Implementation of an Image Stitching Algorithm to a Low-Cost Digital Microscope

Renan Botan
Universidade Federal do Espírito Santo
Instituto Federal do Espírito Santo
Email: renanbotan@gmail.com

Klaus Fabian Coco
Universidade Federal do Espírito Santo
Email: klaus.coco@ufes.br

Karin Satie Komati
Instituto Federal do Espírito Santo
Email: kkomati@ifes.edu.br

***Abstract*—In this work we propose an implementation approach to a feature-based algorithm that stitch images which comes form MicroScanner, a low cost digital microscope created at Mogai Information Technology. This program solve the trade-off problem between field of view and resolution. These codes were developed in C++ language, with help of OpenCV library. The sequence of pictures taken is used by the program to reduce computing costs and increase reliability of the resulting mosaic. In addition, the results achieved maintain the quality from original images, and presents better quality and more effective results than previous related works.**

I. Introduction

Since its invention in the end of sec XVI, the optical microscope has its fundamental role in many different science fields, in the discovery of diseases, proofing scientific methods, in cellular studies, among many others applications in biology, medicine, and exact sciences [1]. Since the second half of the twentieth-century microscopes have been also used in the assembly of electrical, electronic and mechanical microsystems and will continue to be an indispensable tool for the advancement, growth, and development of these areas [2].

In the last decades, the evolution of the optical microscope has begun to decrease due to the fact that optical limits were reached. It has been found that the maximum resolution of an optical microscope is half the wavelength of light that illuminates the sample; unless fluorescence techniques are used [3]. For example, if the light in used has a wavelength of $400\eta m$ (blue light) the maximum resolution of the microscope will be $200\eta m$, but these values are rarely obtained due to limitations of the material of the lenses.

In1931 the German physicist Ernst Ruska developed the electronic microscope that does not use light to focus the sample, but rather electron beam, which increased the resolution and surpassed the optical limits. Today there are electronic microscopes that reach resolutions up to $0.1\eta m$ [2].

However, the optical microscopes are still being widely used, but most of the time users are forced to trade-off between a high resolution, with lenses capable of increasing the size of things in tens of thousands of times or to have a bigger field of view, with lenses that extend less. Although, the advent of computers, fast processors, big memory and abundant storage space, made possible the development of digital microscopes, and a possibility of solution to the problem between viewed area and resolution through image stitching algorithms [2].

The biggest problem of the digital microscopes now available in the market is that they are considerably expensive [3]. Seeing a market opportunity, the company Mogai Information Technology raised founds to develop the MicroScanner a low-cost microscope with final price estimated to be ten times cheaper than its actual competitors.The idea behind MicroScanner's is to have an optical/digital microscope connected to a cloud. The device can be taken to remote places, such as in the interior of the Amazon, to photograph the samples and upload then automatically to a cloud, allowing the images to be accessed via a browser anywhere in the world by a computer, tablet or smart phone, as long it is connected to the internet. Other than being cheaper, this product solves the displacement problem of specialized labor, which is expensive. Instead, a technician can go to this remote places to collect the samples, and a specialist can analyses it in another country for example, reducing research costs and avoiding sanitary barriers.

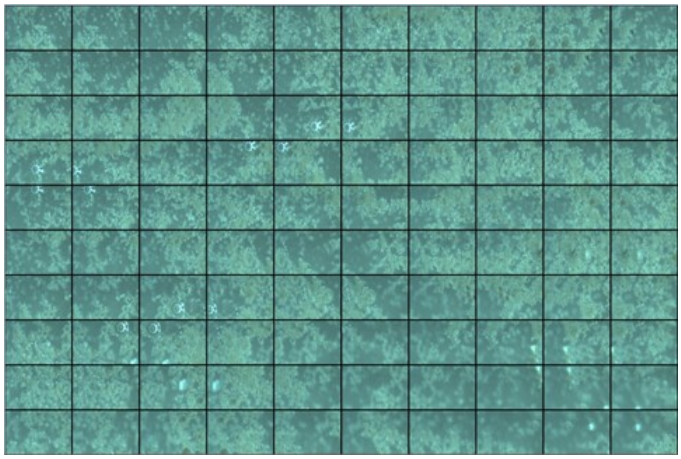


Fig. 1. Matrix with 10x10 images with 2 Mpixels each scanned form a sample of industrial sewer.

In this work we present an implementation approach to a feature-based stitching algorithm to create mosaics with MicroScanner's images to solve the issue among viewed area and resolution. This software will be supposed to run on a cloud server and stitch automatically images that will come in, such as the 100 images shown in Fig. 1, maintaining the quality of the original images, in other words, the final mosaic

will have a bigger resolution than the original images. To achieve the results we are presenting in this paper were used C++ coding and methods from OpenCV library [4].

There is an extensive number of mosaics work already published, such as [5], [6], [7], [8], and some commercial applications [9], [2], [10]. However, in this work we aim to obtain more precise mosaics, and with smoother transitions than [7], trying to achieve a result similar to [10].

II. Proposed Approach

The stitch code we developed in this work follows the schematic presented in Fig. 2. Each step will be discussed in the following subsections.

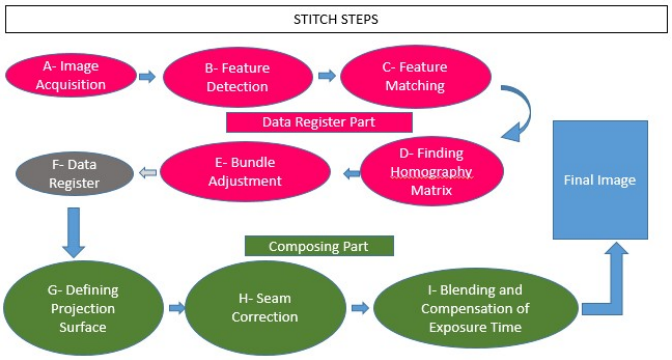


Fig. 2. Steps of the stitching process.

A. Image Acquisition

The MicroScanner works scanning samples of regions such as the one presented in Fig. 1. In this matrix of images, initially 10 pictures were taken form reader's left to reader's right, then the sample is moved one step down, and this time 10 pictures are taken from reader's right to reader's left. This process is repeated while the 100 images from Fig. 1 are not taken. Differently form [10] that has an algorithm to automatically find neighbor images, our work uses this sequence to know which images are neighbors. This procedure does not only reduces computer costs, but in fact, it was necessary to stitch a great number of images successfully because inconsistencies appeared when tested with more than 5 images in the approach proposed in [10].

In the algorithm the names of the images are imported into the program and ordered the way they will be stitched, then they are loaded. It is important to point out that at this point each image is an RGB three-dimensional matrix, which means that this process consumes a lot of memory depending on the number of images.

B. Feature Detection

As mentioned before, this work is a feature-based stitching method. However, before begin describing this approach, it is worth mention that there is a direct alignment approach proposed in [11], which works well stitching images on sequential video frames because of its cheaper processing costs. Although, this method has a small convergence limit, which can leave non-filled gaps between images.

In contrast, the method of features detection and matching has significant robustness when there are enough features because it makes possible finding alignment between two images, and it is effective closing gaps [12], which makes it an interesting approach to the MicroScanner application. On the other hand, this method has the disadvantage of became imprecise and ineffective when there are not enough features [13].

A generic feature detection method can be divided into three main parts:

1) Patches and Keypoints Detection: Patch is known as a region around a keypoint, and they are easier to be found in regions with a big difference of contrast or defining it mathematically, it is where the gradients are bigger. In Image processing, gradient is a two variable function that results, for each pixel of the image, a vector that points out to the bigger color change [13].

2) Descriptors Extraction: In this stage the patches, already extracted, are converted and grouped to more compact and invariant forms, in other words, they are transformed in descriptors. This process consists basically of rotate, scale and sometimes apply some affine transformations in the pathces. It is important to point out that the descriptors must be distinct, robust to noise and to geometric and photometric deformations [12].

3) Descriptors Pre-Matching: Finally, a pre-selection of the descriptors is made to eliminate the bad candidates to future matches. It can be done using a threshold value to the euclidean distance between descriptors from different images. Since the descriptors are scale and rotation invariant, changing the size of the images will not change the absolute value of this distance [13].

Knowing that a feature-based approach works following the three steps presented above, it is possible to present the *Speeded-Up Robust Features* (SURF) method, a very robust, fast and effective algorithm, that uses the Hessian matrix to compute position and scale of the keypoints, and a neighborhood of 20x20 around them, to be divided into 25 sub-regions of 4x4 pixels, and for each of this is applied a 2D Haar Wavelet transformation to extract descriptors [14].

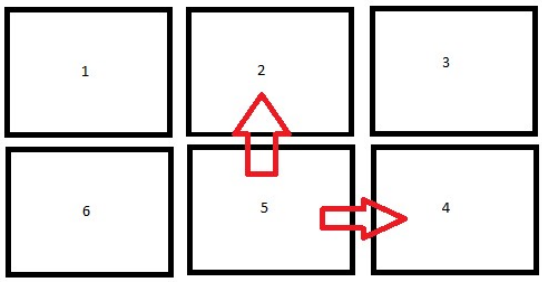


Fig. 3. Matching order, and how homographies are estimated.

One of the downsides of SURF is that it is not very robust against light variations and changes in points of view [14] when compared to the *Scale-Invariant Feature Transform* (SIFT) algorithm [15]. However, to the MicroScanner application, it is not a problem since the illumination is constant and there is no change in the point o view.

In this work, we used SURF to detect the keypoints and extract the descriptors, but the descriptors pre-matching step was implemented independently by us using the order of the pictures (Fig. 3) to reduce processing time and increase the reliability of the results. It was done in two parts. The first one analyses a pair of horizontal neighbor images; if a descriptor is inside a threshold value of pixels, above or below, it is considered a good match. Empirically was discovered that 50 is a good number, although it could be changed to attend a different set of samples. Fig. 4 illustrates how is is done.

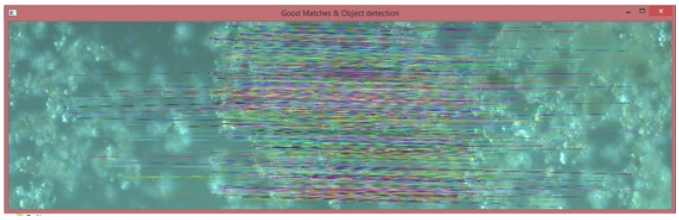


Fig. 4. Pre-matching features horizontally.

The second step consists in comparing the euclidean distance between two descriptors in vertical neighbor images if this distance is smaller than a threshold value it is considered a good match. Again, we discovered empirically that 200 pixels is a good distance to the samples used. It is worth to mention that this process of matching features between vertical neighbors could be done the same way it is done to horizontal neighbors. However, the use of the euclidean distance was used in this part to increase the robustness of the match.

C. Features Matching

To finally match the features, was used the *RANdon SAmple Consensus* (RANSAC) algorithm due to its robustness and low computing cost. This algorithm works in an opposite way when compared with other refinement algorithms because instead of using a bigger number of descriptors to have a better result, it takes a sample of *k* reliable features and use *S* attempts to obtain a probability *p* of success in each attempt, and a total probability *P* of success [16].

To represent it mathematically, it was considered the probability of *S* attempts fail, as shown in equation (1).

$$1 - P = (1 - p^k)^S \tag{1}$$

So the minimum number of attempts *S* that could give an effective result can be found with equation (2).

$$S = \frac{log(1-P)}{log(1-p^k)} \tag{2}$$

In [17] is given examples of some heuristics and the number of attempts *S* to obtain a 99% probability *P* of success. After applying a heuristic *S* times, it is possible to determine which features have correspondence, in other words, the *inliers*, and which features do not have correspondence or *outliers* [16].

D. Finding The Homography Matrix

The homography matrix is used in point transformations in a space, such as rotation, translation, and affine [13]. In 3D it can be represented as shown in equation (3).

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \times \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \Leftrightarrow Hx_1 \tag{3}$$

In this work, we need to project all images in the same composing plane and with its respective overlap. To achieve that, the homography matrix can be projected in 2D. Solving the equation (3) in homogeneous coordinates, (i.e., $x'_2 = \frac{x_2}{z_2}$ and $y'_2 = \frac{y_2}{z_2}$), equations (4) and (5) are obtained.

$$x'_2 = \frac{H_{11}.x_1 + H_{12}.y_1 + H_{13}.z_1}{H_{31}.x_1 + H_{32}.y_1 + H_{33}.z_1} \tag{4}$$

$$y'_2 = \frac{H_{21}.x_1 + H_{22}.y_1 + H_{23}.z_1}{H_{31}.x_1 + H_{32}.y_1 + H_{33}.z_1} \tag{5}$$

As can be seen, equations (4) and (5) have 9 variables, which means that it is necessary at least 3 points to calculate the homography matrix. These points are the positions of the descriptors found by SURF and selected by RANSAC. The OpenCV method *findHomography* [18] is the method used in this work to apply RANSAC and discover the *Hs*, and it returns a 2D homography matrix in homogeneous coordinates following the model shown in equation (6). Where T_x and T_y are translations in x and y respectively, and $rot + proj$ are rotations plus projections values [13].

$$H = \begin{bmatrix} rot + proj & rot + proj & T_x \\ rot + proj & rot + proj & T_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Equation (7) shows an output example from the code. As can be seen, the two first values from the third line are small enough and can be considered zero. This happens due to the numerical iterative approximations.

$$H = \begin{bmatrix} 1.00725 & -0.00313 & -1002.11567 \\ 0.00140 & 1.00156 & 23.08991 \\ 4.04535e-6 & -3.42769e-6 & 1 \end{bmatrix} \tag{7}$$

It is important to point out that for each neighbor image there is one homography matrix, which means that images are matched in pairs. Fig. 3 illustrates this process, in addition, it is worthy to explain that image 5 can have a homography matrix defined from image 1, another form image 2, another from image 3, and finally another form image 4. It means that image 5 can have its global position defined by 4 homography matrices. However, in this work, we used only two of them, one form the horizontal (i.e image 4), and another form the vertical neighbor (i.e. image 2). This reduces processing time in the next step.

E. Bundle Adjustment

In the last section was said that each image has its position defined by 2 homography matrices, which means that it is necessary a global alignment step in the process. This avoids errors mainly when the number of composing images rises up. This process is known as bundle adjustment [19]. This function works with camera intrinsic parameters and its matrix is represented in equation (8), where f_x and f_y are focal lengths in x and y axis respectively, which represents 3D distance values from the composing plane until the camera sensor. Whereas c_x and c_y determine the focal center of the camera [19].

$$H = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \tag{8}$$

At this point, it is valid to remember that each image's position was defined by homographies matrices. OpenCV have the *estimator* [20] method to transform homographies in intrinsic parameters matrices. In addition, OpenCV's bundle adjustment function considers that all images were taken by the same virtual camera, which was just rotated, and not translated [4]. To improve the results, the composing surface considered in the intrinsic parameter matrix is re-projected behind and more distant from the camera to reduce the angles between images, and consequently reducing projections errors [13]. Equation (9) shows the new form from of the intrinsic parameters matrix.

$$H = \begin{bmatrix} -af_x & 0 & c'_x \\ 0 & -bf_y & c'_y \\ 0 & 0 & 0 \end{bmatrix} \tag{9}$$

The bundle adjustment algorithm used as an after the process of feature detection suffer from two main problems. First, the presence of outliers detected as inliers by RANSAC may delay convergence of the algorithm. Second, the fact that a feature can be used to calculate more than one homography matrix, making the feature overestimated $\binom{m}{n}$ times, where m represents how many times the feature was counted, and n the number of homography matrices that is was used. This means that an outlier used to estimate a big number of homographies can produce an ineffective result.

F. Data Register

This section does not represent a step in the stitching process. However, it is worth mention that until this point all that was done was collect data from the images, and calculate how it could be composed, although, nothing was projected in no surface. The next sections will discuss the composing parts.

If the quality of the images were to be reduced to save memory, it is interesting to use an anti-aliasing filter to remove distortions generated in the compressing process [13]. Although, we did not use this filter in this work because what is intended to be obtained is an image bigger than the original.

One last thing worth comment is that a wave correction [10] step could be used before start composing. It is useful to correct accidental camera inclinations occurred when a picture is being taken by a photographer. Furthermore, MicroScanner's camera stays steady all the time, eliminating this way the necessity of this step.

G. Defining Projection Surface

Choosing a projection surface is a process that involves keeping local appearance and, at the same time, obtain a uniform composition. For a microscopy application a flat surface is the one which better represents the environment photographed, and the one used in this work. However, when the number of the images increase in this plane surface it is difficult to maintain the local appearance without stretching too much the images. This size expansion occurs mainly when there is no much matched features in the composition, and its main consequence is memory overflow, which crashes the software. In situations like these sometimes a solution could be the use of a cylindrical or spherical surface [13].

H. Seam Correction

To smooth seam marks in overlapped regions OpenCV estimates masks just for these regions and apply a seam correction algorithm [21] just on this masks, to reduce processing time. It works computing differences between pixels and decides what erase, such as objects that moved leaving ghost effects, and what to keep. This is done basically applying a weighted arithmetic mean leaving the more intense pixels [21]. This step is necessary because living things in the sample analyzed can move, and it also helps reduce low features correspondence issues in the final mosaic.

I. Blending

After the correction of seam marks, there is still the need to correct exposure differences and some remaining problems where there is low features correspondence. In this step was used the multi-blending process [22], which uses a Laplacian pyramid where each level selects a frequency band to smooth visual images differences. The main advantage of this algorithm is that it can act in low or high frequencies independently, and the number of bands is defined automatically by the OpenCV blending algorithm [4]. Differently from the seam correction step that uses masks, the blending process is applied in the entire image [22].

In case the exposure differences are too big, algorithms such as [23] can help improve the results. However, illumination and exposure time of the pictures were constant and there were no need to use it.

III. Results

In this work, the samples which the images were taken are from an industrial sewer, and in the results presented were used the 100 images shown in Fig. 1, each with resolution of 2MP (1920x1080), the objective magnification value of the MicroScanner's lens was 10x, and the computer used to

process these images was an intel i5 2GHz with a cache of 3MB and 8GB of RAM memory.

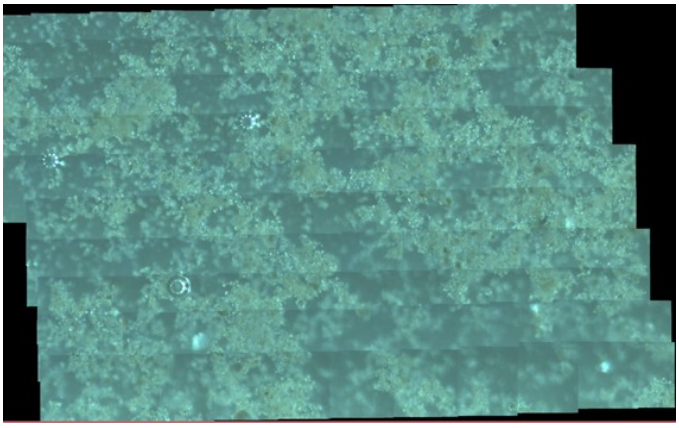


Fig. 5. 10x10 image projection having its position defined only by one homography matrix form its previous neighbor. It was not used bundle adjustment, seam correction or blending.

The MicroScanner movement followed the same idea presented *Image Acquisition* section. Initially, in Fig. 5 is shown a projection of these images using only one homography matrix per image from its previous neighbor to determine its position. Images were just warped and projected into a flat surface without bundle adjustment, seam correction or blending. The idea used in this process was similar the one used in Fig. 6 [7], but it clearly shows problems with features matching, in addition, it presents seam marks and exposure time differences and gaps between images.

Fig. 6. Example of a result obtained in [7].

Improving the results, Fig. 7a shows a mosaic created with 20 images, using bundle adjustment, but neither seam correction nor blending were used. Fig. 7c gives a zoom in Fig. 7a and shows that due to bundle adjustment step, images are better positioned and warped when compared with Fig. 5.

Finally, Fig. 7b presents an ideal result, where were used the bundle adjustment, seam correction, and blending steps. Fig. 7d gives a zoom in Fig. 7b. When Fig. 7d is compared with Fig. 7c it is possible to see how smoother transitions between images are, and how little features misplacement are corrected in Fig. 7d.

However, when the number of features is increased the final mosaics start to get more distorted as can be seen in Fig. 8a. The main reason for this problem is a low number of features matched in at least one of the images. This makes the bundle adjustment algorithm stretch too much some images, the ones on the left, in this case, to close gaps between them [13].

The stitching process although, involves an artistic effort as much as a computational one [13]. Motivated by this affirmation, an attempt to reduce the vertical euclidean distance parameter from 200 to 170 pixels returned the Fig. 8b, which is a much more acceptable result. This happened because when this threshold value was reduced it reduces the number of pre-selected features sent to RANSAC match, and as it works better when there is a more reliable sample of features instead of a bigger number of then, the final mosaic consequently looks better.

Finally, it is important to discuss the algorithm's convergence time, which still has space to get smaller. This gets worse as the number of images is increased. We observed that at least 70% of the time spent by our stitching code is in the bundle adjustment step, and this proportion rate gets bigger when the number of images increases. This happens because OpenCV's bundle adjustment function does not allow varying its internal convergence parameters. A possible solution and suggestion to future work is the use of CERES library [24], which have a more flexible bundle adjustment function, to implement this part of the code.

IV. Conclusion

Based on the data and analyzes presented, it is possible to affirm that the creation of feature-based mosaics is a medium to high complexity task, especially when the number of images to be stitched increases, and that this process involves an artistic and empirical effort as much as computational one.

The several techniques of image processing and computer vision used in this work were presented and discussed, such as feature detection and matching, homography estimation, bundle adjustment, seam correction, and blending.

Our work presents an approach specialized to the MicroScanner device which is a product that promises reduce equipment and labor costs, as well as easily deal with sanitary barriers issues. As initially proposed, the results obtained do not reduce the quality from original images, and have better quality and more effective results than previous presented works, especially for smaller sets of images. Finally, it solves the issue between field of view and resolution.

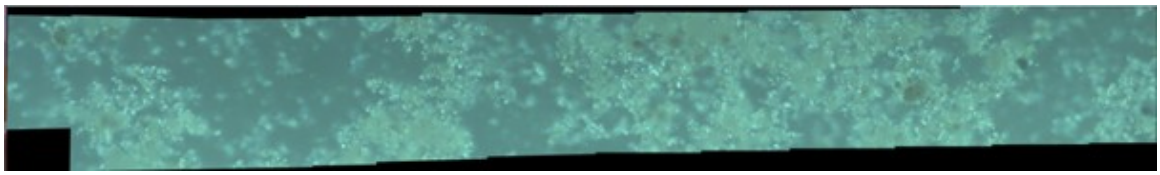
Acknowledgment

The authors would like to thank the company Mogai Information Technology for the partnership and sponsorship of this work.

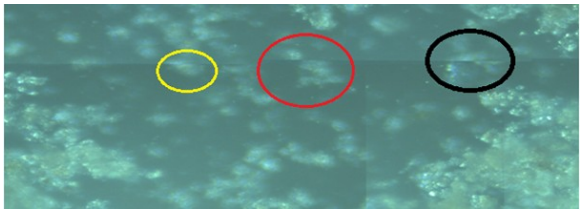
References

[1] (2016, Jun.) Resolving power of microscopes. [Online]. Available: http://sciencelearn.org.nz/Contexts/Exploring-with-Microscopes/Sci-Media/Images/Resolving-power-of-microscopes/

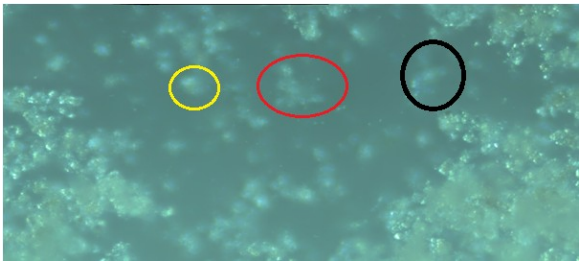
(a) 2x10 mosaic, using bundle adjustment,but without seam correction and blending. Total time: 31.34min. Bundle adjustment time: 27.77min. Size: 13.15MP (11,360x1,158).



(b) 2x10 mosaic, using bundle adjustment, seam correction and blending. Total time: 32.48min. Bundle adjustment time: 27.77min. Size: 17.42MP (11,130x1,579).

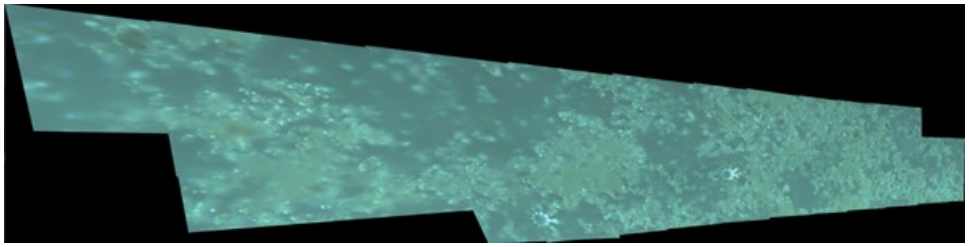


(c) Zoon in Fig. 7a, transitions not smooth.

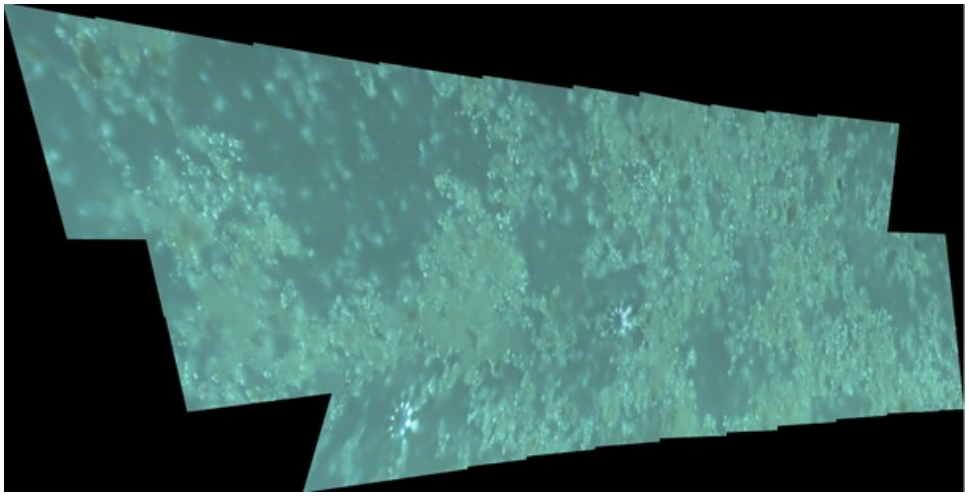


(d) Zoon in Fig. 7b, smooth transitions.

Fig. 7. 2x10 mosaic result and analysis of bundle adjustment, seam correction and blending use.



(a) 4x10 mosaic,vertical eucidian distance 200, using bundle adjustment, seam correction and blending. Total time: 162.80 min. Bundle adjustment time: 153.49 min. Size: 184.23MP (27,101x6,798).



(b) 4x10 mosaic, vertical eucidian distance 170, using bundle adjustment, seam correction and blending. Total time: 166.87 min. Bundle adjustment time: 156.91 min. Size: 164.33MP (17,893x9,184).

Fig. 8. 4x10 mosaic result and analysis of bundle adjustment, seam correction and blending use.

[2] B. Potsaid, Y. Bellouard, and J. T. Wen, "Design of an adaptive scanning optical microscope for simultaneous large field of view and high resolution," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 460–465.

[3] (2016) Leica. [Online]. Available: https://www.leica-microsystems.com/science-lab/

[4] (2016, Sep.) Opencv. [Online]. Available: http://opencv.org/

[5] D. L. Milgram, "Computer methods for creating photomosaics," *IEEE Transactions on Computers*, vol. 100, no. 11, pp. 1113–1119, 1975.

[6] M. Brown and D. G. Lowe, "Recognising panoramas," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 1218–1225 vol.2.

[7] L. A. Amorin, L. d. A. Silva, F. M. Queiroz, R. d. M. Salvador, R. F. Vassalo, and M. F. Sarcinelli, "Construção de mosaico utilizandio imagens aéreas adquiridas de forma autonomatica," *XII Simpósio Brasileiro de Automação Inteligente*, 2015.

[8] J. Chalfoun, M. Majurski, T. Blattner, W. Keyrouz, P. Bajcsy, and M. Brady, "Mist: Microscopy image stitching tool," in *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1757–1757.

[9] S. E. Chen, "Quicktime vr: An image-based approach to virtual environment navigation," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 29–38.

[10] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[11] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 251–258.

[12] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 510–517.

[13] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[14] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[15] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] C. V. Stewart, "Robust parameter estimation in computer vision," *SIAM review*, vol. 41, no. 3, pp. 513–537, 1999.

[18] (2016, Sep.) Opencv api reference. [Online]. Available: http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[20] (2016, Sep.) Camera calibration and 3d reconstruction. [Online]. Available: http://docs.opencv.org/2.4/modules/core/doc/intro.html

[21] M. Uyttendaele, A. Eden, and R. Skeliski, "Eliminating ghosting and exposure artifacts in image mosaics," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2. IEEE, 2001, pp. II–II.

[22] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics (TOG)*, vol. 2, no. 4, pp. 217–236, 1983.

[23] M. T. Uyttendaele and R. S. Szeliski, "System and method for exposure compensation," Nov. 2 2004, uS Patent 6,813,391.

[24] (2016, Sep.) Ceres. [Online]. Available: http://ceres-solver.org/