key point. To find better matches we also implement the ratio test, introduced in [L04].

With the matches computed, we use an OpenCV implementation of the RANSAC algorithm (explained in [FB80]) to find the homography matrix between each pair of images. Using those homographies we create a blank canvas where the transformed images will be fitted. Then each image is transformed, using the corresponding homography matrix, to the center image's frame of reference and fitted into the canvas.

Finally, the output panorama is saved into the corresponding folder.

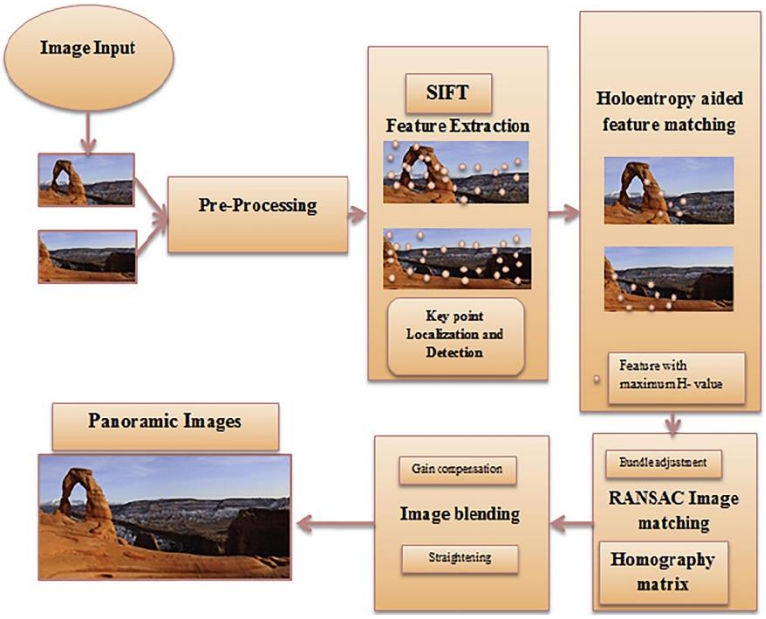

Figure 1: Outline of the panorama creating process [DMT18]

3 Evaluation

After successfully creating a panorama from multiple images, we conducted some tests to roughly determine which configuration works best for our program, given that we can only modify the key point algorithm used (SIFT or ORB) and the number of key points to search for.

First, using images of average resolution (< 200 KB per image) we measured

the time taken to execute the four main processes (shown in Table 1 and Table 2): key point and descriptor computation, matches computation, homographies computation and stitching process. As the computation time can be affected by other processes being executed at the time of testing, an average of three measurements is used.

Number of key points (ORB)	200	500	1000	5000	10000
Kp & Des computation time (s)	0.26162	0.288109	0.355049	0.353718	0.423444
Matches computation time (s)	0.006638	0.027571	0.162898	1.823076	6.599836
Homographies computation time (s)	0.003335	0.004654	0.012301	0.01261	0.019623
Stitching process time (s)	4.341989	7.25223	7.759748	7.600191	6.772401

Table 1: Average testing times of the program sections with small images (< 200 KB per image) using ORB.

Number of key points (SIFT)	200	500	1000	5000	10000
Kp & Des computation time (s)	1.501589	1.508879	1.558036	1.912677	2.002797
Matches computation time (s)	0.004321	0.024933	0.085606	2.010441	3.091475
Homographies computation time (s)	0.012781	0.01064	0.01276	0.022622	0.034232
Stitching process time (s)	5.584198	6.515455	6.56497	6.06009	6.1915

Table 2: Average testing times of the program sections with small images (< 200 KB per image) using SIFT.

As seen in Table 1 and Table 2, the homographies computation times and stitching process times are apparently independent to key point number change and appear to be similar in both algorithms, so that will not give us much information other than the need to optimize the stitching process efficiency. On the other hand, the key point and descriptor computation time increases slightly with key point increase, while the matches computation time increases drastically with the same increase of key points, more so for the ORB algorithm than the SIFT. However, SIFT is slower calculating the key points and descriptors.

Using ORB and working on implementing a faster matching algorithm (such as Flann Based Matching) could improve the overall performance of the program.

Looking at the results, the desirable option seems to be 5000 key points using SIFT (Figure 8). Below 500 key points should not be considered to ensure the correct functioning of the program.

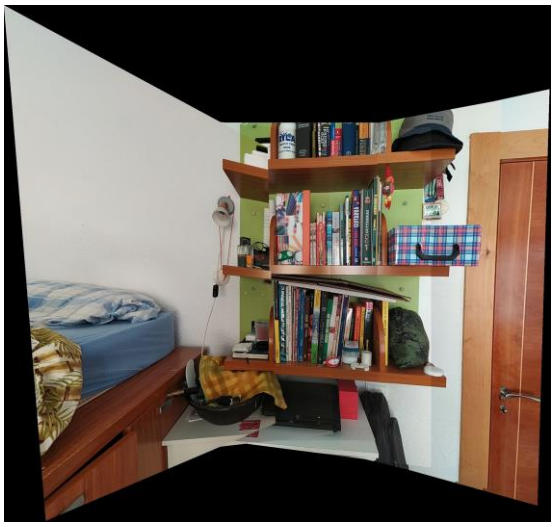

Figure 2: Resulting panorama using ORB and 500 key points

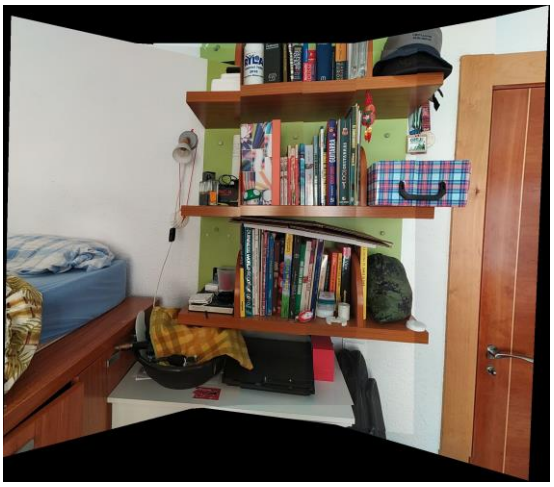

Figure 3: Resulting panorama using ORB and 1000 key points

Figure 4: Resulting panorama using ORB and 5000 key points

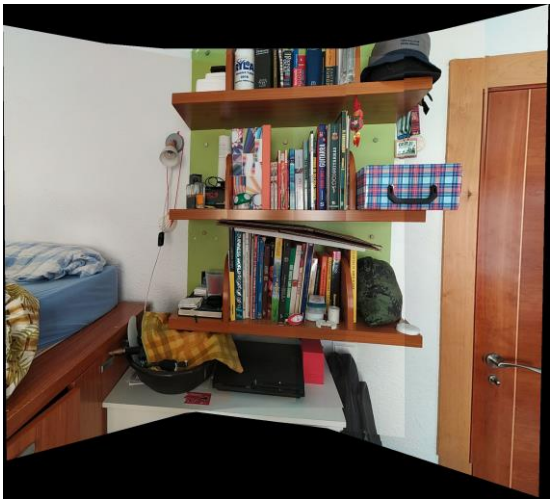

Figure 5: Resulting panorama using ORB and 10000 key points

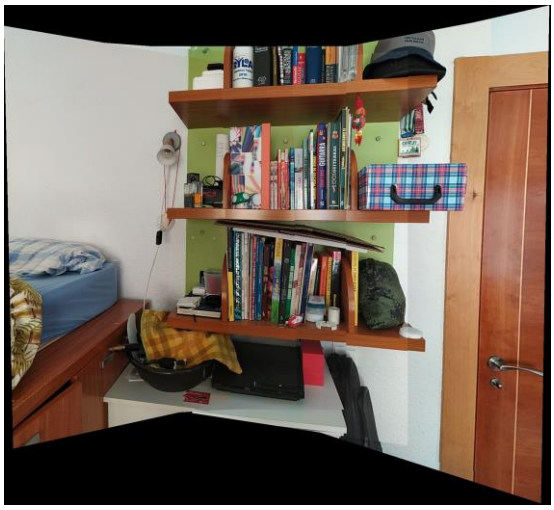

Figure 6: Resulting panorama using SIFT and 500 key points

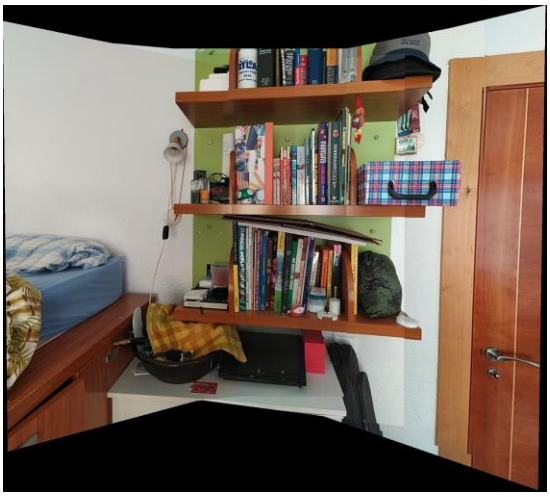

Figure 7: Resulting panorama using SIFT and 1000 key points

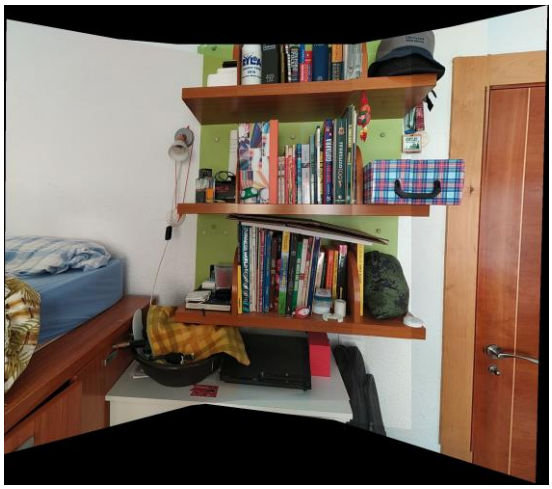

Figure 8: Resulting panorama using SIFT and 5000 key points

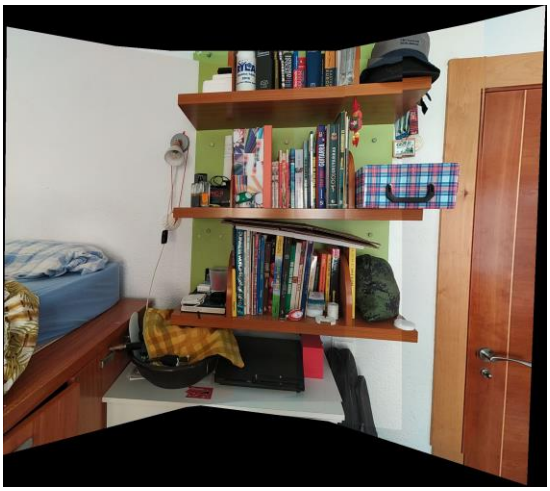

Figure 9: Resulting panorama using SIFT and 10000 key points

We have also tested the program with big images (> 2 MB per image), and the results show how big the impact of the resolution of the image has on the stitching process time, and how it also increases key point and descriptor computation time, especially with SIFT.

	SIFT 500	SIFT 5000	ORB 500	ORB 5000
Kp & Des computation	14.96613	16.21879	2.127309	2.238265
Matches computation	0.034907	2.39217	0.035904	1.967719
Homographies computation	0.021941	0.050373	0.020945	0.022944
Stitching process	83.14823	78.11571	81.46392	78.61721

Table 3: Testing times of the program sections with big images (> 2 MB per image)

4 Conclusion

Our program successfully creates a panorama image from multiple images given in order, by finding features with ORB, matching them, using RANSAC to compute the homography matrices, and then fitting all the images together in the center image's frame of reference.

It works decently for small images, but it needs serious optimization in the stitching process to be able to deal with big images.

Some improvements that could be made to the project are implementing a faster algorithm for feature matching, optimizing the stitching process, using blending to soften the seams between the pictures and cropping of the resulting image.

Reference

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski. ORB: an efficient alternative to SIFT or SURF. 2011

[L99] David G. Lowe. Object Recognition from Local Scale-Invariant Features. 1999

[L04] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. 2004

[DMT18] Ane Delphin, D., et al. Holoentropy measures for image stitching of scenes acquired under CAMERA unknown or arbitrary positions. Journal of King Saud University – Computer and Information Sciences (2018), https://doi.org/10.1016/j.jksuci.2018.08.006

[FB80] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. 1980