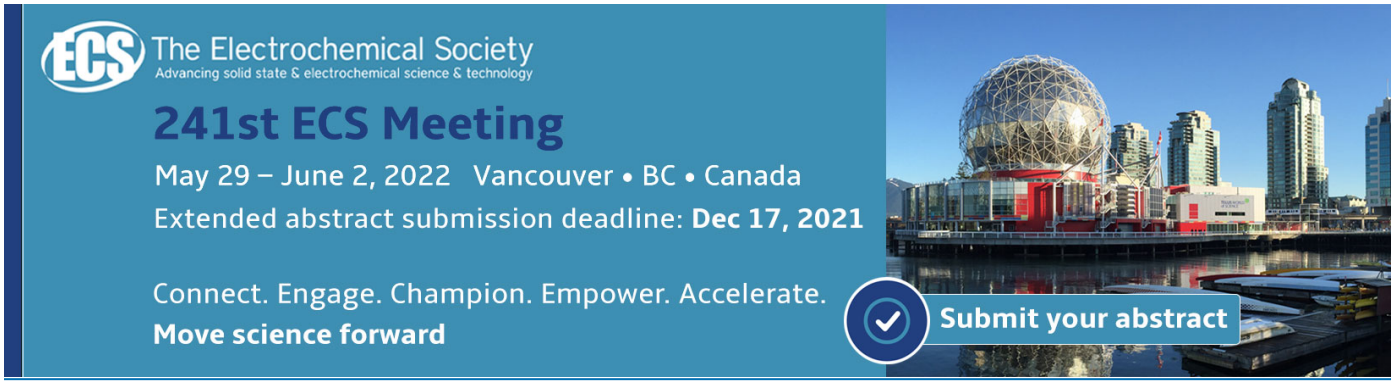
Real-time and automatic map stitching through aerial images from UAV



Real-time and automatic map stitching through aerial images from UAV

J N Goh, S K Phang* and W J Chew

School of Computer Science and Engineering, Taylor's University, 1, Jalan Taylors, 47500 Subang Jaya, Selangor, Malaysia

*E-mail: sweeking.phang@taylors.edu.my

Abstract. Real-time aerial map stitching through aerial images had been done through many different methods. One of the popular methods was a features-based algorithm to detect features and to match the features of two and more images to produce a map. There are several feature-based methods such as ORB, SIFT, SURF, KAZE, AKAZE and BRISK. These methods detect features and compute homography matrix from matched features to stitch images. The aim for this project is to further optimize the existing image stitching algorithm such that it will be possible to run in real-time as the UAV capture images while airborne. First, we propose to use a matrix multiplication method to replace a singular value decomposition method in the RANSAC algorithm. Next, we propose to change the workflow to detect the image features to increase the map stitching rate. The proposed algorithm was implemented and tested with an online aerial image dataset which contain 100 images with the resolution of 640 × 480. We have successfully achieved the result of 1.45 Hz update rate compared to original image stitching algorithm that runs at 0.69 Hz. The improvement shown in our proposed improved algorithm are more than two folds in terms of computational resources. The method introduced in this paper was successful speed up the process time for the program to process map stitching.

1. Introduction

In the traditional method to do mapping process, a satellite was used to generate the mapping from a high altitude [1]. This method has the disadvantage of not easy to use by civilian to do mapping for a specific region in real-time. An Unmanned Aerial Vehicle (UAV) can carry the high definition (HD) camera to capture the images from low altitude, the mapping process can be done by image processing software to stitch the image capture by UAV. The process for UAV to capture the image include:

1. Set the coordinate point to capture images.
2. Assign the UAV to fly through the coordinate point.
3. Mark the coordinate point with the image taken.
4. Mosaicking the image by using image processing software.
5. Generate mapping results.

The image processing software needs to do two main steps which are tilt correction and image mosaicking. The traditional method to generate the mapping results is time-consuming. In some conditions such as earthquakes, fire disasters, and floods, the rescue team needs to confirm the condition of the disaster immediately. The solution to generating a real-time map was needed. During the mosaicking phase, there are several methods to compare two similar pictures such as image-based,

object-based, least squares, and feature-based [2–4]. These researches used Kanade-Lucas-Tomasi (KLT) feature detection method and RANSAC algorithm to remove noisy matches points to do map stitching achieve a 5 Hz update rate. The research removes common appearance enhancements algorithm to increase the map stitching rate. The research proven the algorithm can be used in onboard UAV MasterMind processor to achieve real-time image stitching. The features-based has the advantage of stitching with discontinuous image and shadow shading. Hence, it was suitable for UAV mapping.

There are three famous feature-based methods such as Oriented FAST and Rotated BRIEF (ORB) [5], Scale-Invariant Feature Transform (SIFT) [6] and Speeded Up Robust Features (SURF) [7]. A research documented in [8] has shown the ORB is the fastest method among others feature-based algorithm. In 2018, Chen et al. used ORB algorithm to mosaic autonomic panoramic UAV images [9]. Another research done in 2018 by Ma et al. shown an improved ORB algorithm [10]. A research done by Northwestern Polytechnical university shows a result of an average 10Hz of processing rate to mosaic each frame from 1080p input video [11]. The algorithm used was ORB monocular SLAM. The Intel i7-4710 CPU with 16 GB RAM and GTX960 GPU was used for this research. The system of the computer is 64-bit Linux. In 2020, De Lima et al. has shown a result of an average 68.97Hz frame to frame processing rate for the 2.7k(2704×1521) resolution input image [12]. The method used was parallel hashing to match the binary ORB descriptor rapidly.

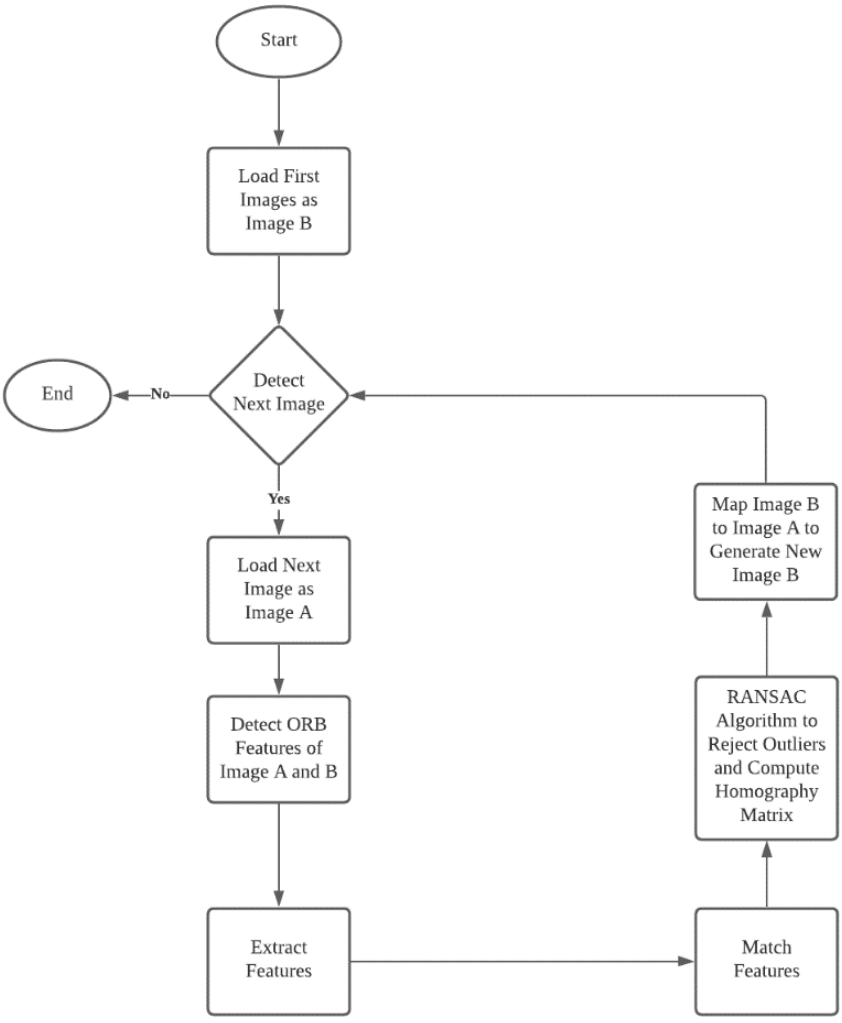


Figure 1. Map stitching workflow for ORB based features algorithm.

In this paper, we will focus on the ORB features-based method to do map stitching. Although the images are assumed to be fetched by an airborne UAV, the development and application for such UAV will not be discussed in this paper, as there are many documented work about it [13-15]. The general workflow of the map stitching process is shown in Figure 1. The base images B and next images A will first load to the program, ORB feature detector was used to detect the image features. Then, extract the descriptor of the features and match the feature base on the descriptor. Next, RANSAC algorithm was used to eliminate outliers and compute the homography matrix. After that, map image B to image A base on the homography matrix.

In the following sections, the original ORB and RANSAC algorithm, improvement of RANSAC, and the change of workflow will be explained in detail in methodology. The results and discussion section will compare and discuss the original ORB features-based method to stitch 100 images, ORB features-based with improved RANSAC algorithm to stitch 100 images, and ORB features-based with improved RANSAC algorithm and improved workflow to stitch 100 images. The input images are 640×480 resolution.

2. Methodology

A total of 100 pictures from open-source dataset capture by UAV was loaded to our algorithm. Three different codes were run 100 times to get the percentage of good quality images and average map stitching rate due to the algorithm contain random variable. The three different codes are original ORB features-based method to stitch 100 images, ORB features-based with improved RANSAC algorithm to stitch 100 images, and ORB features-based with improved RANSAC algorithm and improved workflow to stitch 100 images.

2.1. ORB Features-Based Algorithm

ORB features-based algorithm is a combination of Oriented FAST and Rotated BRIEF. Oriented FAST is modified from FAST algorithm. In ORB FAST, a pixel P is chosen from a grayscale image shown in Figure 2 [16], for P as the centre, compare the value of the to the surrounding 16 pixels with the radius of 3. If there are consecutive n pixels on the circumference, the grayscale value is larger or smaller than the grayscale value of the P, then p is considered features point. The value of n normal set as 12. To speed up this process, the algorithm will first compare the position 1, 5, 9, 13 with the P. If at least 3 or more of these pixels are greater or lesser to the P, then continue to compare other positions, else it will exclude this point. The algorithm will loop through all the pixels in the image.

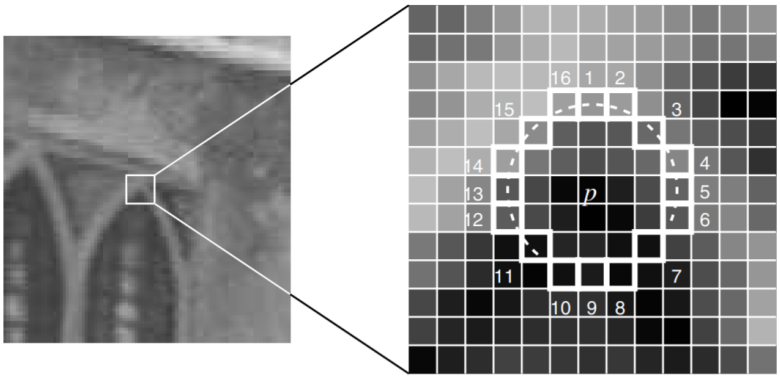


Figure 2. Fast 9-16 Method.

Next, the Non-Maximum Suppression was used to remove multi-features in a near location. The absolute sum of the difference between P with other 16 pixels was computed. The feature that has the highest value of the absolute sum of different will be chosen as the features point and others will eliminate.

To make sure the features point is scale-invariant, which mean if the camera takes the photo with different distance between the camera and object, the same features point can be calculated. Hence, the image pyramid was employed to make the image have a different scale.

First, the gaussian blur applied. This is a convolution of the gaussian blur to the image. The mathematical formula was shown in Equation (1).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)S \tag{1}$$

L is a blurred image. G is the gaussian blur operator show in Equation (2) and I is the image. The x and y are coordinates of the pixel. σ is the scale factor, where higher value result in a more blurred image.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\left(\frac{-(x^2+y^2)}{2\sigma^2}\right)} \tag{2}$$

After applying gaussian blur, the image was down-samples from the original image with the scale of half for each layer to create n layer scale image. In each layer, the Harris filter was used to reject the edge to get the corner point. To make sure the features point is rotationally invariant, the gray-scale centroid method was used to calculate the direction of the features point.

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{3}$$

m is the moment and the formula to calculate it shown in Equation (3), I is the grayscale image. The centroid C is shown in Equation (4).

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{4}$$

Assume the center is O, the angle $\overrightarrow{OC}$ is the direction of the feature point. The angle can be calculated by Equation (5).

$$\theta = \arctan(m_{01}, m_{10}) \tag{5}$$

A descriptor was needed to match the features point, the rBRIEF is the algorithm to calculate the descriptor. The rBRIEF is modified from BRIEF.

$$\tau(p: x, y) \coloneqq \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) \geq p(y) \end{cases} \tag{6}$$

In Equation (6) the grayscale intensity of p(x) is compared to p(y), if it is greater than its return value 1, else return 0. BRIEF algorithm randomly chooses n pair coordinate around the feature point to compare to get binary value. For n pair coordinate, can get n bits binary descriptor in Equation (7).

$$f_n(p) \coloneqq \sum_{1 \leq i \leq n} 2^{i-1} \tau(p: x_i, y_i) \tag{7}$$

To make sure the descriptor is rotational invariant, the n ×2 pair coordinate store in matrix s, shown in Equation (8).

$$s = \begin{pmatrix} x_1 & , \dots , & x_n \\ y_1 & , \dots , & y_n \end{pmatrix} \tag{8}$$

Then, the rotational angle gets from Equation (5) was multiply to matrix s to get rotational matrix s_θ, shown in Equation (9).

$$s_\theta = R_\theta s \tag{9}$$

Hence, we get Equation (10).

$$g_n(p, \theta) \coloneqq f_n(p) | (x_i, y_i) \, \epsilon s_\theta \tag{10}$$

In BRIEF, the value compares by the n pair coordinate is not directly comparable to the grayscale intensity but compares the average grayscale intensity around the coordinate. This will reduce the noise effect. The n value in rBRIEF was chosen based on a greedy search from a 300k pair dataset, to get a 256 set of comparisons that is high variance and low correlation in between to allow easy to differentiate the 256-bit descriptor. After getting the descriptor, the descriptor can compare by

calculating the Hamming distance to get the match features. If the hamming distance is lower than the threshold value, then the two features compared are matched.

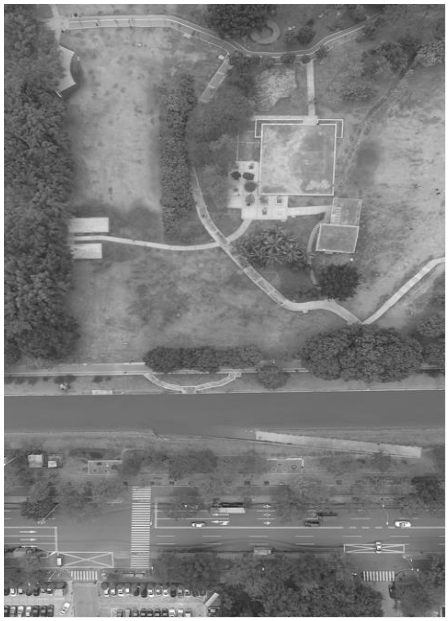


Figure 3. Gray Scale Image A.

Figure 4. Gray Scale Image B.

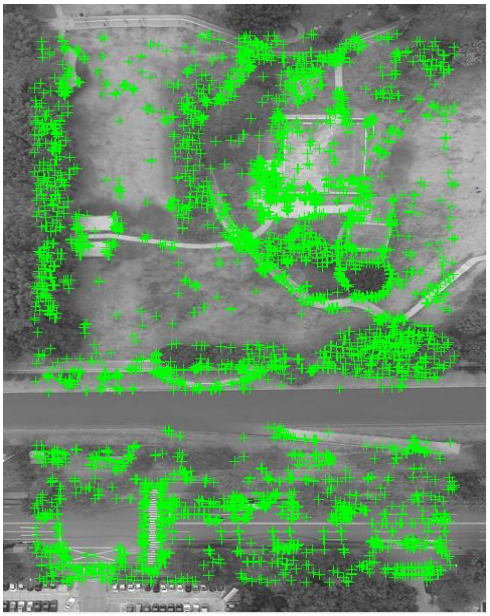


Figure 5. Gray Scale Image A with ORB Feature Points Plotted.

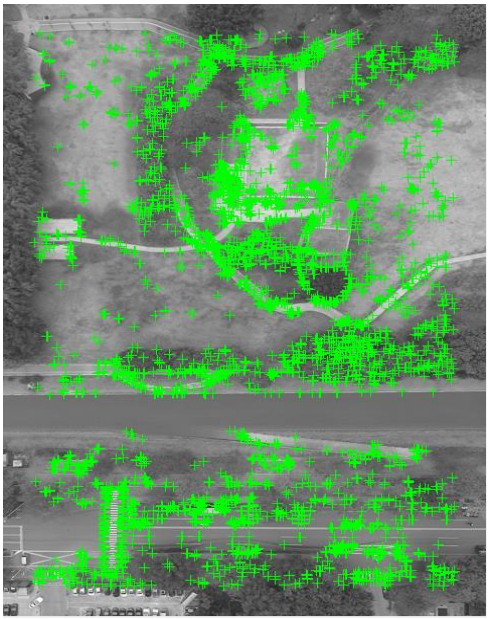


Figure 6. Gray Scale Image B with ORB Feature Points Plotted.

In Figure 3 and Figure 4, the first 2 loaded image change to gray scale image. Next, Figure 5 and Figure 6 are the gray scale image with ORB features plotted.

2.2. RANSAC Algorithm

After performing the matching process, some matching is incorrect. Hence, the RANSAC algorithm was used to reject outlier and to compute the homography matrix. RANSAC algorithm for 2-dimensional mapping got 5 main steps:

1. Random choose 4 set matched features point that will not cross each other as hypothesized inliers, because in the Equation (11) the model has 8 unknow, need at least 8 linear equations to solve, 1 set match point can give 2 linear equations.
2. Fit the model with the 4 set match features point to calculate the homography matrix, H.
3. After getting the matrix H, fit the rest of the match features point to Equation (11). Then, classify the point as inlier or outlier. If few inliers found for this model, then this model was failed.
4. Fill in the model with hypothesized inliers and classified inliers.
5. Evaluate this model by calculating the residual error of all inliers. The equation to calculate residual error show in Equation (12).

Repeat step 1 to 5 until get the lowest residual error model. Then the model is the best for mapping.

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{11}$$

$$\sum_{i=1}^{n}\left(x_i \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 + \left(y_i \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 \tag{12}$$

At the end use the homography matrix to map one image to another image.

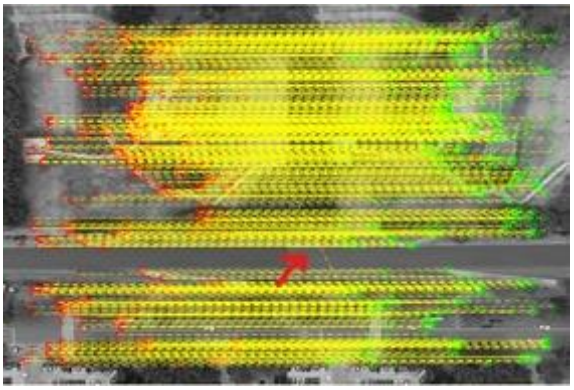


Figure 7. Gray Scale Image A and B with All Match Points.

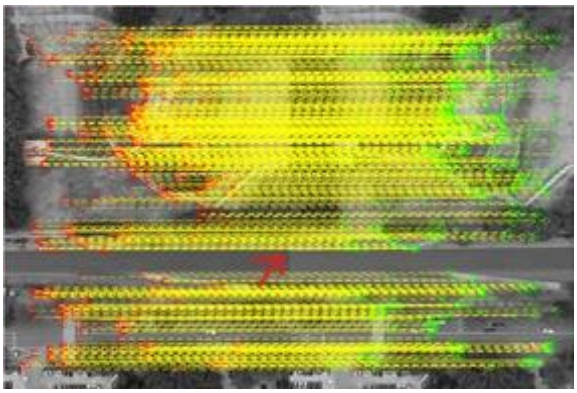


Figure 8. Gray Scale Image A and B with Inlier Match Points.

Figure 7 show the match points contain a little outlier. After applied RANSAC algorithm, the Figure 8 show the match point without the outliers.

2.3. RANSAC Algorithm improvement and workflow modification

There are two modifications made for the code compares to original ORB based method. First, RANSAC algorithm use matrix multiplication instead of for loop. In MATLAB software, it has fast computational speed at double-precision general matrix-matrix multiplication. This is because

MATLAB uses high optimized libraries such as Linear Algebra Package (LAPACK) and Basic Linear Algebra Subprograms (BLAS) there are good caching of data.

There is two part in original code was writing by using for loop. First, to calculate the estimate homography matrix (H). Second is map the matched coordinate image A by using H to a imaginary image B, then calculate the distance of the coordinate point between imaginary image B and real image B.

For the first part, assume the N values of matched features was chosen. Then the $2N+1$ simultaneous equation can be solved by using a N rows times $2N+1$ column matrix. The original code filled in the $\frac{1}{N}$ rows by calculating the values from N matched features coordinates. After filled in, then solve the matrix by using singular value decomposition to solve the matrix. The improved code, filled in the nine simultaneous equations by using the index number to fill the equation all in one time. Then, the 2N rows $2N+1$ columns matrix was multiplying with the transpose matrix itself to get a square matrix. Then, eigenvalue decomposition was done to solve the matrix. The results will be same for a square matrix calculated by eigenvalue decomposition method compares to singular value decomposition.

For the second part, the original code was map one coordinate of matched point image A to imaginary image B for each round of the for loop. The improved code directly maps all the n matched point image A to imaginary image B and store in a 2 rows n columns matrix. The matched point can be thousand number. Hence this method will reduce significant amount of computational time by changing thousand loops to a single multiplication matrix. The flow chart of the original RANSAC code and improved RANSAC code shown in Figure 9 and Figure 10.

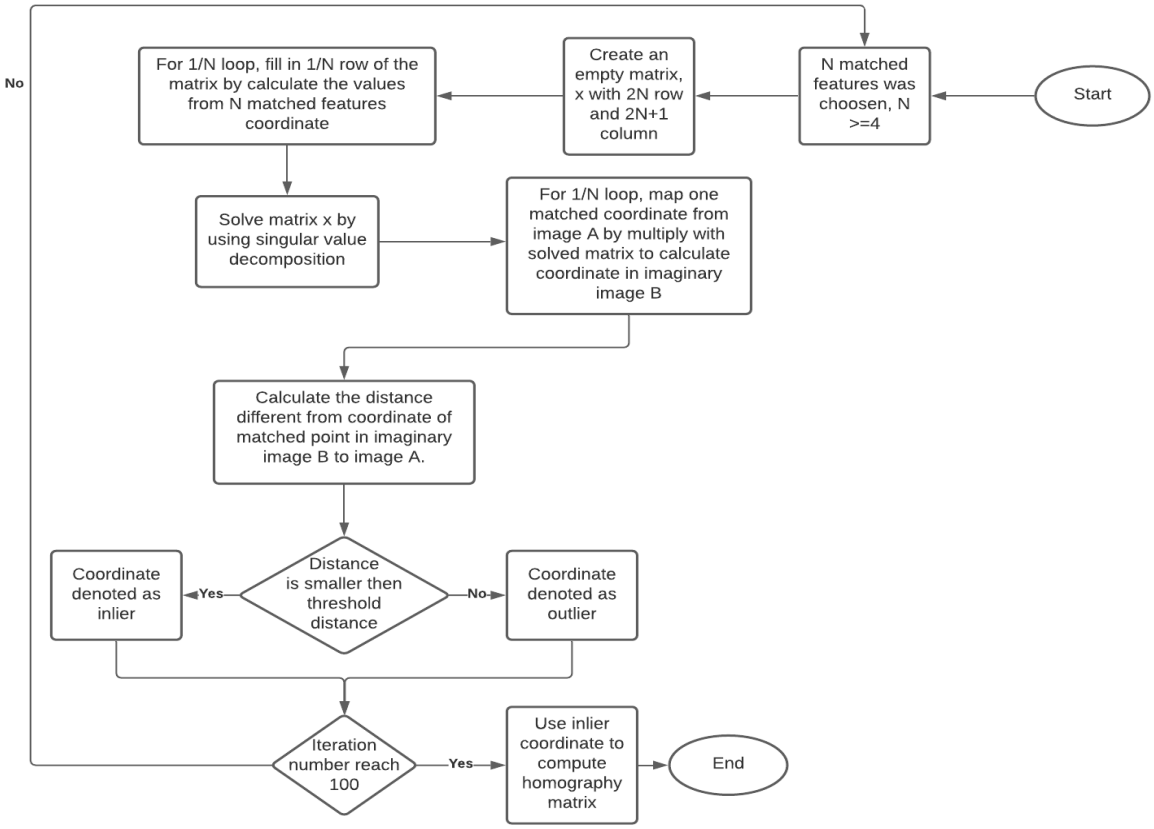


Figure 9. Original RANSAC Code Flow Chart.

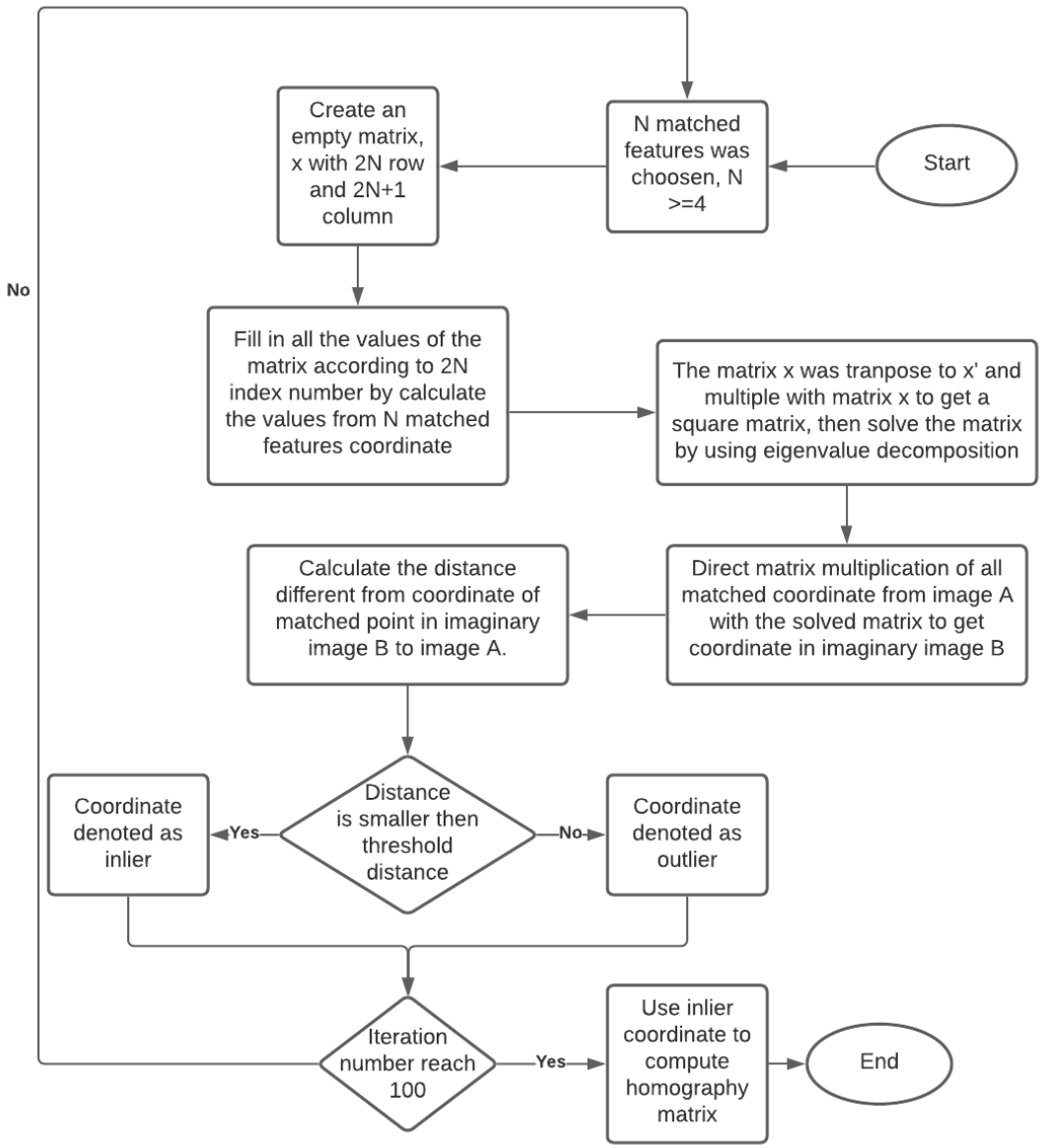


Figure 10. Improved RANSAC Code Flow Chart.

Second, when the stitched map becomes larger, the time for the ORB detector to detect the features from the stitched map increase. Hence the workflow was changed as shown in Figure 11. If the image number more than 20, the incoming image A will split in half. Because the newly refresh area is on the right side of the map. Hence, the ORB detector only needs to detect the right side of the stitched map. Note the stitched map is image B show in the workflow.

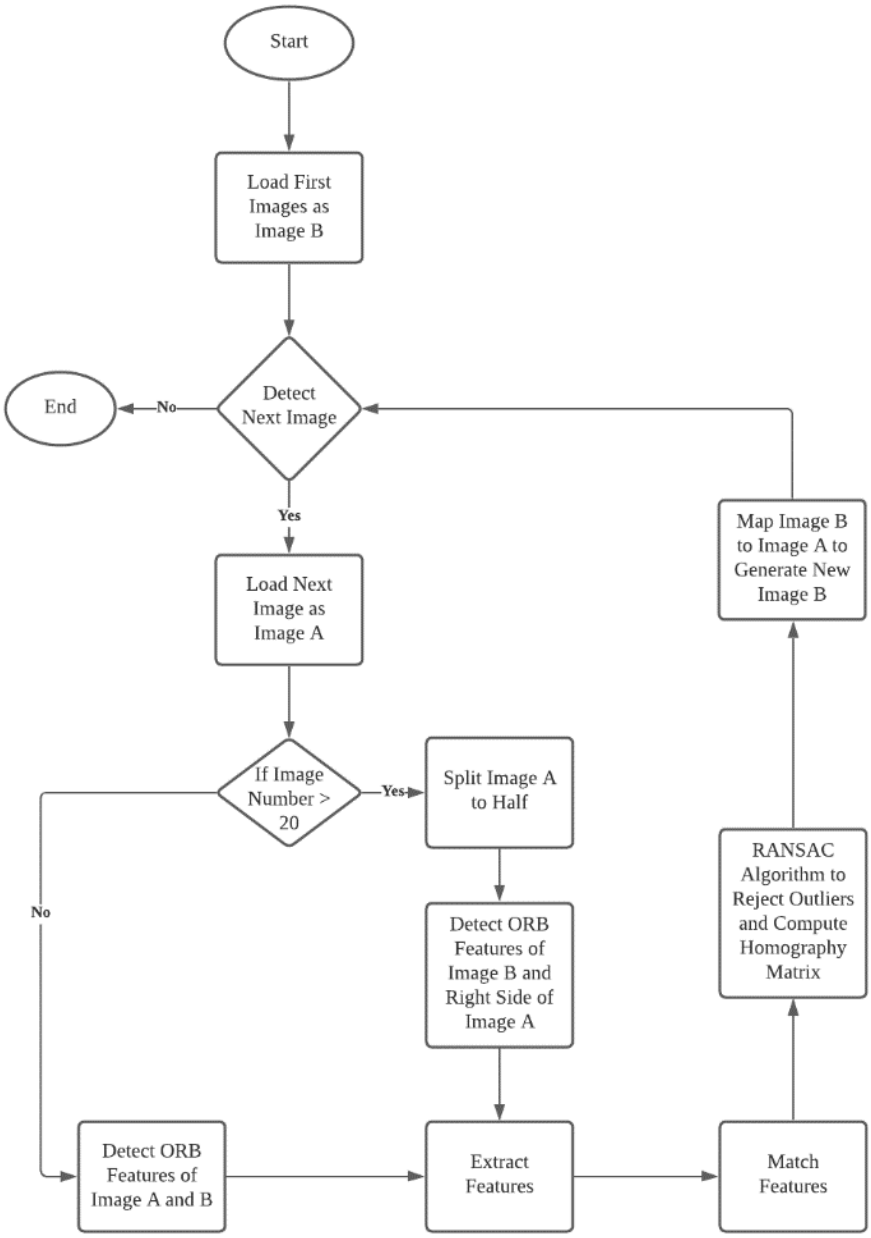


Figure 11. Improved Map Stitching Workflow.

3. Results and Discussion

One out of hundred for each of the results from Original ORB Features-based Method, ORB Features-based with improved RANSAC Algorithm, and ORB Features-based with improved RANSAC Algorithm and improved Workflow to stitch 100 images were shown in Figure 12, Figure 13 and Figure 14 respectively. The stitched map shown are clear and without deform. The results were denoted as good quality images. The image with deformation in scale or very blur will be denoted as poor-quality images.

Figure 12. Original ORB Features-based Method to Stitch 100 Images

Figure 13. ORB Features-based with Improved RANSAC Algorithm to Stitch 100 Images

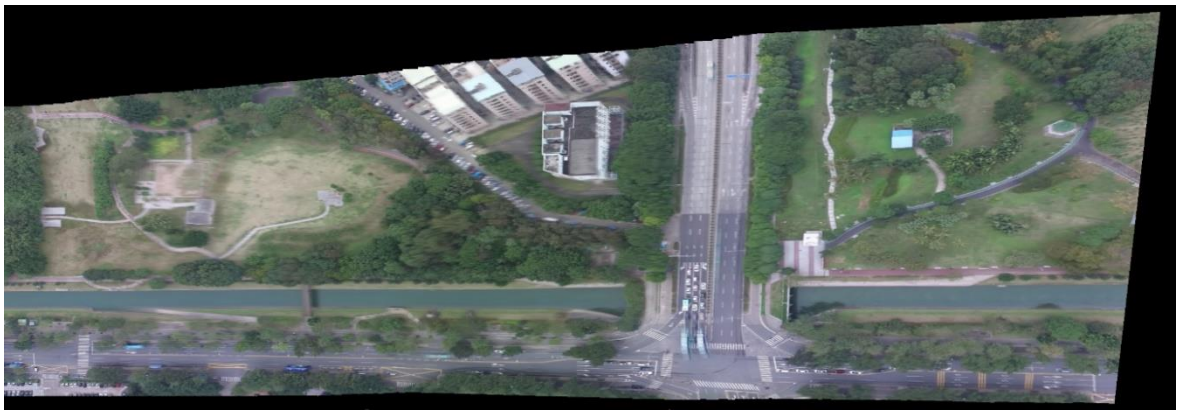


Figure 14. ORB Features-based with Improved RANSAC Algorithm and Improved Workflow to Stitch 100 Images

Table 1. Processing Time for Five Process in Original and Improved Algorithm.

Algorithm	Number Of Iteration		Average Time Taken To				
	RANSAC Algorithm	Program	Detect & Extract Image A Features, t(s)	Detect & Extract Image B Features, t(s)	Match Features, t(s)	Compute RANSAC algorithm, t(s)	Map Image B To Image A, t(s)
ORB Features Based Algorithm	100	100	0.03419	0.05847	0.23476	0.79562	0.18541
Improved RANSAC Algorithm	100	100	0.03482	0.06857	0.27101	0.027	0.20054
Improved RANSAC and workflow Algorithm	100	100	0.03544	0.04467	0.22287	0.02637	0.21736

From Table 1, it shown the RANSAC algorithm number of iteration and number of program iteration were both set to 100 in original and improved algorithm. It was clearly to note that the average time taken to compute RANSAC algorithm was decrease significantly from 0.79562s to 0.027s when the RANSAC algorithm had been improved. This will lead to increase the map stitching rate shown in Table 2. However, the average time taken to detect and extract image A and B features, to match features, and to map image B to image A were slightly increase from 0.03419, 0.05847, 0.23476, and 0.18541s to 0.03482, 0.06857, 0.27101, and 0.20054s respectively. This was due to the RANSAC algorithm contain random variable to compute the homography matrix, then the slightly different matrix values computed will cause the stitched image map slightly different. In result after stitched images number increase the precision of the images will decrease. Hence, the average time taken increase. In summary, the average time taken to compute RANSAC algorithm decrease 96.6%.

For the change of workflow, the average time taken to detect and extract image B features was decrease from 0.06857s to 0.04467s. This will also lead to increase the map stitching rate shown in Table 2. Also, it will lead to decreasing in average time taken to match features from 0.27101 to 0.22287s. Because of the decrease in number of features extracted, the homography matrix values will be different. This will cause other process time taken to had slightly different. In summary, the average time taken to detect and extract image B features was decrease 34.85%. However, due to its original time taken was small, it only contributes to decrease the time 0.02s in overall.

Table 2. Final Results for Original and Improved Algorithm

Algorithm	Average Total Time Taken To Stitch Two Images, t(s)	Map Stitching Rate, Hz	Good Quality Of Image,%
ORB Features Based Algorithm	1.44082	0.69	99
ORB Features Based with Improved RANSAC Algorithm	0.73769	1.36	100
ORB Features Based with Improved RANSAC and workflow Algorithm	0.69161	1.45	100

From Table 2, the good quality of the map stitched are all 99% and above. This mean in overall 100 times of running the program at least 99 times the stitched images were similar to the good quality images were shown in Figure 12, Figure 13 and Figure 14. The poor-quality images were the image with deformation of scale or critical blur. The improved RANSAC make the map stitching rate increase nearly double from 0.69 Hz to 1.36 Hz. The improved workflow makes the map stitching rate increase little from 1.36 Hz to 1.45 Hz. This was important to enable people with the MATLAB programming skill to use MATLAB software to write a program to stitch the map from the large number of images transfer from UAV in real-time.

4. Conclusion

In this paper, RANSAC algorithm was improved to change the for loop to matrix multiplication. This method reduces 96.6% of the processing time to compute the homography matrix. For stitched large number of images, the structure of the stitching process was improved to add an identifier to identify if the image number more than 20, the program will split the input images to half. Since the images size reduce to half, the time taken detect and extract the features reduce 34.85%. The improved RANSAC and workflow was successful to increase the map stitching rate from original 0.69 Hz to 1.45 Hz. However, the images input to the program is (640×480) resolution, this paper did not examine the performance of the program for high quality input images. Furthermore, the improved structure of the workflow is specific for the drone fly in one direction straight and small angle swing without any large angle change of direction or reverse. The results may be different, or the program may do not work for the drone fly in free direction. The results obtained was using a laptop with Intel i7-8550U CPU and 8GB RAM. Other laptop with different specification for example higher RAM memory will decrease the large matrices computational time. Hence, it will increase map stitching rate with the same code. Future research may focus on speed up the map stitching rate with input high resolution images and the drone fly in free direction. The structure of the workflow can be further improved for specified structure used for specified drone flying direction.

References

[1] Stratoulias D, Tolpekin V, De By R A, Zurita-Milla R, Retsios V, Bijker W, Hasan M A and Vermote E 2017 A Workflow for Automated Satellite Image Processing: from Raw VHSR Data to Object-Based Spectral Information for Smallholder Agriculture *Remote Sens.* **9**

[2] Cui J Q, Phang S K, Ang K Z Y, Wang F, Dong X, Ke Y, Lai S, Li K, Li X, Lin F, Lin J, Liu P,

Pang T, Wang B, Wang K, Yang Z and Chen B M 2015 Drones for cooperative search and rescue in post-disaster situation *2015 IEEE 7th Int. Conf. on Cybernetics and Int. Sys. (CIS) and IEEE Conf. on Robotics, Automation and Mechatronics (RAM)* (Siem Reap: IEEE) pp 167–74

[3] Phang S K, Cui J, Ang Z Y K, Wang F, Dong X, Ke Y, Lai S, Li K, Li X, Lin F, Lin J, Liu P, Pang T, Wang B, Wang K, Yang Z and Chen B M Urban Post-Disaster Search and Rescue Solutions with Unmanned Aircraft Systems *Proc. of the 2015 Int. Conf. on Electronics, Information and Communication* (Singapore: ICEIC) pp 91-2

[4] Cui J Q, Phang S K, Ang K Z Y, Wang F, Dong X, Ke Y, Lai S, Li K, Li X, Lin J, Liu P, Pang T, Wang K, Yang Z, Lin F and Chen B M 2016 Search and Rescue Using Multiple Drones in Post-Disaster Situation *Unmanned Syst.* **04** 83–96

[5] Rublee E, Rabaud V, Konolige K and Bradski G 2011 ORB: An efficient alternative to SIFT or SURF *2011 Int. Conf. on Computer Vision* (Sophia Antipolis: Lecture Notes in Computer Science) pp 2564–71

[6] Zhang Y H, Jin X and Wang Z J 2017 A new modified panoramic UAV image stitching model based on the GA-SIFT and adaptive threshold method *Memetic Comput.* **9** 231–44

[7] Bay H, Tuytelaars T and Van Gool L 2006 SURF: Speeded Up Robust Features *Computer Vision -- ECCV 2006* ed A Leonardis, H Bischof and A Pinz (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 404–17

[8] Tareen S A K and Saleem Z 2018 A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK *2018 Int. Conf. on Comput., Mathematics and Engineering Technologies -- iCoMET* (Pakistan, Sukkur: IEEE) pp 1–10

[9] Chen J, Luo L, Wang S and Wu H 2018 Automatic Panoramic UAV Image Mosaic Using ORB Features and Robust Transformation Estimation *2018 37th Chinese Control Conf. -- CCC* (China, Wuhan: IEEE) pp 4265–70

[10] Ma C, Hu X, Fu L and Zhang G 2018 An Improved ORB Algorithm Based on Multi-Feature Fusion 2018 *IEEE 27th Int. Symposium on Industrial Electronics -- ISIE* (Cairns, QLD: IEEE) pp 729–34

[11] Bu S, Zhao Y, Wan G and Liu Z 2016 Map2DFusion: Real-time incremental UAV image mosaicing based on monocular SLAM *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems -- IROS* (Daejeon: IEEE) pp 4564–71

[12] De L R, Cabrera-Ponce A A and Martinez-Carranza J 2021 Parallel hashing-based matching for real-time aerial image mosaicing J. *Real-Time Image Process.* **18** 143–56

[13] Phang S K and Chen X 2021 Autonomous tracking and landing on moving ground vehicle with multi-rotor UAV *J. Eng. Sci. Technol.(JESTEC)* **16** 2795–815

[14] Sidik R and Ray Y 2021 Interactive map-based optical distribution point (ODP) mapping design and implementation *J. Eng. Sci. Technol.(JESTEC)* **16** 2128–38

[15] Li J, Goh W W and Jhanjhi N Z 2021 A design of IOT-based medicine case for the multi-user medication management using drone in elderly centre *J. Eng. Sci. Technol.(JESTEC)* **16** 1145–66

[16] Rosten E and Drummond T 2006 Machine Learning for High-Speed Corner Detection *Computer Vision -- ECCV 2006* ed A Leonardis, H Bischof and A Pinz (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 430–43